DGK Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 901

Dennis Wittich

Deep Domain Adaptation for the Pixel-wise Classification of Aerial and Satellite Images

München 2023

Bayerische Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5313-7

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 386, Hannover 2023

DGK Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 901

Deep Domain Adaptation for the Pixel-wise Classification of Aerial and Satellite Images

Von der Fakultät für Bauingenieurwesen und Geodäsie
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

von

Dennis Cyrill Wittich, M. Sc.

Geboren am 28.04.1990 in Konstanz

München 2023

Bayerische Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5313-7

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 386, Hannover 2023

Adresse der DGK:

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften (DGK)

Alfons-Goppel-Straße 11 ● D – 80 539 München

Telefon +49 – 331 – 288 1685 ● E-Mail post@dgk.badw.de

http://www.dgk.badw.de

Prüfungskommission:

Vorsitzender: Prof. Dr.-Ing. habil. Christian Heipke

Referent: apl. Prof. Dr. techn. Franz Rottensteiner

Korreferenten: apl. Prof. Dr.-Ing. Claus Brenner
Prof. Dr. Konrad Schindler

Tag der mündlichen Prüfung: 23.05.2023

ISSN 0065-5325 ISBN 978-3-7696-5313-7

Abstract

This thesis addresses domain adaptation for the pixel-wise classification of remotely sensed data using deep neural networks (DNN) as a strategy to reduce the requirements of DNNs with respect to the availability of training data. The focus is set on a setting in which labelled data are only available in a source domain D^S, but not in a target domain D^T, referred to as unsupervised domain adaptation (UDA) in computer vision. The two domains are assumed to be different but related. A new method is proposed that is based on adversarial training of an appearance adaptation network (AAN) that modifies images from D^S such that they look like images from D^T. Together with the original label maps from D^S, the adapted images are used to train the DNN so that it performs well in D^T. The AAN has to change the appearance of objects of a certain class such that they resemble objects of the same class in D^T, i.e. the appearance adaptation has to be *semantically consistent*. Many approaches try to achieve this goal by incorporating cycle consistency in the adaptation process, but such approaches tend to hallucinate structures that occur more often in D^T. In contrast, in this thesis, a joint training strategy of the AAN and the classifier is proposed, which constrains the AAN to adapt the images such that they are correctly classified. To further improve the performance of the classifier after UDA, two extensions are proposed, both aiming to improve the appearance adaptation with respect to the semantic consistency. Furthermore, the problem of finding the optimal values of the trained network parameters is addressed, proposing a new unsupervised entropy based parameter selection criterion, which compensates for the fact that there is no validation set in D^T that could be monitored during UDA. In a further variant, the new method is combined with an existing method for UDA from the literature referred to as *adaptive batch normalisation*. Different variants of the method are extensively evaluated in 20 adaptation scenarios related to the application of land cover classification based on aerial imagery, using datasets from 5 cities, all consisting of high-resolution digital orthophotos and height data. The method is further evaluated in 6 adaptation scenarios related to the application of bi-temporal deforestation detection based on pairs of satellite images. The proposed method achieves a positive transfer in most cases. On average, it can improve the performance in the target domains compared to the performance of classifiers that were trained only in D^S. For land cover classification, the initial performance gap of 10.7 % can be reduced to 7.5 % in the mean F_1 score, and for bi-temporal deforestation detection the performance gap is reduced from 35.8 % to 11.3 % in the F_1 score of the foreground class. In a few cases the method even achieves a performance that is comparable to the one achieved by a classifier trained in D^T. In adaptation scenarios between the Vaihingen and Potsdam datasets from the ISPRS semantic labelling benchmark the method outperforms others from recent publications by about $5 - 20$ % with respect to the mean F_1 score.

Keywords: Domain Adaptation, Pixel-wise Classification, Deep Learning, Aerial Images, Remote Sensing, Appearance Adaptation

Kurzfassung

Die vorliegende Dissertation befasst sich mit der Domänenadaption für die pixelweise Klassifikation von Fernerkundungsdaten unter Verwendung von tiefen neuronalen Netzen (DNN) als Strategie zur Verringerung der Anforderungen von DNNs in Bezug auf die Verfügbarkeit von Trainingsdaten. Der Schwerpunkt liegt auf dem Szenario, in dem gelabelte Daten nur in einer Quelldomäne D^S, aber nicht in einer Zieldomäne D^T verfügbar sind, was in der Computer Vision als unüberwachte Domänenanpassung (UDA) bekannt ist. Hier wird davon ausgegangen, dass die beiden Domänen unterschiedlich, aber verwandt sind. Es wird eine neue Methode vorgeschlagen, in der ein Netzwerk zur Anpassung der Erscheinungsform (AAN) trainiert wird, welches Bilder aus D^S so anpasst, dass sie wie Bilder aus D^T aussehen. Zusammen mit den ursprünglichen Labels aus D^S werden die adaptierten Bilder verwendet, um ein DNN an D^T anzupassen. Das AAN muss das Aussehen von Objekten einer bestimmten Klasse so verändern, dass sie Objekten der gleichen Klasse in D^T ähneln, d.h. die Anpassung muss *semantisch konsistent* sein. Viele Ansätze versuchen, dieses Ziel zu erreichen, indem sie die sogenannte *cycle consistency* in den Anpassungsprozess einbeziehen, aber solche Ansätze neigen dazu, Strukturen zu halluzinieren, die in der Zieldomäne häufiger vorkommen. Im Gegensatz dazu wird in dieser Arbeit eine gemeinsame Trainingsstrategie für das AAN und den Klassifikator vorgeschlagen, die das AAN dazu bringt, die Bilder so anzupassen, dass diese korrekt klassifiziert werden. Um das AAN in Bezug auf die semantische Konsistenz weiter zu verbessern, werden zwei Erweiterungen der Methode vorgeschlagen. Außerdem wird das Problem der optimalen Wahl der Parameter der trainierten Netzwerke adressiert. Hierzu wird ein neues unüberwachtes Kriterium zur Auswahl der Parameter basierend auf der Entropie in D^T vorgeschlagen, das die Tatsache kompensiert, dass es in D^T keinen Validierungsdatensatz gibt, der zur Parameterwahl genutzt werden könnte. In einer weiteren Variante wird die neue Methode mit einer bestehenden Methode für UDA aus der Literatur kombiniert, die als *adaptive batch normalisation* bekannt ist. Verschiedene Varianten der Methode werden in 20 Adaptionsszenarien für die Aufgabe der Landbedeckungsklassifikation mit Luftbildern umfassend evaluiert, wobei Datensätze aus 5 Städten verwendet werden, die alle aus hochauflösenden digitalen Orthophotos und Höhendaten bestehen. Die Methode wird außerdem in 6 Adaptionsszenarien für die Detektion von Rodungen auf der Grundlage von Satellitenbildpaaren bewertet. Die Methode erzielt in den meisten Fällen eine Verbesserung der Klassifikationsgüte. Bei der Klassifizierung der Bodenbedeckung kann die anfängliche Lücke in der Klassifikationsgüte von $10,7\,\%$ auf $7,5\,\%$ im mittleren F_1-Maß und bei der bi-temporalen Entwaldungserkennung von $35,8\,\%$ auf $11,3\,\%$ im F_1-Maß der Vordergrundklasse reduziert werden. In einigen Fällen erreicht die Methode sogar Qualitätsmaße, die vergleichbar mit jenen sind, die ein Klassifikator erreicht, der in D^T trainiert wurde. Bei der Adaption zwischen den Datensätzen der ISPRS-Benchmark aus Vaihingen und Potsdam übertrifft die vorgeschlagene Methode Verfahren aus neueren Veröffentlichungen um ca. $5 - 20\,\%$ im mittleren F_1-Maß.

Nomenclature

Abbreviations

RS	Remote sensing
MSI, nDSM	Multi-spectral image, normalised digital surface model
LCC, BDD	Land cover classification, bi-temporal deforestation detection
ML, DL	Machine Learning, Deep Learning
TL, DA, UDA	Transfer learning, domain adaptation, unsupervised domain adaptation
NN, DNN	Neural network, deep neural network
CNN, FCN	Convolutional neural network, fully convolutional neural network
BN	Batch normalization
ABN	Adaptive batch normalization
ReLU	Rectified linear unit
LReLU	Leaky rectified linear unit
MB-SGD	Mini-batch stochastic gradient descent
MB-SGD-M	Mini-batch stochastic gradient descent with momentum
ADAM	Adaptive momentum

Symbols

Images and Image Classification

x, X	Pixel intensity value, image
y, Y	class label, label map
$\mathcal{X}, \mathcal{Y}$	image space, label map space
$\hat{y}, \hat{Y}$	Predicted class label, predicted label map
γ, Γ	One-hot encoded label, one-hot encoded label map
$\hat{\gamma}, \hat{\Gamma}$	Predicted probability vector, predicted probability map (multi-class classification)
ψ, Ψ	Predicted probability, predicted probability map (binary classification)
h, w, d	Height, width, depth (of an image, matrix or map)
r, c, q	Indices for row, column and channel
S, n_L	Class structure, number of classes
n_C	Number of input channels in a classification setting

Datasets

T, n_T	Labelled data set, number of samples in T
U, n_U	Unlabelled data set, number of samples in U

Deep Learning and CNN

A	Activation map
K	Kernel matrix
$\mathbf{v}$	Feature vector
a	Activation (output of single neuron)
u	Output of neuron before activation
$\mathbf{a}$	Activation vector
Θ	Parameter set
θ	Parameter
$f(\cdot)$	Activation function or normalisation function
$\mathcal{L}(\cdot)$	Loss function
λ	Learning rate
n_B	Batch size
p	Side length of image patches
f_a	Activation function
W	Context window

Domain Adaptation

D^S, D^T	Source domain, target domain
$\mathcal{T}$	Learning task

Proposed Method for Appearance Adaptation

$\mathcal{C}$	Classification network
$\mathcal{A}$	Appearance adaptation network
$\mathcal{G}$	Auxiliary image generation network
$\mathcal{D}$	Discriminator network

General

μ, σ	Mean, standard-deviation
i, j, k	Indices
ω	Weight (of loss or class)

Contents

1 Introduction

1.1 Motivation

The task of pixel-wise image classification is to assign a class label to each pixel in an image according to a pre-defined class structure (Guo et al., 2018). In remote sensing (RS) applications, the images are usually georectified multispectral images (MSI) originally acquired from an airborne platform or a satellite. Pixel-wise elevation information is often also available, e.g., obtained by 3D-reconstruction from overlapping aerial or satellite images. In this thesis, two applications for pixel-wise classification in RS are addressed. The first application is land cover classification (LCC) based on airborne imagery and elevation information. Here, the goal is to classify each pixel in the aerial image according to a set of land cover classes, such as *vegetation*, *building* and *road*. This is, for example, useful for automatically updating maps or localising changes in the land cover. The second application is bi-temporal deforestation detection (BDD). In this application, the input to the classifier consists of a pair of images of the same area, and the desired output is a binary map that indicates whether deforestation has happened between the acquisition dates of the images at each pixel location. Such an automated classifier is useful for national or international organisations that track legal and illegal deforestation activities (Assis et al., 2019).

For several years, research on the topic of pixel-wise classification has been dominated by supervised classification methods based on Deep Learning (DL), in particular Fully Convolutional Neural Networks (FCNs) (Long et al., 2015a), e.g. (Marmanis et al., 2016; Zhang et al., 2019a). Here, a classifier is trained using training samples, i.e. images for which the reference label maps are available. The requirement of FCNs for the availability of a large set of training samples is one of the main problems related to DL in RS, as the generation of training samples often requires a labour-intensive interactive labelling process that is costly and time-consuming and should be avoided if possible (Zhu et al., 2017b). If too few labelled images are available for training, it is very likely that the trained classifier will perform poorly on unseen data, thus, the classifier will have a poor generalization ability. In particular, when training on too small a dataset, the classifier can overfit to the training data, which means that it rather memorises the training examples instead of learning to make predictions based on patterns that are also to be expected in other scenes. However, training on a large dataset does not necessarily lead to a high performance on unseen data. It is also important that the training data are representative for the classification task, i.e. that the patterns which are seen during training should also occur in the images that are to be classified after training, in which case the classifier is transferable to other data.

In the literature, many strategies have been proposed to increase the generalization ability of a classifier without having to manually label additional data. A common strategy to avoid overfit-

ting is to regularise the classifier explicitly by penalising numerically large parameter values during training. There are also methods for implicit regularisation, such as dropout regularisation (Srivastava et al., 2014). Another common strategy to increase the generalization ability of a classifier is to artificially increase the diversity of the training data, e.g. by data augmentation (Shorten and Khoshgoftaar, 2019).

An alternative strategy to those already mentioned is *Transfer Learning* (TL) (Pan and Yang, 2009). Here, the data are assumed to be available in different domains. The goal of TL is to transfer knowledge from a source domain (D^S) in which training samples are abundant to a target domain (D^T) in which only a limited amount or no training data are available. TL often follows a two-step procedure. In the first step, a classifier is trained using the data from D^S. In the second step, the classifier is adapted to the target domain, using the available data in D^T. A special setting of TL is *Domain Adaptation* (*DA*) (Tuia et al., 2016; Wang and Deng, 2018). Here, the domains share the same feature space, but the data may follow different distributions. This is a standard assumption in RS (Tuia et al., 2016); methods that assume the feature spaces of all domains to be identical are called homogeneous DA methods in computer vision (Wang and Deng, 2018). In both domains, there is a learning task to be solved that is characterized by the same class structure. Regarding the task of land cover classification, this corresponds, for example, to a situation in which labelled images from one area (source domain) are to be used to train a classifier which is then used to classify images of another area (target domain) that were acquired with a sensor of the same type, considering the same class structure. However, the objects in the target domain may have a different appearance, for example due to a different capturing season or due to regional differences. In such a case, performing a naïve transfer, i.e. training a classifier on source domain data only and applying it to D^T without any adaptation may lead to poor results. In particular, the results are much worse compared to the performance of a classifier that was trained in the target domain. This effect is referred to as *performance gap* while the difference between the domains causing the performance gap is commonly called *domain gap* (Xu et al., 2022; Zhao et al., 2023).

A special setting of DA is known as *unsupervised DA* (UDA) in computer vision (Wang and Deng, 2018), while in RS it is sometimes referred to as *semi-supervised DA* (Tuia et al., 2016). Here, no training labels are available in D^T, thus, only the images from D^T are used to adapt a classifier from D^S to D^T. This setting is particularly interesting because images from D^T are always available as they are to be classified in the first place. The main goal of UDA is to use the information available in D^S to find a better solution for the task in D^T, which requires the domains to be related (Pan and Yang, 2009). In (Tuia et al., 2016) this requirement is concretised by requiring the knowledge of the classifier trained in D^S to be sufficient, although not perfect when applied to D^T. Measuring this requirement is, however, difficult, because it would require access to the reference label maps in D^T, which are not available in the UDA setting. UDA is known to be a difficult task and can even lead to a *negative transfer* (Xu et al., 2022), that is, a lower performance in the target domain after adaptation compared to training on data from the source domain only. The opposite case, in which the adapted classifier outperforms the classifier trained on the source domain only is called a *positive transfer* (Wang and Deng, 2018). UDA is of great importance in RS because, on the one hand, there is a very limited amount of freely available data with annotations (Zhu et al., 2017b) and, on the other hand, the appearance of both natural and man-made objects in remotely sensed

images has a large variability, which makes it difficult for a classifier to generalize well when applied to new domains. These factors can lead to a large performance gap. Although this is certainly a major challenge, recent advances in the development of methods for UDA in related fields show that such methods can compensate for the existing domain gap to some extent.

In the RS applications LCC and BDD, the domains may be associated with images from different geographical regions or from different points in time, but with the same input channels. The source domain corresponds to images for which pixel-level class labels are known, e.g. from previous projects, while the target domain corresponds to a new set of images to be classified according to the same class structure. In the UDA scenario, considered in this thesis, this is to be achieved without having to generate (new) training labels in D^T, even though there may be a domain gap, i.e. even though the distributions of the data in the two domains are different (Tuia et al., 2016). This domain gap may result in a performance gap, i.e. a classifier trained using the labelled data from D^S will perform worse in D^T compared to a classifier trained using labelled data from D^T. The goal of UDA is to reduce this performance gap, i.e. to achieve a positive transfer, while at the same time avoiding a negative transfer. Practically, such a method enables to reuse existing training data for the automated classification of new domains that are different but related without having to generate additional labels. Consequently, UDA has a very high potential to reduce labelling effort and, thus, to save time and costs.

UDA is a very active field of research and many methods have been proposed. However, to the best of the author's knowledge there is no method in the literature which could fully compensate for the considerable domain gap occurring in RS applications: even if a positive transfer is achieved, the adapted classifier still performs significantly worse compared to a classifier that was trained using labelled target domain data, e.g. (Peng et al., 2022; Soto et al., 2021; Ji et al., 2020). Of course, training a classifier in the target domain is not possible in a real UDA scenario, but it is frequently done in research to assess the performance of an adapted classifier, e.g. in (Peng et al., 2022; Soto et al., 2021; Ji et al., 2020). There is also a considerable amount of research on UDA in the field of computer vision. In this context, a commonly addressed domain adaptation scenario is to adapt a classifier trained on synthetic images for pixel-wise street scene classification to real images, e.g. (Hoffman et al., 2018; Zhang et al., 2018a). There are many such methods, and they often achieve a very good classification performance after adaptation. Unfortunately, methods that work well in this scenario are often not transferable to other applications, in which the domain gap is of a different nature. Many of these methods make explicit or implicit assumptions about the label distributions in the source and target domains, for example, that the label distribution is similar in both domains (Zhang et al., 2017; Hoffman et al., 2016; Huang et al., 2018a). Such assumptions cannot generally be made in RS, where the label distributions may be very different between the two domains. Without proper modifications, methods that were developed in for the application of street scene classification may result in a strong negative transfer in RS applications, which was e.g. shown in (Soto et al., 2021). The lack of methods that reliably overcome the domain gap in RS applications is the main motivation for this work presented in this thesis.

In general, there are different strategies for UDA (Xu et al., 2022; Tuia et al., 2016; Wang and Deng, 2018). Methods based on *instance transfer* start with training a classifier using data from

D^S. This classifier is then applied to images from D^T to predict pixel-wise labels, referred to as *semi-labels*, and the classifier is re-trained using images from the target domain with the semi-labels in an iterative process. Approaches based on this strategy have the disadvantage that they assume a very good initial performance of the (source) classifier in D^T, which may not be guaranteed after training in D^S. The second strategy for UDA, often used in the context of Deep Learning (Wang and Deng, 2018), is based on *representation transfer*. Such methods map images from both domains to a common and domain-invariant representation space in which a classifier trained with samples from D^S can also be used to classify data from D^T, e.g. (Tzeng et al., 2017; Liu et al., 2020). However, it has been shown that such an approach is quite difficult to train and can often lead to negative transfer (Wittich and Rottensteiner, 2019; Gritzner and Ostermann, 2020; Tsai et al., 2018; Zhao et al., 2019).

Consequently, the main approach presented in this thesis follows another strategy, referred to in this work as *appearance adaptation*, based on methods for style transfer (Zhu et al., 2017a; Liu et al., 2017); this task is also called *image-to-image translation* in computer vision, but will be referred to as appearance adaptation in this thesis. Appearance adaptation is the task of creating a modified version of an image from one domain such that the appearance of objects in the adapted image is similar to the appearance of objects in images from another domain. A common approach to perform appearance adaptation is based on adversarial training of two networks: an *appearance adaptation network* that learns to adapt images from D^S to look like images from D^T, and a *domain discriminator* that learns to predict whether an image came from D^T or was generated by the appearance adaptation network (Isola et al., 2017). Because the label information is not changed in the appearance adaptation, the adapted source images with known labels can be used to re-train a classifier for D^T and, thus, to perform the domain adaptation. Such a strategy was originally applied to street scene segmentation (Hoffman et al., 2018; Zhang et al., 2018a), examples for its application in RS are (Benjdira et al., 2019; Tasar et al., 2020a,b; Li et al., 2020b; Soto et al., 2021; Gritzner and Ostermann, 2020; Zhao et al., 2023). Alternatively, the images from the target domain can be adapted to look like those from the source domain, which allows to present them to a classifier that was trained in the source domain. This strategy is e.g. used in (Soto et al., 2020). The method proposed in this thesis is based on the first strategy but uses a novel training scheme. Unlike existing approaches, only a single appearance adaptation network that adapts images from D^S to D^T is used, which is jointly trained with the classifier.

The main challenge of appearance adaptation is to produce images that are representative for D^T but are *semantically consistent* at the same time (Tasar et al., 2020b): it is not sufficient that the adapted images give a similar overall impression as the images from D^T, but the appearance of objects must be adapted in such a way that after the image adaptation objects of a certain class look similar to the objects of the same class in D^T (e.g., pixels corresponding to buildings should look like buildings in D^T after the adaptation). Achieving semantic consistency is particularly difficult when the distributions of labels are very different in D^S and D^T (Soto et al., 2020; Gritzner and Ostermann, 2020). Soto et al. (2020) tried to solve this problem by training the appearance adaptation network before training the classifier, applying the *cycle consistency* constraint used in CycleGAN (Zhu et al., 2017a), in which images adapted from D^S to D^T and back again using a second adaptation network must have the same grey values as the original images. This CycleGAN-

based approach resulted in artefacts in the adapted images, a problem known as *hallucination of features* (Cohen et al., 2018). Gritzner and Ostermann (2020) also trained a CycleGAN before training the classifier and attempted to fit the label distributions based on label maps predicted in D^T using the classifier trained in D^S, but this did not lead to significant improvements in UDA. Approaches that attempt to constrain the label distribution in D^T to be similar to the one in D^S, e.g. (Zhang et al., 2017), may even be detrimental in RS, where real differences in label distributions are present. Consequently, achieving semantic consistency remains an unsolved problem in UDA based on appearance adaptation, especially when there are large differences in the distributions of labels in D^S and D^T. An example for semantically consistent and inconsistent image adaptations is shown in Figure 1.1. In the semantically consistent adaptation, the buildings were adapted such that they look like buildings in D^T. In contrast, in the semantically inconsistent adaptation, most of the buildings were adapted such that they look like trees in D^T. This thesis proposes solutions for this problem of appearance adaptation, aiming to increase the performance of UDA based on appearance adaptation.

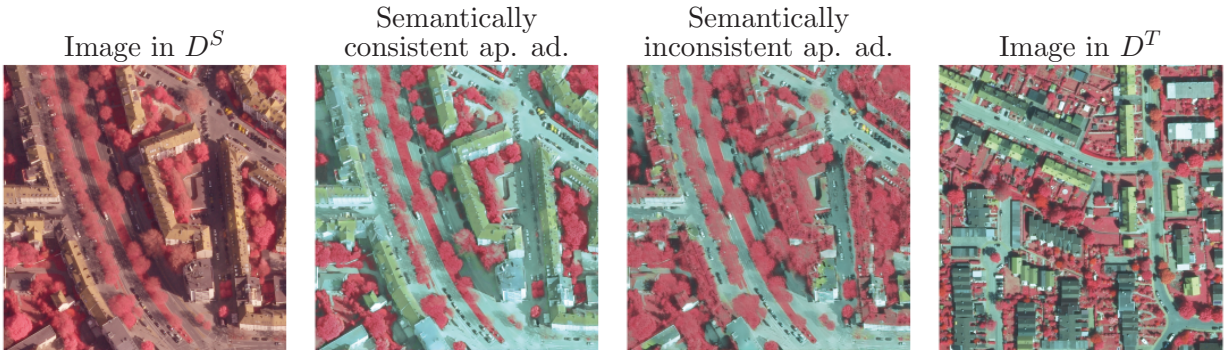

Figure 1.1: Example of semantically consistent and inconsistent appearance adaptation (ap. ad.). The right image serves as an example for appearance of objects in D^T. In the semantically inconsistent adaptation, after adaptation the buildings look like trees in D^T. After applying the appearance adaptation in a semantically consistent way, the buildings look like buildings in D^T.

Another unsolved problem in UDA is the problem of *parameter selection*, i.e. the selection of the values of the network parameters in the adaptation process to be used for classification. Parameter selection can also be seen as the problem of selecting a termination criterion for training. This aspect is not only relevant for methods based on appearance adaptation, but also for methods that are based on other strategies. Gritzner and Ostermann (2020) report that the classification error in D^T shows large fluctuations in the UDA process and may even increase after reaching a minimum, a behaviour what was also observed in (Wittich and Rottensteiner, 2019; Wittich, 2020). In classical machine learning and in supervised deep learning approaches, a validation dataset is often used to select an appropriate epoch to terminate the training process (Prechelt, 1998). However, in the addressed UDA scenario, there are no labelled samples in D^T and hence there is no validation set. That is why in UDA, the number of training epochs is often set empirically without proper reasoning, e.g. (Tasar et al., 2020b; Benjdira et al., 2019; Musto and Zinelli, 2020). This strategy is considered to be very problematic as it is not unlikely to result in a negative transfer. In principle, if multiple domains with labelled data are available at training time, the number of epochs could be selected based on the performance of UDA on these domains. However, there is no guarantee

that the optimal number of training epochs in one UDA scenario is also optimal for other pairs of source and target domains. Consequently, the parameter selection is considered an important, yet unsolved problem in UDA. In this thesis, an approach for parameter selection is proposed that does not require labelled data in the target domain.

1.2 Contributions and Scientific Goals of this Thesis

The aim of this thesis is to solve the aforementioned unsolved problems and to develop an approach for UDA that achieves a stable reduction of the performance gap for domain adaptation scenarios in the context of RS. To that end, a new method for UDA based on appearance adaptation is proposed. The new method for appearance adaptation applies adversarial training (Goodfellow et al., 2014), where a discriminator network learns to distinguish adapted source images from real target images, while the appearance adaptation network learns to deceive the discriminator by adapting the images such that they look like images from D^T (Zhu et al., 2017a). However, unlike most existing approaches, no cycle consistency nor variants of this approach are used to constrain the adaptation, because they have been shown to fail in challenging adaptation scenarios, for example in the context of deforestation detection (Soto et al., 2021) or land cover classification (Ji et al., 2020). Instead, semantic consistency is achieved by *joint training of the networks for appearance adaptation and classification*. Thus, the appearance adaptation network learns to adapt images from D^S to look like images from D^T, but the adapted images must also be correctly classified, which is assumed to be an appropriate way to avoid regions belonging to a certain class in the original images to look like regions corresponding to a different class in D^T after the adaptation. To compensate for the negative effects of different label distributions in D^S and D^T, two methods are proposed to extend the adversarial training scheme. While the first method consists of a regularisation of the discriminator output, the second one involves an *auxiliary image generation network* in the training process.

The proposed method for UDA based on appearance adaptation is further combined with a method based on representation transfer. That method, referred to as *adaptive batch normalisation* (Li et al., 2018), aims to align the distributions of activation maps by adjusting the parameters of the batch normalisation layers using the images from the target domain. Combining the two methods is motivated by the assumption that they can complement each other and result in a even higher performance after UDA compared to using only one of the methods. Finally, to solve the parameter selection problem, a termination criterion is proposed that does not require reference labels in D^T, but instead takes into account the entropy of the predictions in D^T. The contributions of this work can be summarised as follows:

1. A new approach for UDA is presented that follows the strategy of appearance adaptation but uses a novel training scheme. It is based on semantically consistent appearance adaptation and relies on domain adversarial training of an appearance adaptation network jointly with the classification network. This joint training procedure is crucial as it allows to consider the predicted label maps of the adapted images while training the appearance adaptation network. The proposed approach requires only a single adaptation network, used to adapt

images from D^S to D^T. As a result, this approach is less memory demanding and easier to tune compared to methods based on cycle consistency (Zhu et al., 2017a). Compared to (Murez et al., 2018a), which is considered to be the most similar method to the proposed one and which also applies joint training of the appearance adaptation and the classification networks, the proposed one is more simple, because fewer loss terms and fewer networks are used.

2. To further improve semantic consistency, two methods are introduced that extend adversarial training. Both methods aim to mitigate the problems due to large differences in the label distributions in D^S and D^T. While in the first method a regularization of the discriminator output is employed to prevent the discriminator from learning trivial solutions for discriminating samples from different domains, in the second one an auxiliary generator is introduced that relaxes the constraint posed by the adversarial training scheme. To the author's best knowledge, the first method has solely been investigated in his own work and the second one has not been proposed in the literature so far.

3. As an extension, the proposed method based on appearance adaptation is combined with adaptive batch normalisation, a method for UDA from the literature (Li et al., 2018) that follows the strategy of representation transfer. Combining the methods is motivated by the assumption that they can complement each other, because they are based on different strategies. In practice, adaptive batch normalisation is optionally applied as a subsequent adaptation step after having adapted the classifiers using the proposed method for UDA based on appearance adaptation.

4. A new criterion for selecting the parameter values is proposed that does not require any labelled validation data in D^T and relies on an entropy-based confidence measure for the predictions in D^T. To the author's best knowledge there is no published work proposing a comparable criterion.

5. The method is evaluated in extensive experiments, considerably outperforming recent methods from the literature for land cover classification. It is also compared to a recent method for UDA in the context of bi-temporal deforestation detection, where the proposed method results in a much higher performance on average.

In this thesis, the following scientific questions are examined:

1. Do the different variants of the proposed method for UDA achieve a stable positive transfer when adapting between different domains? By how much can the performance gap be reduced and where are the limitations?

2. Is the joint training scheme of the classifier and the appearance adaptation network sufficient to achieve semantically consistent adaptations? Do the proposed methods for improving semantic consistency actually result in a higher semantic consistency and if so, does this also lead to a higher classification performance after UDA?

3. Can the performance of UDA be improved further by combining the proposed method for UDA based on appearance adaptation with adaptive batch normalisation?

4. Does the proposed parameter selection method based on the entropy in the target domain lead to better results compared to running the UDA process for a fixed number of epochs?

5. Does a variant of the proposed method outperform approaches from the literature for UDA when evaluated on the applications of land cover classification and bi-temporal deforestation detection?

In order to answer these questions, extensive experiments are conducted that use multiple domains for two applications, namely land cover classification and bi-temporal deforestation detection.

1.3 Thesis Outline

The remainder of this thesis is structured as follows. Chapter 2 gives an overview of basic concepts that are relevant in the context of UDA for the pixel-wise classification of images using deep neural networks. This includes the relevant fundamentals of Deep Learning as well as an introduction to UDA. Furthermore, methodological approaches are presented on which the presented method is based. In Chapter 3, an overview of the relevant literature is given, with a focus on works that deal with UDA for the pixel-wise classification of images. In Chapter 4, the new approach for UDA is presented. The focus is set on the new appearance adaptation based method and the corresponding methodological extensions aiming at improving semantic consistency. Furthermore, the way in which the strategies of appearance adaptation and representation transfer are combined is described, and the proposed approach for parameter selection is presented. Chapter 5 gives an overview of the datasets used in the experiments as well as the experimental setup and evaluation protocols. The experimental results are presented and discussed in Chapter 6. Finally, Chapter 7 draws conclusions and provides an outlook on possible future work.

2 Basics

This chapter covers the basic concepts relevant to this thesis. After a formal introduction to the task of pixel-wise classification in Section 2.1, the relevant fundamental concepts of Deep Learning are discussed in Sections 2.2-2.4. After that, Section 2.5 addresses the task of appearance adaptation. Finally, the tasks of transfer leaning and unsupervised domain adaptation are formally defined and the used terminology is introduced in Section 2.6. Section 2.6 further introduces an approach for unsupervised domain adaptation from the literature, which will be used to extend the proposed method.

2.1 Machine Learning for Pixel-Wise Classification

Let X be an image with height h, width w and d channels and let $x_{r,c,q}$ be a pixel value of X in row $r \in \{1, ..., h\}$, column $c \in \{1, ..., w\}$ and channel $q \in \{1, ..., d\}$. Pixel-wise classification is the task of assigning a class-label $y_{r,c}$ to each pixel $x_{r,c}$ according to a predefined class structure $S = \{L_1, ..., L_{n_L}\}$ with n_L classes. The set of labels $\{y_{r,c}\}_{(r,c)\in I}$, with $I = \{1, ..., h\} \times \{1, ..., w\}$ being the image domain, corresponds to the label map Y. If Machine Learning (ML) is used for the classification, a classifier $\mathcal{C}$ with a set of parameters Θ_C is to be learned that maps an image $X \in \mathcal{X}$ to the predicted label map $\hat{Y} \in \mathcal{Y}$, thus $\hat{Y} = \mathcal{C}(\Theta_C, X)$. Here, $\mathcal{X}$ denotes the space of possible images and $\mathcal{Y}$ is the space of possible label maps.

To simplify the notation in this thesis, the so called *one-hot encoding* is introduced. Here, each label y is encoded as a vector γ with n_L entries $(\gamma_1, ..., \gamma_i, \ldots, \gamma_{n_L})$, where

$$\gamma_i = \begin{cases} 1 \text{ if } y = L_i \\ 0 \text{ otherwise.} \end{cases} \tag{2.1}$$

Accordingly, one-hot encoded label maps Γ are introduced, where the element $\gamma_{r,c}$ in row r and column c in Γ denotes the one-hot encoding for the label $y_{r,c}$ in the label map Y.

Although the parameters Θ_C could in principle be set manually, the core idea of ML is to determine the parameters automatically based on a set $T = \{(X_j, Y_j)\}_{j=1}^{n_T}$ of n_T training samples. In particular, the j-th training sample (X_j, Y_j) consists of an image X_j and the corresponding reference label map Y_j. The set T is referred to as training set or training data. In the literature, there are a variety of ways that can be used to implement $\mathcal{C}$. Depending on the used model, two approaches can be distinguished. In classical ML, models such as Support Vector Machines (Cortes and Vapnik, 1995), Decision Trees (Breiman et al., 1984) or Random Forests (Ho, 1995) are used to implement $\mathcal{C}$ based on hand-crafted features. The alternative approach to classical ML is called Deep Learning

(DL), where one tries to learn the feature extraction and the classifier simultaneously. While classical ML usually requires a careful selection of features that are used as input to the classifier, in DL often the raw features are used as input. The idea of DL is that the neural networks learn to extract relevant features jointly with learning the rules for classification, which has been shown to outperform approaches based on classical ML in various applications. This is usually done using a Deep Neural Network (DNN) (cf. Section2.2). For image-related tasks, specialised architectures such as Convolutional Neural Network (CNN) or Fully Convolutional Neural Networks (FCN) are commonly used, which are described in Sections 2.3 and 2.4, respectively.

In this work, two applications for pixel-wise classification in RS are addressed that differ with respect to the input data and the class structure. The two applications are:

1. **Land Cover Classification:** The first application is land cover classification (LCC) based on airborne imagery and, if available, elevation information. In this application, each pixel in an orthorectified multispectral aerial image (MSI) is to be classified according to a set of land cover classes, such as vegetation, buildings and roads. Often, information about the elevation is available, for example generated by image based 3D reconstruction or by methods based on LiDAR. Such information can be used to generate rasterised normalised digital surface models (nDSM). In the nDSM, each pixel contains the height of objects above the terrain. If the nDSM has the same geometrical resolution as the MSI, it can be treated as additional channel in the image to be classified. Consequently, X is a composite image that contains the MSI and the nDSM.

2. **Bi-temporal Deforestation Detection:** The second application is bi-temporal deforestation detection (BDD) where the input to the classifier consists of a pair of two georeferenced satellite images that show the same area but that were acquired at different dates. The desired output of the classifier is a binary map that indicates whether deforestation has happened at each pixel location in between the two acquisition dates. Such an automated classifier is useful for national agencies that track legal and illegal deforestation activities. In this application, each image X serving as input to the classifier is a composite image that contains the earlier image and the later image.

2.2 Deep Neural Networks

In recent years, Deep Neural Networks have become increasingly important in the field of ML. DNN can come in many different variants that are often tailored to specific applications. However, the basic building block of all DNN is always an artificial neuron (McCulloch and Pitts, 1943). Furthermore, in most cases, the main training scheme is the same, regardless of the DNN variant or application.

In this section, the main principles of DNN are explained by first introducing the so called multilayer perceptron, which is a very simple variant of a DNN in Section 2.2.1. Afterwards, Section 2.2.2 explains how a DNN is trained in a supervised way. Section 2.2.3 introduces some

strategies that aim to improve the generalisation capability of DNN, and in Section 2.2.4 the concept of adversarial training is introduced.

2.2.1 Neuron and Multilayer Perceptron

The basic building block of any DNN is an artificial neuron (McCulloch and Pitts, 1943) N that implements a parametrized mapping function $a = N(\Theta_N, \mathbf{v})$, which can e.g. be used as a classifier. Here, $\mathbf{v} = (v_1, ..., v_{n_v})$ is an input feature vector with n_v features, $\Theta_N = (\theta_1, ..., \theta_{n_v}, \theta_b)$ is a vector of the trainable parameters and a is the scalar output of the neuron. The output is computed by applying a non-linear activation function f_a to the linear combination of the input values $\mathbf{v}$. This can be formulated as

$$a = f_a(u), \tag{2.2}$$

where u is the result of the linear combination

$$u = \sum_{i=0}^{n_v} \theta_i \cdot v_i + \theta_b. \tag{2.3}$$

using the first n_v parameters in Θ_N as weights for the features and adding the bias parameter θ_b to the result.

The choice of f_a is arbitrary in principle and depends on the application. For example, the logistic sigmoid function

$$f_{sig}(u) = \frac{1}{1 + e^{-u}} \tag{2.4}$$

is commonly used to map values to the range of $(0, 1)$ in order to interpret them as probabilities. When using this activation function, a single neuron can be considered as a binary classifier that predicts the probability for one of the classes given a specific feature vector.

Multilayer Perceptron: In order to obtain a more complex classifier, multiple neurons are combined, often by arranging them in a layered structure. An classifier that consists of multiple neurons arranged in layers is the multilayer perceptron (MLP) (Rosenblatt, 1962). A MLP M consists of an input layer, n_{HL} hidden layers, and an output layer. Except for the input layer, each layer of a MLP has a set of parameters and the union of the parameters of all layers corresponds to the model parameters Θ_M. Let $\mathbf{u}^{(j)} = (u_1^{(j)}, ..., u_{n_{N,j}}^{(j)})$ be the vector of all outputs of the $n_{N,j}$ neurons in layer j before applying an activation functions and let $\mathbf{a}^{(j)} = (a_1^{(j)}, ..., a_{n_{N,j}}^{(j)})$ be the vector of all outputs of the $n_{N,j}$ neurons in that layer after applying the activation function, thus

$$a_i^{(j)} = f_a^{(j)}(u_i^{(j)}) \; \forall \; i \in [1, ..., n_{N,j}] \tag{2.5}$$

with

$$u_i^{(j)} = \sum_{n=0}^{n_{N,(j-1)}} \theta_{M,i,n}^{(j)} \cdot a_n^{(j-1)} + \theta_{M,i,b}^{(j)} \; \forall \; i \in [1, ..., n_{N,j}], \tag{2.6}$$

where $\theta_{M,i,n}^{(j)}$ is the weight associated with the connection of the i-th neuron in the layer j and the n-th neuron in the layer $j-1$, $n_{N,(j-1)}$ is the number of neurons in the layer $j-1$ and $f_a^{(j)}$ is the activation function used in layer j. The final output vector of the MLP corresponds to the vector

of activations $\mathbf{a}^{(O)}$ of the output layer, where $O = n_{HL} + 1$. An example for a MLP is illustrated in Figure 2.1.

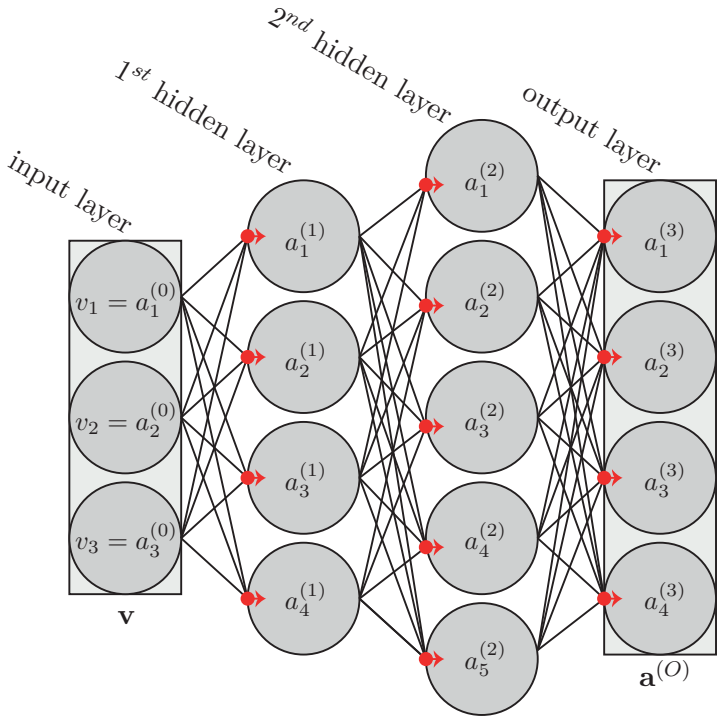

Figure 2.1: Schematic illustration of a multi-layer perceptron M with $n_{HL} = 2$ hidden layers designed to map a feature vector with three elements to a vector with four elements. The red dots indicate the computation of the output of each neuron according to Equation 2.5. It shall be noted that this architecture serves as an example for a very shallow DNN. Common architectures have a much higher number of layers and much more neurons per layer, and consequently they have a much higher number of parameters.

In a multi-class classification scenario with n_L classes there are n_L neurons in the output layer. The outputs $\mathbf{u}^{(O)}$ of the output layer before applying an activation function correspond to unnormalised class scores that are called *logits*. Commonly, the logits are normalised using the softmax function f_{sm} to be interpreted as a probability distribution. In particular, the softmax function normalises a vector such that that all values lie in the range of $(0, 1)$ and the sum of all values is one. The i-th entry $a_i^{(O)}$ in the normalised output vector $\mathbf{a}^{(O)} = f_{sm}(\mathbf{u}^{(O)})$ is computed according to

$$a_i^{(O)} = \frac{e^{u_i^{(O)}}}{\sum_{j=1}^{n_L} e^{u_j^{(O)}}} \; \forall \; i \in [1, ..., n_L], \tag{2.7}$$

where $u_i^{(O)}$ and $u_j^{(O)}$ are the unnormalised outputs of neuron i and j, respectively, in the output layer and n_L is the number of classes. The output $\mathbf{a}^{(O)}$ is interpreted as the predicted probability distribution $\hat{\gamma}$, in which the k-th entry $\hat{\gamma}_k = P(y = L_k|\mathbf{v})$ is the probability that the unknown class label y is the k-th class label L_k given the observed feature vector $\mathbf{v}$. During inference, a vector $\mathbf{v}$ is presented to the network and the class with the highest posterior probability (i.e., the highest softmax score) is considered as the final class prediction $\hat{y}$.

2.2.2 Supervised Training of Neural Networks

In a supervised scenario, a neural network N is trained, i.e. its parameters Θ_N are determined, by minimising a certain loss function for all samples in the training set $T = \{(\mathbf{v}_i, y_i)\}_{i=1}^{n_T}$, where n_T is the number of training samples, $\mathbf{v}_i$ is the i-th data point in the training set and y_i is the corresponding reference label. The loss function basically serves as a measure of the divergence between the network prediction and the known reference labels. Thus, if the loss function for the training data is minimised, the network makes better predictions for the training data. In order to minimise the loss, several optimisation strategies can be used. Those relevant for this thesis are introduced in Section 2.2.2.1.

In the following, several loss functions that are relevant for this theses are introduced. Commonly, different loss functions are used for multi-class classification and for binary classification. Let γ_i be the one-hot encoding for the reference label y_i as introduced in Section 2.1 and $\hat{\gamma}_{i,k}$ be the predicted probability $P(y_i = L_k|\mathbf{v}_i)$ of the network for a sample $\mathbf{v}_i$ to belong to class L_k. In a multi-class classification scenario, the network output $\mathbf{a}_i^{(O)}$, normalised using the softmax function, can directly be interpreted interpreted as the predicted label probability distribution, thus, $\mathbf{a}_i^{(O)} = \hat{\gamma}_i = N(\mathbf{v}_i)$. However, in a binary classification with the class structure $S = \{L_+, L_-\}$, the network commonly predicts a scalar value, normalised using the sigmoid function, which is interpreted as the probability $P(y_i = L_+|\mathbf{v}_i)$ for the sample $\mathbf{v}_i$ to belong to L_+. As the probabilities by definition sum up to one, the probability $P(y_i = L_-|\mathbf{v}_i)$ for the sample to belong to the other class can be computed to $P(y_i = L_-|\mathbf{v}_i) = 1 - P(y_i = L_+|\mathbf{v}_i)$.

A commonly used metric to model the loss in both scenarios is the cross-entropy. This metric is described in the following paragraphs for the scenarios of binary and multi-class classification, respectively.

Binary Cross-Entropy: In the scenario of binary classification, the number of classes is $n_L = 2$ and the class structure is $S = \{L_+, L_-\}$. The one-hot encoding of a label y_i is $\gamma_i = (\gamma_{i,+}, \gamma_{i,-})$. The value of $\gamma_{i,+}$ for the i-th data point is $\gamma_{i,+} = 1$ if the reference label y_i is L_+, and $\gamma_{i,+} = 0$ otherwise. The value of $\gamma_{i,-}$ is $1 - \gamma_{i,+}$. Correspondingly, the network output is the scalar $\hat{\gamma}_{i,+}$, which is the predicted probability $P(y_i = L_+|\mathbf{v}_i, \Theta_N)$ for $\mathbf{v}_i$ to belong to class L_+. To that end, the network has one neuron in the output layer, which uses the sigmoid function as non-linearity. Using this notation, the binary cross-entropy loss is given by

$$\mathcal{L}_{bce}(\Theta_N, T) = -\frac{1}{n_v} \sum_{i=1}^{n_v} \gamma_{i,+} \cdot log(\hat{\gamma}_{i,+}) + (1 - \gamma_{i,+}) \cdot log(1 - \hat{\gamma}_{i,+}), \tag{2.8}$$

where n_v is the number of samples over which the loss is calculated. Note that this loss formulation was already used in the context of logistic regression (Cox, 1958).

Multi-Class Cross-Entropy: In a multi-class classification scenario, the class structure is $S = \{L_1, ..., L_{n_L}\}$, where the number of classes is $n_L > 2$. The reference label y_i for the i-th data point $\mathbf{v}_i$ is one-hot encoded as a binary vector γ_i with the k-th entry being $\gamma_{i,k} = 1$ if the reference label for $\mathbf{v}_i$ is L_k and $\gamma_{i,k} = 0$ otherwise. Correspondingly, for a data point $\mathbf{v}_i$, the network predicts

the probability distribution $\hat{\gamma}_i$ where the k-th entry $\hat{\gamma}_{i,k}$ corresponds to the predicted probability $\hat{\gamma}_{i,k} = P(y_i = L_k|\mathbf{v}_i, \Theta_N)$ that $\mathbf{v}_i$ belongs to the class L_k. To that end, the network has L_k output neurons, and the output is normalised using the softmax function. Based on this notation, the multi-class cross-entropy loss becomes

$$\mathcal{L}_{ce}(\Theta_N, T) = -\frac{1}{n_v} \sum_{i=1}^{n_v} \sum_{k=1}^{n_L} \gamma_{i,k} \cdot log(\hat{\gamma}_{i,k}), \tag{2.9}$$

with n_v as in Equation 2.8. In order to minimise this loss, the classifier has to maximise the predicted probability for the correct class for all samples over which the loss is calculated.

Weighted Cross-Entropy A disadvantage of the multi-class cross-entropy loss is that it does not consider the frequency of the different classes in the dataset used for training. Particularly in scenarios in which some classes are underrepresented, i.e. occur less frequently in the training data compared to other classes, a trained classifier will tend to have a worse classification performance for these classes. One approach to counteract this phenomenon is to assign a weight ω_k to each class L_k. By assigning a higher weight to less frequent classes, the respective samples contribute more to the loss which can result in an improved classification performance for samples of that class. The weights are considered by extending equation 2.9, resulting in the formulation of the weighted cross-entropy loss

$$\mathcal{L}_{wce}(\Theta_N, T) = -\frac{1}{n_v} \sum_{i=1}^{n_v} \sum_{k=1}^{n_L} \omega_k \cdot \gamma_{i,k} \cdot log(\hat{\gamma}_{i,k}), \tag{2.10}$$

with ω_k being the weight of class L_k and the other symbols as in Equation 2.9. Note that the relative weight of a class may also be increased if samples for that class are classified with a lower accuracy compared to the other classes. Many approaches for determining the weights have been proposed in the literature, most of which are based on the frequency of samples for each class in the training set, e.g. (Ronneberger et al., 2015).

2.2.2.1 Optimisation Strategies

In ML, optimisation is applied to find a set of parameters $\hat{\Theta}$ that minimise the loss function $\mathcal{L}$ for the training data T. A straight-forward approach would be to calculate the positions of local extrema, where the gradient $\nabla\mathcal{L}(\Theta, T)$ becomes zero. However, this is practically not possible for neural networks, because the loss function is too complex. In such a scenario, one could perform a second order Taylor series approximation, which is a common optimisation strategy for logistic regression. Yet again, such an approach is not applicable in the context of DL, because a second order approximation requires to calculate the Hessian matrix $\nabla\nabla\mathcal{L}(\Theta, T)$ having the dimension $n_p \cdot n_p$, where n_p is the number of parameters. As the number of parameters of a DNN is often in the range of several millions, computing and storing the Hessian matrix is practically not feasible.

For these reasons, DNNs are commonly trained using variants of gradient descent (Goodfellow and Vinyals, 2015). Here, the parameters Θ_N of a network N are randomly initialised, for example by drawing from a normal distribution, which results in the initial parameter vector $\Theta_N^{(0)}$. Note that

in this notation, the superscript denotes the training iteration. The parameters are then iteratively updated by changing the parameter state according to the local gradient vector, i.e. by moving in the direction of the steepest descent. In gradient descent, the step-width is controlled by the hyper-parameter λ, referred to as *learning rate*. The update rule can be expressed as

$$\Theta_N^{(t+1)} = \Theta_N^{(t)} - \lambda \cdot \nabla \mathcal{L}(\Theta_N^{(t)}, T), \tag{2.11}$$

where t is the index of the current training iteration and

$$\nabla \mathcal{L}(\Theta_N^{(t)}, T) = \begin{pmatrix} \frac{\delta \mathcal{L}}{\delta \theta_1}(\Theta_N^{(t)}, T) \\ \vdots \\ \frac{\delta \mathcal{L}}{\delta \theta_{n_p}}(\Theta_N^{(t)}, T) \end{pmatrix} \tag{2.12}$$

is the gradient vector in iteration t. In order to compute the gradient, the loss function has to be differentiable with respect to each parameter $\theta_i \in \Theta_N$. In neural networks, the gradient vector can efficiently be computed by *back-propagation* (Rumelhart et al., 1986).

In the basic variant of gradient descent, the loss is computed over all training samples in T. In practice, DNNs are usually trained using mini-batch stochastic gradient descent (MB-SGD) where instead of calculating the loss over all training samples, a subset of the training set, referred to as a mini-batch, is used in each iteration. This variant enables much faster training and drastically decreases the memory requirement in comparison to the basic variant of gradient descent. However, using MB-SGD introduces an additional hyper-parameter, namely the batch size n_B, which corresponds to the size of each mini-batch. The batch size can have major influence on the training procedure. In particular, when it is too large it might not be possible to train the classifier due to the large memory requirements. On the other hand, if it is too small, the mini-batch may no longer represent the training dataset well. As a consequence, the resulting gradients may no longer point into the direction of a good parameter state, which means that the procedure will converge only slowly or not at all. Gradients that are strongly affected by the random selection of the samples in a mini-batch are commonly referred to as *noisy gradients*. Similarly, the learning rate λ has a major influence on the training. On the one hand, too small a learning rate will result in a slow convergence of the training procedure. On the other hand, too large a learning rate can result in a divergence or a poor classification performance after training.

When using MB-SGD, each training iteration consists of the following steps. First, a mini-batch is constructed by randomly drawing n_B training samples from the training dataset. Next, during the forward pass, each sample in the mini-batch is presented to the network. The resulting predictions for each sample and the corresponding reference are used to calculate the loss. Consequently, the gradient vector is determined using back-propagation. Lastly, the parameters are updated based on the gradient according to Equation 2.11.

There exist many variants of MB-SGD that improve the optimisation process, e.g. by reducing the required number of training iterations. Furthermore, some variants aim at making the optimisation less sensitive to the choice of hyper-parameters, simplifying the process of hyper-parameter tuning. Some variants also address optimisation problems related to MB-SGD which can improve the final performance of the DNN. In the following paragraphs, the variants used in this thesis are discussed.

Gradient Descent with Momentum: When training a DNN using MB-SGD, the optimization process can be slowed down by noisy gradients. A frequently used countermeasure is to extend MB-SGD by considering the estimate of the first moment $\dot{\Theta}_N$ of the parameters. The vector $\dot{\Theta}_N$ contains n_p scalar variables that describe the estimate of the first moment of each parameter, where n_p is the number of parameters of the network N. $\dot{\Theta}_N$ is initialised by $\dot{\Theta}_N^{(0)} = (0, \ldots, 0)$ and the parameter vector Θ_N is initialised as in regular MB-SGD. In each update step t, the gradient does not affect Θ_N directly, but instead is used to update the estimate of the first moment $\dot{\Theta}_N$ according to

$$\dot{\Theta}_N^{(t+1)} = \beta_0 \cdot \dot{\Theta}_N^{(t)} + \nabla\mathcal{L}(\Theta_N^{(t)}, T), \tag{2.13}$$

where β_0 is a hyper-parameter called the friction parameter. The actual parameters are updated using the estimated moment:

$$\Theta_N^{(t+1)} = \Theta_N^{(t)} - \lambda \cdot \dot{\Theta}_N^{(t+1)}, \tag{2.14}$$

where λ denotes the learning rate. In this work, this extended optimization strategy is referred to as MB-SGD-M.

Adaptive Momentum: Another commonly used optimisation strategy is referred to as *adaptive momentum* (ADAM) (Kingma and Ba, 2014). It extends MB-SGD-M by considering the second moment of each parameter to increase the rate of change for slowly changing parameters. ADAM introduces two scalar hyper-parameters β_1 and β_2, the vector of first moment estimates $\bar{\Theta}_N$ and the vector of second moment estimates $\hat{\Theta}_N$. Both vectors are initialised by zeros and the parameter vector Θ_N is again initialised as in regular MB-SGD. In each iteration t, first, the estimate of the first moment $\bar{\Theta}_N^{(t)}$ of the parameter vector is updated

$$\bar{\Theta}_N^{(t+1)} = \beta_1 \cdot \bar{\Theta}_N^{(t)} + (1 - \beta_1) \cdot \nabla\mathcal{L}(\Theta_N^{(t)}, T). \tag{2.15}$$

Next, the estimate of the second moment $\hat{\Theta}_N^{(t)}$ of the parameter vector is updated

$$\hat{\Theta}_N^{(t+1)} = \beta_2 \cdot \hat{\Theta}_N^{(t)} + (1 - \beta_2) \cdot \nabla\mathcal{L}(\Theta_N^{(t)}, T)^2, \tag{2.16}$$

with

$$\mathcal{L}(\Theta_N^{(t)}, T)^2 = \begin{pmatrix} \left(\frac{\delta\mathcal{L}}{\delta\theta_1}(\Theta_N^{(t)}, T)\right)^2 \\ \vdots \\ \left(\frac{\delta\mathcal{L}}{\delta\theta_{n_p}}(\Theta_N^{(t)}, T)\right)^2 \end{pmatrix} \tag{2.17}$$

Finally, each parameter θ_i is updated according to

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \lambda \cdot \frac{\bar{\theta}_i^{(t+1)}/(1 - \beta_1^t)}{\sqrt{\hat{\theta}_i^{(t+1)}/(1 - \beta_2^t)} + \epsilon}, \tag{2.18}$$

where $\bar{\theta}_i^{(t+1)}$ is the i-th value in $\bar{\Theta}_N^{(t+1)}$, $\hat{\theta}_i^{(t+1)}$ is the i-th value in $\hat{\Theta}_N^{(t+1)}$, λ denotes the learning rate and ϵ is a small constant value that is added for numerical stability.

2.2.3 Improving Model Generalization

Due to their high complexity, DNNs are prone to overfit to the training data, particularly if the training dataset is too small. The term overfitting is used to describe the situation in which the classifier performs well on the training data set, but badly when applied to unseen data, because it memorises the training data instead of learning to perform the classification based on patterns also appearing in the new data. Several strategies have been proposed to counteract this behaviour by regularising the parameters in the learning process. The strategies relevant for this thesis will be explained in the following paragraphs.

Weight Decay: One strategy to avoid overfitting of a network N with parameters Θ_N is to restrict the magnitude of the parameters (Krogh and Hertz, 1991). This is realized by adding either the $L1$ norm or the $L2$ norm of Θ_N to the loss. In the case of the $L2$ norm, the regularisation loss is

$$\mathcal{L}_{L2}(\Theta_N) = \sum_{i=1}^{n_p} (\theta_i)^2, \tag{2.19}$$

where n_p is the number of parameters of the network. In a classification scenario with the classification loss $\mathcal{L}_{cla}$ the final loss $\mathcal{L}$ becomes

$$\mathcal{L}(\Theta_N, T) = \mathcal{L}_{cla}(\Theta_N, T) + \tau \cdot \mathcal{L}_{L2}(\Theta_N). \tag{2.20}$$

where τ is a weighting parameter that controls the strength of the regularisation.

Data Augmentation: Another possibility to improve the generalization ability of a DNN is to synthetically expand the training dataset without requiring any additional training samples. The main concept of this strategy, referred to as *data augmentation*, is to randomly modify the training samples during training (Shorten and Khoshgoftaar, 2019). In principle, any type of modification can be applied as long as the modified training samples are still representative for the task to be learned. For example, if the task is to classify images, a frequently used variant of data augmentation is to randomly change the brightness and contrast of the images during training to obtain a classifier that performs better on images that have different radiometric properties than the images used for training.

Stopping Criterion and Parameter Selection: The number of training iterations is another important factor that can strongly affect the performance of a classifier on unseen data. While training for too few iterations can lead to underfitting, training for too many iterations may lead to overfitting. Furthermore, the performance of a classifier on both, the training dataset and unseen data, often shows large fluctuations during training. One approach for choosing a proper number of iterations is to treat the number of training iterations as a hyper-parameter which, for example, is set on the basis of previous experiments. However, the optimal number of training iterations can vary a lot from dataset to dataset, which is why using fixed number of iterations may not result in a good classification performance on unseen data.

For this reason, a common strategy is to determine the number of training iterations during training based on a *validation set*, i.e. a subset of the available labelled data which is not used for determining the parameters of the classifier. Commonly, the performance of the classifier on the validation set is tracked during training. After training the model for a fixed number of iterations, the parameter values leading to the best performance on the validation set are used as the final ones. An alternative approach is to stop the training if the performance on the validation set does not improve for a fixed number of iterations. This approach is referred to as *early stopping* (Prechelt, 1998).

Dropout: Srivastava et al. (2014) proposed another strategy to improve the generalization capabilities of DNN referred to as *dropout*. Dropout is usually applied to the neurons in one or multiple layers in a DNN. During training, the activation of the neurons in the corresponding layers are randomly set to zero with a certain probability (dropout probability). This strategy is related to the concept of ensemble learning, where several different classifiers are trained instead of a single one. The authors proposed to no longer set any neurons to zero during inference, but to rescale the respective activations by multiplying them by the dropout probability. The rescaling is necessary to obtain input values for subsequent neurons that have a range of values similar to the one during training.

2.2.4 Adversarial Training

So far, DNN have only been introduced for classification problems. However, they are frequently used to solve other tasks as well. One such task is the generation of synthetic data points that are similar to samples in an unlabelled training set $U = \{(\mathbf{r}_i)\}_{i=1}^{n_U}$ with n_U data samples. A solution for this task was proposed by Goodfellow et al. (2014). The authors address the task of image generation, aiming to train a so called *generator* network $\mathcal{G}$ to predict a synthetic image X^G based on a random variable z, such that $P(U^G) \approx P(U)$, where $U^G = \{X_j^G\}_{j=1}^{n_G}$ is a set of n_G generated images and X_j^G is the j-th generated image in U^G. To train the generator, they introduce the concept of *adversarial training*. It has to be noted that this training scheme is agnostic to the data type; for example, in (Engel et al., 2019), adversarial training is used to create synthetic audio samples.

In adversarial training, two DNNs are trained simultaneously. The first network is the generator $\mathcal{G}$ having a parameter vector Θ_G. It takes a vector $\mathbf{z}$ that is drawn randomly from some distribution as input and predicts an artificial data sample $\mathbf{g} = \mathcal{G}(\Theta_G, \mathbf{z})$. The second network $\mathcal{D}$ is called the *discriminator* and has the parameter vector Θ_D. This network serves as a binary classifier, where the classes to be differentiated are $S_D = \{L_R, L_G\}$. L_R corresponds to the case in which a sample serving as input to $\mathcal{D}$ is a sample from U and L_G means that a sample was generated by $\mathcal{G}$. D maps a data sample $\mathbf{v}_i$, which is either a real sample $\mathbf{r}$ or a generated sample $\mathbf{g}$, to a probability score $\hat{\gamma}_{L_R}$ that corresponds to the probability $P(y_i = L_R|\mathbf{v}_i) = 1 - P(y_i = L_G|\mathbf{v}_i)$ of the respective input sample to correspond to the class L_R, i.e. to originate from U. While $\mathcal{D}$ is trained to correctly predict the label for both, artificial and real samples, $\mathcal{G}$ is trained to deceive the discriminator by maximising the probability $P(y_i = L_R|\mathbf{g}_i)$ for a synthetic sample $\mathbf{g}_i$. Thus, $\mathcal{G}$ aims to generate

samples that are classified by $\mathcal{D}$ as being real samples. Consequently, the networks are optimised with respect to different loss functions. The loss $\mathcal{L}_D$ for the discriminator is formulated as

$$\mathcal{L}_D(\Theta_D, \Theta_G, U) = \frac{-1}{2 \cdot n_B} \left(\sum_{i=1}^{n_B} log\ P(y_i = L_R | \mathbf{r}_i, \Theta_D) + \sum_{i=1}^{n_B} log\ P(y_i = L_G | \mathbf{g}_i, \Theta_D, \Theta_G) \right), \quad (2.21)$$

where n_B is the batch size and $\mathbf{g}_i$ is the i-th synthetic sample in a batch of samples generated by $\mathcal{G}$. A naïve loss for the generator would be $\mathcal{L}_G^{(naive)} = -\mathcal{L}_D$. As the first term in Equation 2.21 does not depend on Θ_G, it does not affect the gradient of the loss with respect to the parameters of G and can therefore be neglected. This leads to

$$\mathcal{L}_G^{(naive)}(\Theta_D, \Theta_G) = \frac{1}{n_B} \sum_{i=1}^{n_B} log\ P(y_i = L_G | \mathbf{g}_i, \Theta_D, \Theta_G). \quad (2.22)$$

In practice, the generator is not trained to minimise the probability that the generated samples belong to L_G, but to maximise the probability that the generated samples belong to L_R. According to Goodfellow et al. (2014), this has positive effects on the optimisation, because this loss provides much stronger gradients for the generator in the early training phase. In particular, they observed that in the early training phase the discriminator will learn very fast to predict a low probability $P(y_i = L_R | \mathbf{g}_i)$ for synthetic samples to originate from U, which leads to small gradients for the derivative of the loss function $\mathcal{L}_G^{(naive)}$. Consequently, the standard loss for the generator becomes

$$\mathcal{L}_G(\Theta_D, \Theta_G) = -\frac{1}{n_B} \sum_{i=1}^{n_B} log\ P(y_i = L_R | \mathbf{g}_i, \Theta_D, \Theta_G). \quad (2.23)$$

The derivative of this loss with respect to the output of the discriminator approaches minus infinity when $P(y_i = L_R | \mathbf{g}_i)$ approaches zero, which results in larger update steps of the parameters of $\mathcal{G}$ in the early training phase. It should be noted that various other loss formulations for adversarial training have been proposed in literature, e.g. in (Mao et al., 2017), but they follow the same conceptual approach.

As suggested in (Goodfellow et al., 2014), adversarial training is usually performed by alternating updates of $\mathcal{G}$ and $\mathcal{D}$. In the update step of $\mathcal{G}$, a mini-batch of random vectors is presented to $\mathcal{G}$, resulting in a mini-batch of generated samples $\{\mathbf{g}_i\}_{i=1}^{n_B}$. The samples are presented to D and the generator loss is calculated according to Equation 2.23. Next, the gradient $\nabla \mathcal{L}_G(\Theta_D, \Theta_G)$ is computed and the parameters Θ_G are updated using a variant of gradient descent. Note that in this step, the parameters Θ_D are not updated. In the update step of $\mathcal{D}$, a mini-batch of generated samples and a randomly drawn mini-batch $\{(\mathbf{r}_i)\}_{i=1}^{n_B}$ of n_B samples from U are presented to $\mathcal{D}$. Based on the predicted probabilities the discriminator loss is calculated according to Equation 2.21 and the parameters Θ_D are updated using a variant of gradient descent. This training scheme is repeated, ideally leading to a stable state, called *equilibrium*, in which further training does no longer lead to an improvement of the generator or the discriminator.

2.3 Convolutional Neural Networks

The term *Convolutional Neural Network* (CNN) (LeCun et al., 1999) generally describes a group of DNNs that make use of so-called *convolutional layers* that exploit and explicitly consider the

spatial arrangement of the pixels in an image. Nowadays, such networks are used for multiple tasks related to image data.

While in a MLP the outputs of the neurons of each layer are arranged in a one dimensional vector, the outputs of neurons in each layer of a CNN are arranged in three dimensional tensors, i.e. they have a structure comparable to the arrangement of grey values in a multi-channel image. Thus, the output of each convolutional layer is a three dimensional activation map A in which the first and second dimensions correspond to the spatial coordinates, and the third dimension is the channel dimension. Note that in this thesis, the term pixel is not only used in the context of images, but it is also used to refer to the vector of values in an activation map at a specific spatial position.

In a CNN, the input to the first layer is an image. Each of the following layers in the CNN takes the output activation map of the preceding layer as input and outputs another activation map that may have a different spatial resolution and a different number of channels. In a CNN for image categorisation, i.e. the task of assigning a single class label to each input image, the last layers usually correspond to a MLP in order to predict a single probability distribution per image. Figure 2.2 shows an exemplary CNN architecture for multi-class image categorisation. In Sections 2.3.1-2.3.3, the basic types of layers of a CNN are presented and in Section 2.3.6, specific CNN architectures which are relevant for this thesis are described in detail.

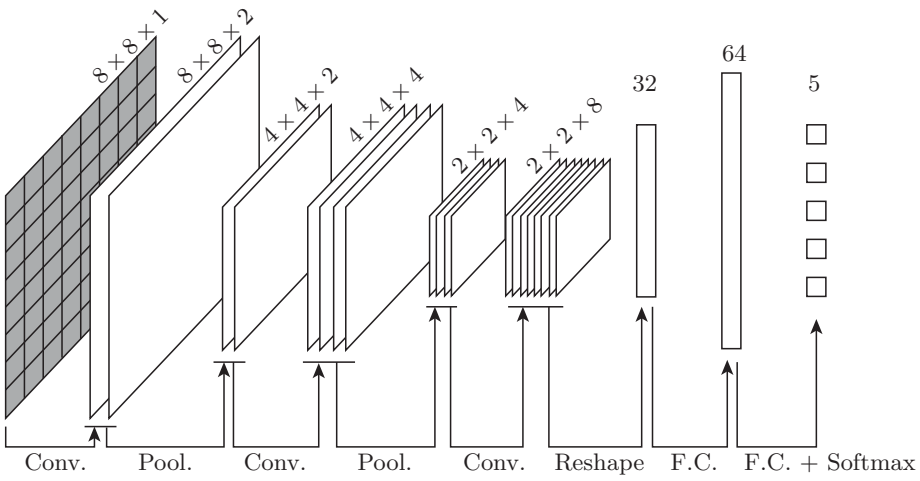

Figure 2.2: Exemplary CNN architecture for multi-class image categorisation. The network maps a single-channel image of size $8 \times 8\,px$ to a probability distribution according to a class structure with $n_L = 5$ classes. The image is first processed by a sequence of convolutional (Conv.) and pooling (Pool.) layers (cf. Sections 2.3.1 and 2.3.2). The activation map produced by the last convolutional layer is reshaped, i.e. the values are rearranged in a one-dimensional array, and processed by two fully connected (F.C.) layers that correspond to a simple MLP. The output of the second fully connected layer is normalised by the softmax function to obtain the class probabilities. At the top of each tensor in the figure, the respective shape is given. In particular, for three-dimensional tensors the shape is given in the form of $height \times width \times depth$ and for one-dimensional vectors only the number of entries is given.

2.3.1 Convolutional Layers

A convolutional layer is based upon the mathematical concept of a convolution. The output activation map $A^{(out)}$, that is the result of a convolutional layer is obtained by convolving the preceding input activation map $A^{(in)}$ with a set of $n_c^{(out)}$ kernel matrices $\{K_1, ..., K_{n_c^{(out)}}\}$, adding a bias parameter b_q per kernel K_q and applying an activation function $f_a(\cdot)$. The activation value $a_{r,c,q}^{(out)}$ of the q-th channel at row r and column c in $A^{(out)}$ is computed as

$$a_{r,c,q}^{(out)} = f_a\left(b_q + \sum_{r_k=1}^{h_k}\sum_{c_k=1}^{w_k}\sum_{q_k=1}^{n_c^{(in)}} k_{q,r_k,c_k,q_k} \cdot a_{r+r_k,c+c_k,q_k}^{(in)}\right), \tag{2.24}$$

where k_{q,r_k,c_k,q_k} refers to the parameter at row r_k, column c_k channel q_k in the q-th kernel. The symbols h_k, w_k, $n_c^{(in)}$ are height, width and depth of the input activation map, or, in the case of the first layer, of the input image, and $a_{r+r_k,c+c_k,q_k}^{(in)}$ is the activation value of the q_k-th channel at row $r+r_k$ and column $c+c_k$ in $A^{(in)}$. Note that if the indices are used according to Equation 2.24, the kernel has to be flipped horizontally and vertically before the operation in Equation 2.24 is applied, to formally correspond to a convolution. Practically, this is not relevant in the context of DL because the weights of each kernel are learned during training. Besides the weights of all kernels, the learnable parameters of a convolutional layer include the corresponding bias parameters. A convolutional layer with two kernels is illustrated in Figure 2.3.

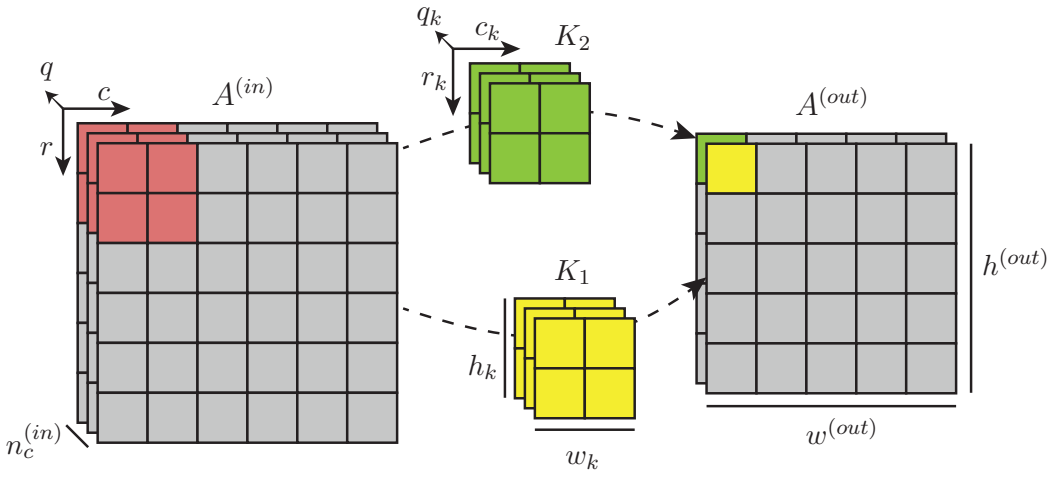

Figure 2.3: Schematic illustration of a convolutional layer with two kernels. The input activation map $A^{(in)}$ has $n_c^{(in)} = 3$ channels, the output activation map $A^{(out)}$ has $n_c^{(out)} = 2$ channels. Each kernel has a spatial size of $h_k \times w_k$ with $h_k = w_k = 2\,px$.

Unlike in a *fully connected layer*, where each neuron in a layer is connected to each neuron in the previous layer, in a convolutional layer the activation of a neuron only depends on a few activations in the input activation map. For example, the green and yellow neurons in $A^{(out)}$ in Figure 2.3 are only connected to the red neurons in $A^{(in)}$. Furthermore, the same parameter values are used at different spatial positions. These aspects of a convolutional layer lead to a considerable reduction of the number of parameters compared to a fully connected layer. This is beneficial for training because less training data are required as fewer unknown parameters need to be determined. Besides this basic variant of a convolutional layer, many variants and modifications exist. Those that are relevant for this thesis are explained in the following paragraphs.

Padding: As the example in Figure 2.3 shows, the output of a basic convolutional layer has a smaller spatial extent if the height h_k or the width w_k of the kernel is larger than one. This can be prevented by padding the input activation map accordingly, i.e. adding additional rows and columns at the borders of the activation map. A commonly used strategy is *zero-padding*, where the values in the padded areas are set to zero. Another option is *repetition-padding*, where the values at the border of the input activation map are used to fill the adjacent padded areas. In particular, the values of each pixel in the padded area is set to the values of the spatially closest pixel in the original activation map.

Strided Convolution: In a basic convolution the kernel is shifted over the whole input activation map with a step size of one pixel. However, the step size can be increased to reduce the spatial size of the output activation map. For example, if the step size is set to two pixels, every second row and column is skipped when shifting the kernel over the input activation map. This leads to a reduction of the height and width of the output activation map by a factor of approximately two compared to the input. Note that the step size in the horizontal direction may be different from the one in vertical direction. However, as in the architectures used in this thesis the two values are always the same, the term step size will be used to refer to the step size in both directions.

Depthwise Separable Convolution: In image processing, a commonly used strategy to speed up the convolution operation is to decompose a two-dimensional kernel K of shape $h_k \times w_k$ into two one-dimensional kernels K_h and K_w of shape $h_k \times 1$ and $1 \times w_k$, respectively (Wiejak et al., 1985). If K_h and K_w are selected such that $K = K_h * K_w$, the result obtained by convolving an image with K is equivalent to the result of first convolving the image with K_h before convolving the output with K_w. However, the latter variant requires only $h_k + w_k$ multiplications per pixel instead of $h_k \cdot w_k$. The main problem of this strategy is that not all kernels can be decomposed (Perona, 1995).

In the context of CNN, Chollet (2017) proposes the so called *depthwise separable convolutional layer* that is based on the concept of separable convolutions. Such a layer corresponds to a sequence of two convolutional layers that address the channel dimension and the two spatial dimensions, respectively. On the one hand, this speeds up the computations, because fewer multiplications are required to calculate the output compared to applying a single convolution. On the other hand, fewer parameters are to be determined during training, which can reduce the amount of training data required to train the network.

The first layer in a depthwise separable convolutional layer is called *pointwise convolution* and corresponds to a regular convolutional layer as presented in Section 2.3.1, but each kernel $K_{P,q}$ considers only a single pixel. Thus, each of the $n_c^{(out)}$ kernels in the pointwise convolution has a shape of $1 \times 1 \times n_c^{(in)}$, where $\mathrm{n}_c^{(in)}$ is the number of channels in the input $A^{(in)}$. In the pointwise convolution, no bias is added and no activation function is applied. The output of the pointwise convolution is $A^{(P)}$ with $n_c^{(out)}$ channels. The second layer performs what is called *depthwise convolution*. Here, each channel q is convolved independently with a kernel $K_{S,q}$ having shape of

$h_k \times w_k \times 1$. In the second layer a bias is added and an activation function is applied. Figure 2.4 illustrates a depthwise separable convolutional layer.

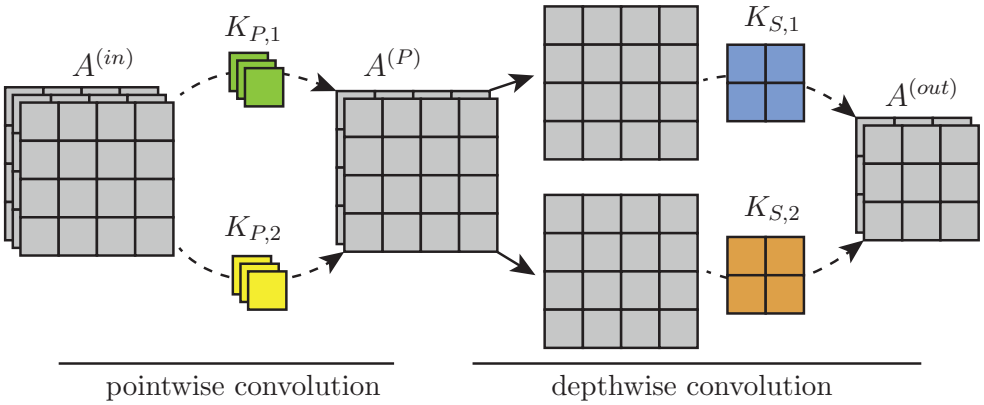

Figure 2.4: Schematic illustration of a depthwise separable convolutional layer. The pointwise convolution with two kernels produces the intermediate output $A^{(P)}$. Each channel in $A^{(P)}$ is then convolved individually using a kernel in the depthwise convolution step. The dotted arrows in the figure correspond to a convolution and the regular arrows are used to indicate the splitting of the channels of $A^{(P)}$.

Omitting the bias terms, such a depthwise separable convolution has $(n_c^{(in)} + h_k \cdot w_k) \cdot n_c^{(out)}$ parameters, while the number of parameters of a comparable regular convolution would be $n_c^{(in)} \cdot h_k \cdot w_k \cdot n_c^{(out)}$. In the example in Figure 2.4, the separable variant has 14 parameters, while the regular convolution with two 2×2 kernels would have 24 parameters.

Note that in many frameworks the order of the operations is flipped, i.e. the depthwise convolution is performed before the pointwise convolution. In this case, the activation function is applied only to the output of the pointwise convolution. However, Chollet (2017) argues that this difference is unimportant once multiple depthwise separable convolutional layers with a constant depth are stacked in an architecture.

2.3.2 Pooling Layer

A method to reduce the spatial size of the activation maps is to use a *pooling layer*. Here, a context window W is shifted over the input activation map $A^{(in)}$, similar to a kernel in a convolution. For each spatial position (r, c) in the output the values of the corresponding pixel are calculated by applying an aggregation function $f_{agg}(\cdot)$ to the set of pixels in the corresponding context window $W(r, c)$. Frequently, the aggregation function corresponds to the channel-wise maximum max or the channel-wise average avg. The values of each pixel $a_{r,c}^{(out)}$ in the output activation map of a pooling layer is computed according to

$$a_{r,c}^{(out)} = f_{agg}(W(r, c)). \tag{2.25}$$

In a pooling layer, the spatial reduction is achieved by using a step size (stride) larger than one. For example, if the context window corresponds to a area of $2 \times 2\,px$ and the step size is set to $2\,px$, a spatial downsampling by a factor of two is achieved.

2.3.3 Batch Normalisation Layer

A frequently used architectural modification of DNNs is to use *batch normalisation* (Ioffe and Szegedy, 2015). The core idea of batch normalisation is to normalise the output of a layer in a DNN such that after normalisation, the mean value and the standard deviation over all samples in a mini-batch are zero and one, respectively. On the one hand, this makes the gradients less noisy (cf. Section 2.2.2.1), which allows DNNs to be trained using larger learning rates, leading to faster convergence of the training procedure. On the other hand, using batch normalisation has also been shown to result in better performing classifiers (Santurkar et al., 2018).

In MLPs, batch normalisation is often considered in the form of batch normalisation layers, which perform the batch normalisation on a batch of n_B input activation vectors $\mathbf{a}_j^{(in)}$ for the same layer and output the normalised activations $\mathbf{a}_j^{(out)}$, where $j = (1, ..., n_B)$ is the sample index in the mini-batch. The basic variant of batch normalisation works as follows. The i-th value $a_{j,i}^{(out)}$ in $\mathbf{a}_j^{(out)}$ is computed according to

$$a_{j,i}^{(out)} = \zeta_i \cdot \left(\frac{a_{j,i}^{(in)} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \right) + \phi_i, \tag{2.26}$$

where $a_{j,i}^{(in)}$ is the i-th value in $\mathbf{a}_j^{(in)}$, ϵ is a constant scalar that is added for numerical stability, $\mu_i = \frac{1}{n_B} \sum_{j=1}^{n_B} a_{j,i}^{(in)}$ is the empirical mean, and $\sigma_i^2 = \frac{1}{n_B} \sum_{j=1}^{n_B} (a_{j,i}^{(in)} - \mu_i)^2$ is the the empirical variance of $a_{j,i}^{(in)}$ of the i-th value in $\mathbf{a}^{(in)}$. ζ_i and ϕ_i denote learnable parameters.

In CNNs, a variant of batch normalisation that follows the concept of weight sharing is commonly used. Instead of normalising each activation in an activation map independently, neurons that correspond to the same channel are normalised jointly. In this work, this variant is referred to as *2D batch normalisation*. In particular the activation $a_{j,r,c,d}^{(out)}$ in row r, column c and channel d of a 2D batch normalisation layer for the j-th sample in a mini-batch are calculated according to

$$a_{j,r,c,d}^{(out)} = \zeta_d \cdot \left(\frac{a_{j,r,c,d}^{(in)} - \mu_d}{\sqrt{\sigma_d^2 + \epsilon}} \right) + \phi_d, \tag{2.27}$$

where

$$\mu_d = \frac{1}{n_B} \sum_{j=1}^{n_B} \sum_{r=1}^{h^{(in)}} \sum_{c=1}^{w^{(in)}} a_{j,r,c,d}^{(in)} \tag{2.28}$$

and

$$\sigma_d^2 = \frac{1}{n_B} \sum_{j=1}^{n_B} \sum_{r=1}^{h^{(in)}} \sum_{c=1}^{w^{(in)}} (a_{j,r,c,d}^{(in)} - \mu_d)^2 \tag{2.29}$$

are the empirical mean and the empirical variance, respectively, for the activations in the d-th channel in the input activation map $A^{(in)}$. $a_{j,r,c,d}^{(in)}$ is the activation in row r, column c and channel d in the j-th input activation map $A_j^{(in)}$ with height $h^{(in)}$ and width $w^{(in)}$. ζ_d and ϕ_d again denote learnable parameters.

An aspect that has to be considered when using either variant of batch normalisation is how to deal with this operation during inference. In principle, the same operations could be used, but this

might be problematic if the batch size is reduced during inference to a few samples or to a single sample. In particular, the empirical mean and the empirical variance for each neuron (or channel) may no longer be meaningful. An alternative approach, proposed by Ioffe and Szegedy (2015), is to compute so called *running averages* of the empirical mean and the empirical variance for each neuron (or channel) that serve as estimates $\bar{\mu}$ and $\bar{\sigma}^2$ of the empirical mean and variance for each neuron (or channel), respectively. In particular, in training iteration t, the estimates $\bar{\mu}^{(t)}$ and $\bar{\sigma}^{2(t)}$ of each parameter are updated according to

$$\bar{\mu}^{(t)} \leftarrow (1 - \beta_{BN}) \cdot \bar{\mu}^{(t-1)} + \beta_{BN} \cdot \mu, \tag{2.30}$$

and

$$\bar{\sigma}^{2(t)} \leftarrow (1 - \beta_{BN}) \cdot \bar{\sigma}^{2(t-1)} + \beta_{BN} \cdot \sigma^2, \tag{2.31}$$

where β_{BN} is a hyper-parameter that controls how fast the running averages adapt to new values and μ, σ are mean and variance of the activation of the corresponding neuron or channel. The running averages are commonly initialised by $\bar{\mu}^{(0)} = 0$ and $\bar{\sigma}^{2(0)} = 1$.

During inference, the estimates of the training iteration replace the empirical mean and the empirical variance. In particular, the regular batch-normalisation step from Equation 2.26 becomes

$$a_{j,i}^{(out)} = \zeta_i \cdot \left(\frac{a_{j,i}^{(in)} - \bar{\mu}_i}{\sqrt{\bar{\sigma}_i^2 + \epsilon}} \right) + \phi_i, \tag{2.32}$$

where $\bar{\mu}_i$ is the estimated mean and $\bar{\sigma}_i^2$ is the estimated variance of the i-th neuron in $\mathbf{a}^{(in)}$. Accordingly, the 2D batch-normalisation step from Equation 2.27 becomes

$$a_{j,r,c,d}^{(out)} = \zeta_d \cdot \left(\frac{a_{j,r,c,d}^{(in)} - \bar{\mu}_d}{\sqrt{\bar{\sigma}_d^2 + \epsilon}} \right) + \phi_d, \tag{2.33}$$

where $\bar{\mu}_d$ is the estimated mean and $\bar{\sigma}_d^2$ is the estimated variance of the activations in the d-th channel in $A^{(in)}$.

2.3.4 Activation Functions in CNNs

Another aspect that is important in the context of DL is the choice of the activation functions in the hidden layers. Although in principle any non-linear and differentiable function can be used, only a few are frequently used in recent architectures for DL. In particular, saturating functions like the logistic sigmoid function, whose derivative approaches zero for very large and very small inputs, are problematic due to the so-called *vanishing gradient problem*. That is, if several layers with a saturating non-linearity function are stacked, layers close to the input will have a gradient with a very small magnitude, resulting in a slow convergence of the training procedure.

This problem was addressed by Nair and Hinton (2010), who propose to use an non-saturating activation function referred to as *rectified linear unit*. This activation function is defined as

$$f_{RL}(u) = \begin{cases} u, & \text{if } u > 0 \\ 0, & \text{otherwise} \end{cases}. \tag{2.34}$$

As the gradient remains constant for inputs $u > 0$, this activation function is non-saturating. It has been shown to result in faster convergence of the training compared to using activation functions like the logistic sigmoid function (Hara et al., 2015).

The rectified linear unit has one disadvantage, which is that the derivative is zero for inputs $u \leq 0$. This can lead to so-called *dead neurons*, i.e. neurons whose parameters are no longer updated during training and which do not contribute to the forward pass because their output is always zero. To solve this problem, Maas et al. (2013) introduced the so called *leaky rectified linear unit*, which is defined as

$$f_{LRL}(u) = \begin{cases} u, & \text{if } u > 0 \\ \theta_L \cdot u, & \text{otherwise} \end{cases}, \tag{2.35}$$

where θ_L is a hyper-parameter that defines the slope of the function for $u \leq 0$. By introducing this slope for negative values, the neurons always contribute to the forward pass, which prevents the problem of dying neurons.

2.3.5 Parameter Initialisation

Yet another important aspect regarding the training of DNNs is the initialisation of the learnable weights in the network. The standard approach is to draw the initial values for the parameters randomly from some distribution, e.g. a standard normal distribution. However, to optimise the convergence speed of a DNN, the choice of the distribution from which the initial values are drawn is important (He et al., 2015). He et al. (2015) take into account the effect of the rectified linear unit and suggest the following strategy. The bias parameters are initialised by zeros and the remaining weights are drawn from Gaussian distributions, which are centred at zero and have different standard deviations σ^j. In particular, the standard deviation σ^j of the Gaussian used to initialise the weights in the j-th layer is $\sigma^j = \sqrt{2/n^j_{p,in}}$, where $n_{p,in}$ is the number of connections of each neuron in layer j to the neurons in the preceding layer $j-1$.

2.3.6 CNN Architectures

Another important aspect of CNNs is the choice of the network architecture, i.e. the selection and arrangement of the layers and the choice of the corresponding hyper-parameters. The network architectures that are used in this thesis are based on recent CNN architectures for image categorisation from literature. Particularly important is the *Xception network* (cf. Section 2.3.6.2), which is based on the *ResNet* architecture (cf. Section 2.3.6.1).

2.3.6.1 Residual Networks

In the early years of DL, much improvement in the classification accuracy was related to increasing the learning capacity of CNNs, i.e. by using deeper and deeper networks with more and more parameters. However, simply increasing the number of layers leads to optimisation problems at some point. In particular, He et al. (2016) observed that increasing the the number of convolutional layers in a CNN eventually leads to an increasing training error. This is counter-intuitive,

considering the fact that additional layers could theoretically be expected to learn to perform an identity mapping. Thus, networks with more layers could, in principle, achieve the same training error as networks with fewer layers. The authors deduced that it is difficult for a convolutional layer to learn an identity mapping, resulting in optimisation problems in the training of very deep networks. As a consequence, He et al. (2016) proposed a CNN architecture for image categorisation, called *ResNet*, which is based on the concept of *residual learning*. In particular, they introduced so called *residual blocks* that usually consist of two or three convolutional layers that learn a mapping $f_R(\cdot)$ to predict a residual activation map $R = f_R(A^{(in)})$ based on an input $A^{(in)}$. The final output $A^{(out)}$ of the residual block is the sum of R and $A^{(in)}$, thus, $A^{(out)} = f_R(A^{(in)}) + A^{(in)}$. Using the residual blocks, an identity mapping can be performed by predicting a residual of zero, which is assumed to be relatively easy to learn. He et al. (2016) propose to compute the sum of input and residual before applying the last activation function. Figure 2.5 illustrates a residual block with two convolutional layers and rectified linear unit as activation function.

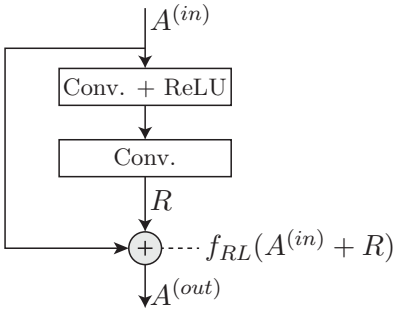

Figure 2.5: Illustration of a residual block in a ResNet CNN architecture. Conv.: Convolutional layer. ReLU, f_{RL}: Rectified linear unit (cf. Section 2.3.4).

The actual ResNet architecture uses several subsequent residual blocks at decreasing spatial scales. The activation maps at the lowest spatial resolution are processed by a MLP similar to the example given in Figure 2.2. Note that the full architecture of the ResNet is not presented here, because it is not relevant for this thesis.

2.3.6.2 Xception Network

The *Xception network*, proposed by Chollet (2017), is a CNN architecture for image categorisation that builds upon the concept of residual learning. The main modification is the integration of depthwise separable convolutions into the residual blocks. There are two types of such modified blocks, called *Xception blocks*. Chollet (2017) argues that, when using the same amount of parameters, this modification leads to an increased learning capacity compared to using, for example, residual blocks with regular convolutions. The Xception blocks are depicted in Figure 2.6.

Both types of Xception blocks take an activation map $A^{(in)}$ of shape $h^{(in)} \times w^{(in)} \times d^{(in)}$ as input. The Xception block of type A performs a spatial downsampling by a factor of two, and the block of type B preserves the spatial resolution. The strided convolution (layer 2 in block of type A) uses a 1×1 kernel and a step size of $2\,px$. To compute the downsampled residual, maximum pooling (layer 5 in block of type A) with a window size of $3 \times 3\,px$ and a step size of $2\,px$ is applied. All depthwise

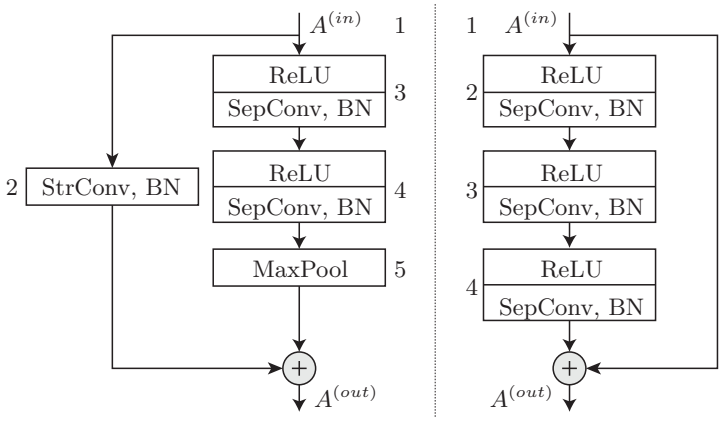

Figure 2.6: Illustration of the Xception blocks of type A (left) and B (right). The numbers next to the boxes are the layer indices used in Tables 2.1 and 2.2. StrConv: Strided convolution with step size of 2; SepConv: Depthwise separable convolution; BN: 2D batch normalisation. MaxPool: Max pooling with a stride of 2. ReLU: Rectified linear unit.

separable convolutions use 3×3 kernels in the depthwise convolution step (cf. Section 2.3.1). All layers use zero-padding with a width of 1 *px* (cf. Section 2.3.1). Note that in contrast to the original residual blocks described in Section 2.3.6.1, no activation function is applied to the output of the element-wise addition in each block. Tables 2.1 and 2.2 provide details about the dimensions of the output activation map of each layer.

Layer	Layer type	h	w	d
1	Input $A^{(in)}$	$h^{(in)}$	$w^{(in)}$	$d^{(in)}$
2	StrConv, BN	$h^{(in)}/2$	$w^{(in)}/2$	$d^{(out)}$
3	ReLU, SepConv, BN	$h^{(in)}$	$w^{(in)}$	$d^{(out)}$
4	ReLU, SepConv, BN	$h^{(in)}$	$w^{(in)}$	$d^{(out)}$
5	MaxPool	$h^{(in)}/2$	$w^{(in)}/2$	$d^{(out)}$

Table 2.1: Layers of an Xception block of type A. Layers 2 and 3 both take $A^{(in)}$ as input. h, w, d: Height, width and depth of the output of the layer. $d^{(in)}$, $d^{(out)}$: Depth of the input and output of the block. Remaining abbreviations and symbols as in the caption of Figure 2.6.

Layer	Layer type	h	w	d
1	Input $A^{(in)}$	$h^{(in)}$	$w^{(in)}$	$d^{(in)}$
2	ReLU, SepConv, BN	$h^{(in)}$	$w^{(in)}$	$d^{(in)}$
3	ReLU, SepConv, BN	$h^{(in)}$	$w^{(in)}$	$d^{(in)}$
4	ReLU, SepConv, BN	$h^{(in)}$	$w^{(in)}$	$d^{(in)}$

Table 2.2: Layers of an Xception block of type B. Abbreviations and symbols as in Table 2.1.

The full architecture of the Xception network, as suggested in (Chollet, 2017), is illustrated in Figure 2.7. The spatial extents of the activation maps are reduced step by step, resulting in the activation map of the last convolutional layer (layer 17). Global average pooling is applied to this activation map, i.e. the channel-wise average is computed, resulting in a one-dimensional representation. Lastly a fully connected layer is used and the softmax normalisation is applied to obtain probabilistic class scores. Note that the number of output neurons depends on the class structure.

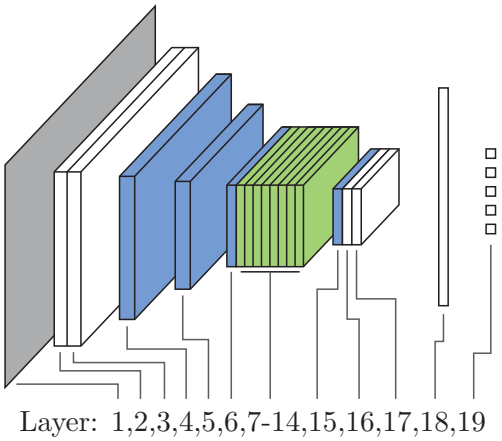

Figure 2.7: Illustration of the Xception architecture. The layer numbers correspond to the numbers in Table 2.3. Blue and green layers correspond to Xception blocks of type A and B, respectively. White layers correspond to regular convolutional layers.

The layers of the Xception network are listed in Table 2.3, where the Xception blocks described in Tables 2.1 and 2.2 are considered as submodules. The convolutional layers 2 and 3 use 3×3 kernels and do not use zero-padding. Layer 2 performs a spatial downsampling with a horizontal and vertical stride of $2\,px$. The depthwise separable convolutions in the layers 16 and 17 use 3×3 kernels in the depthwise convolution step and zero-padding with a width of $1\,px$. Note that there is a minor modification of the Xception block corresponding to layer 15. In this block, the number of kernels of the first depthwise separable convolutions is 728, which corresponds to $d^{(in)}$ for this channel and not to $d^{(out)} = 1024$.

Layer(s)	Layer type	h, w	d
1	Input layer	299	3
2	StrConv, BN, ReLU	149	32
3	Conv, BN, ReLU	147	64
4	Xception block A	74	128
5	Xception block A	37	256
6	Xception block A	19	728
7-14	8× Xception block B	19	728
15	Xception block A	10	1024
16	SepConv, BN, ReLU	10	1536
17	SepConv, BN, ReLU	10	2048
18	Glob. AvgPool, ReLU	1	2048
19	Fully connected, Softmax	1	n_L

Table 2.3: Layers of the Xception network. Glob. AvgPool: Average pooling with a window size of $10 \times 10\,px$. n_L denotes the number of classes. Remaining abbreviations and symbols as in Table 2.1.

It is noted that, although Chollet (2017) designed the architecture for a fixed image size of $299 \times 299\,px$, the architecture can easily be modified to work with a different size of input images. In particular, the global average pooling operation, where the channel-wise average is computed is agnostic to the size of the corresponding input activation map.

2.4 Fully Convolutional Networks

While CNNs were originally designed for the task of image categorization, there are variants designed for the pixel-wise classification of images or pixel-wise regression tasks. This group of CNNs is called *fully convolutional networks* (FCNs) (Long et al., 2015a). Following the formal definition of pixel-wise classification given in Section 2.1, FCNs for classification tasks take as input an image X with height h and width w and predict a corresponding label map $\hat{Y}$, usually with the same spatial extent. Thus, the output A of the last layer corresponds to an activation map of shape $h \times w \times n_L$ in the case of a multi-class classification with $n_L > 2$ classes, and $h \times w \times 1$ in the case of a binary classification. In the first case, A is normalised by applying the softmax function to the predicted vector at each pixel position in order to obtain maps $\hat{\Gamma}$ of pixel-wise probabilistic class scores. Taking the indices of the classes with the highest class scores per pixel results in the final class predictions $\hat{Y}$. In the second case, each value in A is normalised by the logistic sigmoid function. The normalised probabilistic class scores corresponds to the probabilities of the corresponding pixels to belong to either of the two classes. FCNs for regression problems work in a similar way, but usually do not apply an activation function or normalisation to the output of the last layer.

In order to make predictions at pixel-level, the predicted map of probabilistic class scores must have the same spatial extent as the input image. To that end, many FCN architectures follow an *encoder-decoder* strategy, in which the spatial extent of the activation maps is reduced in the first layers (encoder) and increased again step-wise in the later layers (decoder) of the network, e.g. (Ronneberger et al., 2015; Zhang et al., 2018c). Compared to the naïve approach of simply applying a series of convolutional layers at the original scale, such networks can have a much larger number of parameters without consuming too much memory.

In principle, FCNs are trained in the way described in Section 2.2.2. However, there is no longer only one loss term per sample, but instead one loss term per pixel. Consequently, the overall loss corresponds to the average value of the pixel-wise loss terms. The loss can still be computed based on multiple images in a mini-batch, but in order to process multiple images in a mini-batch in a computationally efficient way, they must have the same spatial size. To that end, the mini-batches usually consist of square patches with a fixed side length p. When training on such mini-batches, the cross-entropy loss for the pixel-wise classification becomes

$$\mathcal{L}_{ce,fcn}(\cdot) = -\frac{1}{n_B \cdot p^2} \sum_{b=1}^{n_B} \sum_{i=1}^{p^2} \sum_{k=1}^{n_L} \gamma_{b,i,k} \cdot log(\hat{\gamma}_{b,i,k}), \tag{2.36}$$

where $\hat{\gamma}_{b,i,k} = P(y_{b,i} = L_k | X_b)$ is the predicted probability for the i-th pixel in the b-th image in the batch to belong to class $y_{b,i} = L_k$ and n_B is the batch size. During training, the patches are commonly obtained by cropping a randomly selected area from the original images and from the respective reference label map. The patch extraction allows to apply a variety of geometric data augmentation methods, such as random horizontal or vertical flipping of the patches or applying random rotations. During inference, the images are usually tiled to obtain patches with the same spatial extent as those used during training.

Compared to CNNs for image categorisation, FCNs require layers that perform a spatial upsampling of the activation maps. Such layers are introduced in Section 2.4.1. A further aspect, particularly important for FCNs, is that the reduction of the spatial resolution may lead to imprecise object delineations in the predicted label maps. This aspect and a method to overcome this problem is discussed in Section 2.4.2.

2.4.1 Upsampling Layer and Transposed Convolutional Layer

A commonly used and straight-forward strategy to increase the resolution of an activation map is to upsample it, e.g. by linear, bi-linear or nearest neighbour interpolation. For example, this strategy is used in (Ronneberger et al., 2015; Zhang et al., 2018c). This approach introduces no trainable parameters, but it may lead to imprecise object outlines due to the interpolation. Another strategy, proposed in (Long et al., 2015a), is to use *transposed convolutional layers*. The goal of transposed convolutional layers is to learn the upsampling operation instead of interpolating the activation maps, which can help to recover the spatial information, because the upsampling itself is learned. The procedure can be interpreted as follows: An upsampled version $A^{(us)}$ of the input activation map $A^{(in)}$ is generated and used as input to a regular convolutional layer. $A^{(us)}$ is obtained by adding s_h rows after each row in $A^{(in)}$ and s_w columns after each column in $A^{(in)}$ that are filled with zeros. The parameters s_h and s_w control the change of resolution. For example, if $s_h = s_w = 2$, an upsampling by a factor of approximately three is achieved. Note that the exact upsampling factor also depends on the kernel size. An example is shown in Figure 2.8. Here, the input $A^{(in)}$ with a spatial extent of $3 \times 3\,px$ is upsampled by a factor of three. However, as the upsampled activation map $A^{(up)}$ is convolved with a 3×3 kernel, the output of the transposed convolution has a spatial extent of $7 \times 7\,px$. In order to obtain an integer upsampling factor, the upsampled activation map is commonly padded with zeros to compensate for the size of the kernel.

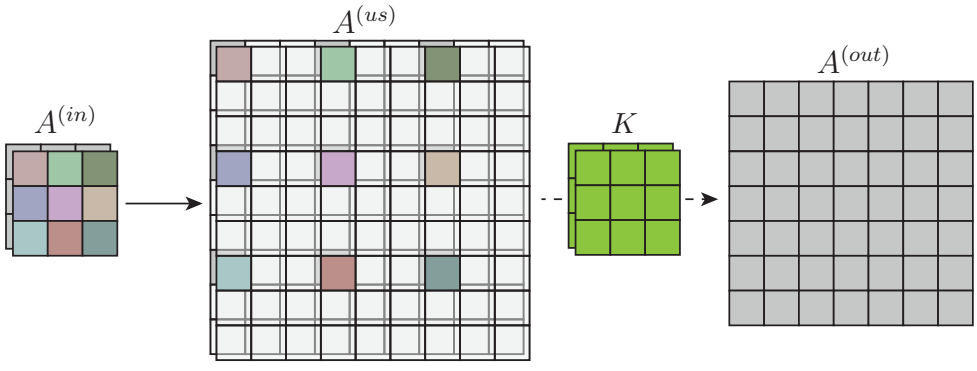

Figure 2.8: Schematic illustration of a transposed convolutional layer. The input activation map $A^{(in)}$ with two channels is upsampled by appending $s_h = 2$ rows filled with zeros to every row and appending $s_w = 2$ columns filled with zeros to every column. The output activation map $A^{(us)}$ then serves as input to a regular convolution, in the example using a single kernel K. Note that identical colours in $A^{(in)}$ and $A^{(up)}$ indicate that the corresponding value is copied from $A^{(in)}$ to $A^{(up)}$.

2.4.2 Skip Connections

Due to the upsampling operations, encoder-decoder networks tend to predict the border areas of objects with a low spatial accuracy. A common strategy to circumvent this problem is to use *skip connections* between corresponding layers of the encoder and the decoder, i.e. between layers having the same spatial resolution. Ronneberger et al. (2015) proposed to implement the skip connections by concatenating the respective activation map of the encoder and the activation map in the decoder. Figure 2.9 illustrates a simple FCN with skip connections. In the first convolutional layer, the spatial resolution is not changed, but it is reduced by a factor of two in the second and third layers. These three layers build the encoder of the network. The activation maps of the layers C1 and C2 are concatenated to the output of the transposed convolutional layers T1 and T2, respectively, which both perform a spatial upsampling by a factor of two. Finally, two convolutional layers are appended, the last one of which predicts the final output of the network. The layers T1, T2, C4 and C5 build the decoder of the network.

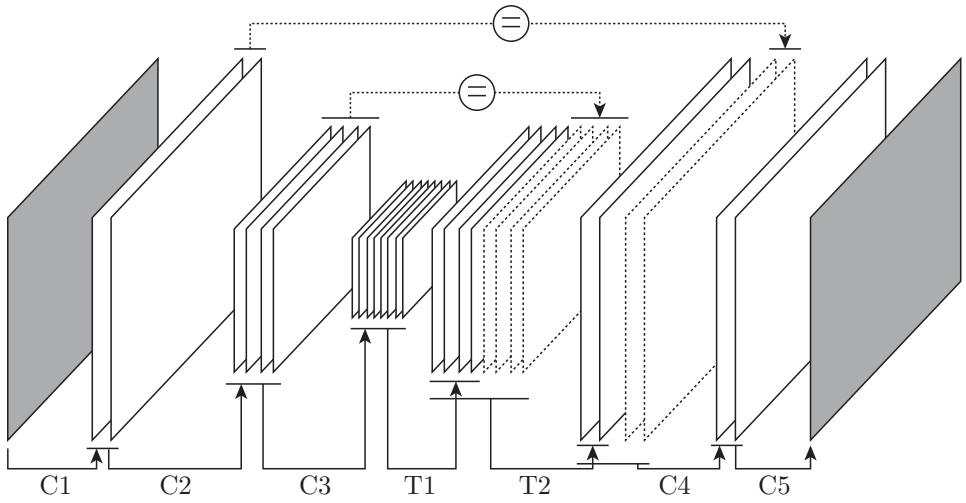

Figure 2.9: Illustration of an encoder-decoder FCN with skip connections. The solid arrows indicate the processing by convolutional layers (C1-C5) and transposed convolutional layers (T1, T2). The skip connections are represented by dotted arrows. The activation maps with a dotted outline are obtained by copying the respective activation maps from the encoder.

2.5 Appearance Adaptation

In the literature, CNNs and FCNs have been used for many different tasks, not only for image classification. One task that is particularly important for this thesis is the generation of an (output) image $\hat{X}$ based on an (input) image X. Early work dealt with applications such as image colorization (Varga and Szirányi, 2016) or image stylisation (Gatys et al., 2016; Ulyanov et al., 2016), the latter referring to the modification of an image according to the style of a reference image. Later work addressed the more generic task of *appearance adaptation*, also known as *image-to-image translation* in computer vision. Here, one has two unpaired image sets U^A and U^B that contain images from a domain D^A and a domain D^B, respectively. The goal of appearance adaptation is to train a FCN $\mathcal{A}$ that maps an input image X^A from U^A to an adapted version $X^{AB} = \mathcal{A}(X^A)$

that has an appearance (style) similar to the images in U^B, but still has the same semantic content as X^A according to a class structure that is usually given only implicitly by the selection of the datasets. This learning task is particularly difficult for the following reasons. First, the definition of *appearance* is arbitrary and depends on the application. Similarly, the meaning of the term *content* is usually arbitrary if the underlying class structure is not explicitly given. Second, the problem is ill-posed because even if these terms are clearly defined, there are usually multiple solutions for $\hat{X}$ that satisfy the appearance and content constraints. Finally, because of the two sets are unpaired, it is not possible to train $\mathcal{A}$ in a supervised way.

In order to achieve the first goal of appearance adaptation, namely the adaptation of the appearance of the images in the target set U^B, most approaches in literature make use of adversarial training (cf. Section 2.2.4). In particular, the network $\mathcal{A}$ is trained in an adversarial way together with a domain discriminator $\mathcal{D}$. While $\mathcal{D}$ is trained to correctly discriminate between images from U^B and images generated by $\mathcal{A}$, the adaptation network $\mathcal{A}$ is trained to predict images that are classified by $\mathcal{D}$ as originating from U^B with a high probability. This scheme is used, for example in (Lee et al., 2018; Zhu et al., 2017a; Tasar et al., 2020b; Huang et al., 2018b). A commonly used architecture for the domain discriminator is the so-called PatchGAN, proposed in (Isola et al., 2017). Here, the discriminator corresponds to a FCN that does not output a single scalar probability for an input image, but instead outputs a map Ψ of probabilistic class scores that are averaged when computing the adversarial loss terms. Each value $\psi_i \in \Psi$ corresponds to a support window in the input image and, thus, describes the probability of that window to originate from U^B. The actual size of the support window depends on the receptive field of $\mathcal{D}$, i.e. the size of the area in the input image that contributes to a single pixel in the output of $\mathcal{D}$. It has to be highlighted, that the adversarial training aims to find an equilibrium state $P(U^{AB}) \approx P(U^B)$, in which the distribution of a set $U^{AB} = \{X_j^{AB}\}_{j=1}^{n_A}$ of n_A adapted images X_j^{AB} is indistinguishable from the distribution of the images in U^B (cf. Section 2.2.4). However, as the discriminator only ever predicts the origin for smaller patches according to the size of its receptive field, only the distribution in the local neighbourhoods, i.e. the local feature distributions, are matched. Considering this aspect, Isola et al. (2017) show that the size of the support window has great impact on the appearance adaptation and suggests an architecture the has a receptive field of $70 \times 70\,px$. The concept of the PatchGAN discriminator is illustrated in Figure 2.10. The black areas illustrate the propagation of the support window to a single pixel in the output of $\mathcal{D}$.

The second goal of appearance adaptation is to preserve the semantic content of an image during the adaptation. Note that in literature, different formulations have been used for this requirement. For example, Zhu et al. (2017a) require that after appearance adaptation, X^A and X^{AB} should be paired up in a meaningful way and Soto et al. (2021) require that the adaptation must preserve the semantic structure. Tasar et al. (2020b) introduce the term of *semantic consistency* to describe pairs of images and adapted images that are semantically equivalent. In this thesis, the terminology introduced in (Tasar et al., 2020b) is used. In particular, the term *semantic consistency* is used to describe image adaptations in which the semantic contents of X^A and X^{AB} are the same according to an underlying class structure. Formally, this means that the label maps predicted for X^A by an ideal classifier for D^A should be identical to the label maps predicted for X^{AB} by an ideal classifier for D^B.

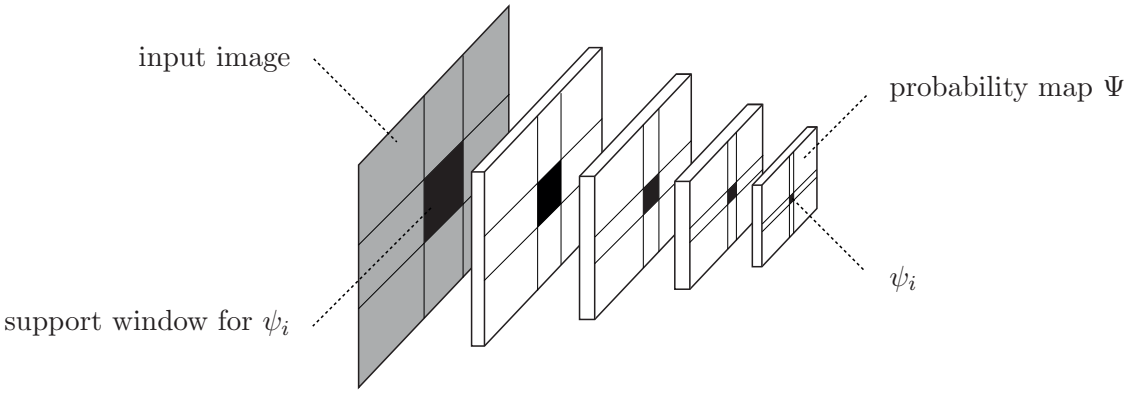

Figure 2.10: Illustration of the PatchGAN discriminator (Isola et al., 2017) for adversarial appearance adaptation. The core idea is to predict a map of probabilistic class scores Ψ by feeding the input image through a sequence of convolutions. Each value $\psi_i \in \Psi$ corresponds to the probability of the corresponding support window to come from a specific domain.

To achieve semantic consistency, many approaches for image adaptation rely on bidirectional image adaptation, i.e. the architecture consists of two adaptation networks $\mathcal{A}^{AB}$ and $\mathcal{A}^{BA}$ that adapt images from U^A such that they look like images from U^B and vice versa, respectively. This allows to apply a regularisation loss based on the concept of *cycle consistency*, requiring images from D^A adapted to D^B and back to D^A to be similar to the original ones, thus $\mathcal{A}^{BA}(\mathcal{A}^{AB}(X^A)) \approx X^A$. The respective regularisation loss $\mathcal{L}_{cyc}$ can be expressed for example as the L1 loss:

$$\mathcal{L}_{cyc}(\cdot) = \frac{1}{n_B \cdot p^2} \sum_{b=1}^{n_B} \sum_{i=1}^{p^2} |\hat{x}_{b,i}^A - x_{b,i}^A|, \tag{2.37}$$

where p is the patch size, n_B is the batch size, $x_{b,i}^A$ is the i-th grey value in the b-th patch in a batch of image patches from U^A and $\hat{x}_{b,i}^A$ is the i-th grey value in $\hat{X}_b^A = \mathcal{A}^{BA}(\mathcal{A}^{AB}(X_b^A))$. Usually this constraint is applied in both directions (Zhu et al., 2017a).

An alternative strategy to perform appearance adaptation in a semantically consistent way was proposed in (Lee et al., 2018). This approach is frequently used, e.g. in (Tasar et al., 2020b; Huang et al., 2018b). The method by Lee et al. (2018) is based on the assumption that each image can be decomposed into two components, *content* and *appearance*. In particular, it is assumed that the content attribute is agnostic to the domain, while the appearance is related to the difference between the domains. An exemplary application is to perform appearance adaptation between two domains, the first domain consisting of images of cats and the second one consisting of images showing dogs. In this example, the content is related to the domain agnostic scene layout, i.e. to the pose of the animal, while the appearance describes what the animal looks like. Following this line of thought, appearance adaptation can be interpreted as the task of exchanging only the appearance component of an image by the appearance of an image from another domain. To that end, Lee et al. (2018) aim to learn to disentangle the information about the content of an image from the information about the domain specific appearance and at the same time to train a network to predict an image with a specific content and appearance.

Practically, for each domain, two CNNs are trained to extract the information about content and appearance, respectively. $\mathcal{E}_c^A$ and $\mathcal{E}_a^A$ are the respective encoders for content and appearance in the

source domain and $\mathcal{E}_c^B$ and $\mathcal{E}_a^B$ are the corresponding encoders in the target domain. Lee et al. (2018) propose to use a vector representation for the appearance and a matrix representation for the content. The extracted representations are then jointly presented to domain specific generators $\mathcal{G}_j^A$ and $\mathcal{G}_j^B$ for the two domains, respectively. In particular, for the first domain D^A, this means that the content representation extracted from an image X^B from D^B is combined with the appearance representation extracted for an image X^A from D^A and used as input to a CNN $\mathcal{G}_j^A$ that predicts an image X^{BA} that is supposed to show the content of X^B but with the appearance from X^A. Accordingly, $\mathcal{G}_j^B$ predicts X^{AB} having the content of X^A but the appearance of X^B. This process is repeated based on the images X^{AB} and X^{BA}, resulting in images $\hat{X}^A$ and $\hat{X}^B$. The main strategy is based on enforcing so called *cross cycle consistency*, i.e. on applying a loss term that enforces that $\hat{X}^A \approx X^A$ and $\hat{X}^B \approx X^B$. Furthermore, to achieve the goal of disentangling content from appearance, adversarial training is employed to make the distribution of content encodings $\mathcal{E}_c^B(X^B)$ in D^B indistinguishable from content encodings $\mathcal{E}_c^A(X^A)$ in D^T. Figure 2.11 gives an overview of the concept of cross cycle consistency. For further details, cf. (Lee et al., 2018).

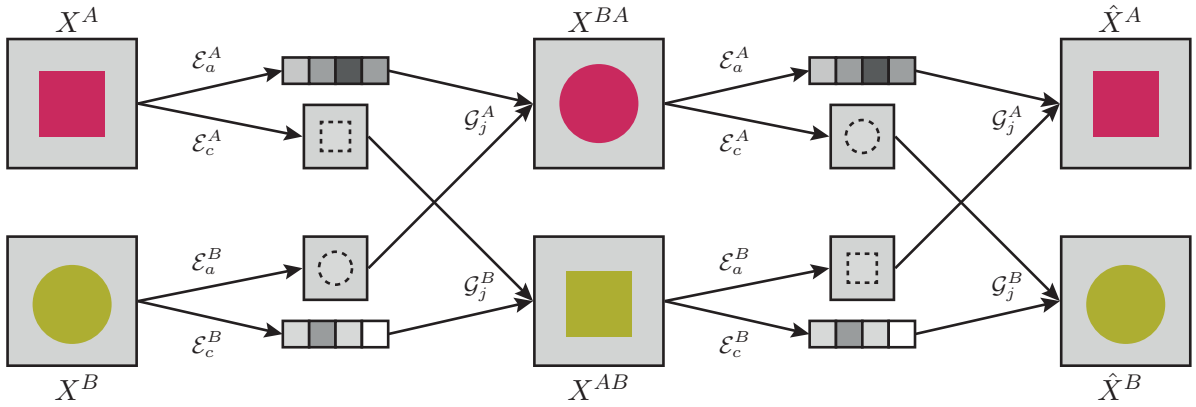

Figure 2.11: Illustration of image adaptation based on cross cycle consistency. The goal is to learn to disentangle appearance and content for images from two domains and to predict images with specific content and appearance. The actual image adaptation is performed by predicting an image that has the content of an image from one domain and the appearance of an image from the other domain, e.g. the image X^{BA} has the content of X^B but the style from X^A. If this process is performed performed twice for an image pair X^A, X^B, the outputs $\hat{X}^A, \hat{X}^B$ should be similar to the original images. Note that in the example, the appearance refers to the colour and the content to the shape of the depicted object.

2.6 Transfer Learning and Domain Adaptation

A basic assumption in ML is that the data which are available to train a classifier in a purely supervised manner are representative for the actual data samples to which the classifier is to be applied in the inference. However, this assumption is often not fulfilled in a real scenario, which can result in a poor classification performance. A generic approach to address this problem is so called *Transfer Learning* (TL).

The term Transfer Learning covers methods that aim to transfer knowledge from a source domain D^S, where there are many training examples, to a target domain D^T, where only a limited amount or no training data is available. The main goal of TL is to use the information available in D^S to find a better solution for the task in D^T, which requires the domains to be related (Pan and Yang, 2009). Note that there also exist variants that consider either multiple source domains, multiple target domains, or both. However, such methods are not addressed in this thesis.

Several survey papers on TL (Pan and Yang, 2009; Wang and Deng, 2018; Csurka, 2017) use the following notation and definitions. A domain $\mathcal{D}$ comprises a feature space $\mathcal{X}$ and the marginal distribution $P(X)$ of the feature space, where $X = \{x_1, ..., x_n\} \in \mathcal{X}$. Consequently, in the common case with two domains, the source domain is defined as $D^S = \{\mathcal{X}^S, P(X)^S\}$ and the target domain as $D^T = \{\mathcal{X}^T, P(X)^T\}$. Each domain is associated with a learning task $\mathcal{T}$, which consists of a label space $\mathcal{Y}$ and a predictive function $f(\cdot)$ which can be regarded as the conditional probability distribution $P(Y|X)$. Thus, the learning task is given by $\mathcal{T}^S = \{\mathcal{Y}^S, P(Y|X)^S\}$ in D^S and by $\mathcal{T}^T = \{\mathcal{Y}^T, P(Y|X)^T\}$ in D^T.

Domain Adaptation: In the literature, there are different formal definitions of *Domain Adaptation* (DA). Following the definitions of (Pan and Yang, 2009; Wang and Deng, 2018; Csurka, 2017), DA refers to a setting of TL in which the learning tasks in both domains are identical. In this case, the remaining difference is caused only by differences in the domains, $D^S \neq D^S$. This can either result from a distribution shift $P(X)^S \neq P(X)^T$, from a change of the feature space $\mathcal{X}^S \neq \mathcal{X}^T$, or both. A setting in which the domains share a common feature space is called *homogenous DA*, otherwise one speaks about *heterogeneous DA*.

In this thesis, however, another definition of Domain Adaptation is used, following (Tuia et al., 2016). In (Tuia et al., 2016), the domains are associated with the joint probabilities $P(X, Y)^S$ and $P(X, Y)^T$ which are assumed to be different but related. The difference between the joint distributions in the two domains can be interpreted as domain gap. As $P(X, Y) = P(X|Y) \cdot P(Y)$ the domain gap can result either from a difference in the distribution of the labels, $P(Y)^S \neq P(Y)^T$, or from a difference in the conditional probabilities $P(X|Y)^S \neq P(X|Y)^T$ of the features given the labels, or both. In RS applications, the marginal distribution $P(Y)$ of the labels depends on the scenes that are depicted in the images related to a domain, and the conditional probability $P(X|Y)$ can be interpreted as the appearance of objects belonging to specific classes. In the addressed applications of land cover classification and bi-temporal deforestation detection, both distributions can in fact be different between two domains and, thus, can be relevant to describe the domain gap.

A further categorisation of DA is made based on the availability of labelled samples in the two domains. In the context of this thesis, two settings are particularly important: supervised DA and unsupervised DA.

Supervised Domain Adaptation: A setting in which labelled images are available in both domains is commonly referred to as *supervised DA* (Tuia et al., 2016; Wang and Deng, 2018). In

particular, it is assumed that in the target domain there are not enough reference labels to train a classifier that achieves a satisfactory performance in that domain. At the same time, plenty of training samples are considered to be available in D^S. In the context of DL, the default strategy in such a scenario is referred to as *supervised pre-training and fine-tuning or re-training* (Krizhevsky et al., 2012; Yosinski et al., 2014). Here, the classifier is first trained in D^S. Starting from the resulting parameter set, the classifier is then further trained on the available training data in D^T. This strategy has proven to result in better performing classifiers in different applications.

Unsupervised Domain Adaptation: The complementary case of supervised DA, in which no reference labels are available in D^T, is known as *unsupervised DA* (UDA) in Computer Vision (Wang and Deng, 2018), while in RS it is sometimes referred to as *semi-supervised DA* (Tuia et al., 2016). In this setting, only images from D^T can be used to adapt a classifier from D^S to D^T. It is particularly interesting because images from the target domain are available in the scenario where a classifier is to be adapted to a new domain in order to perform a classification in that domain. UDA is known to be very challenging and can even lead to a negative transfer, i.e. to a lower performance in the target domain after adaptation compared to the classification performance of a classifier that was trained only in the source domain. The opposite case in which the classifier achieves a higher performance in D^T after UDA is referred to as positive transfer.

This thesis addresses homogeneous UDA for the classification of aerial and satellite imagery. UDA is of great importance when it comes to the classification of such data because, on the one hand, there is often a very limited amount of freely available data with annotations (Zhu et al., 2017b) and, on the other hand, the appearance of both natural and man-made objects has a large variability, making it difficult for a classifier to perform well when applied to a different domain than the one on which it was trained. In a RS application, this corresponds, for example, to a situation where labelled images from one area (source domain) are to be used to classify images of another area (target domain) taken with the same type of sensor and considering the same class structure.

2.6.1 Adaptive Batch Normalisation

One method for UDA, referred to as *adaptive batch normalisation* (ABN) (Li et al., 2018), is particularly relevant for this thesis, because it is used as additional adaptation step in one variant of the proposed method.

The common procedure when using 2D batch normalisation layers in a FCN is to track running averages of the means and standard deviations of the activations in each channel (cf. Section 2.3.3). The set of all running averages approximates the statistics of the images from D^S, i.e. the average values and standard deviations of the activation values, when computed for a whole dataset (domain). Li et al. (2018) argue that in a domain adaptation scenario, it may not be reasonable to use the statistics that were calculated based on the images from the source domain, because the statistics of the target domain may be different. Based on this consideration, Li et al. (2018) propose to compensate for the domain gap by adjusting the running averages in the 2D batch

normalisation layers of a classifier. In particular, the core idea of ABN is to recalculate the running averages using the data points from D^T such that they approximate the mean and standard deviations of the activations obtained for the data from the target domain. This approach is illustrated in Figure 2.12.

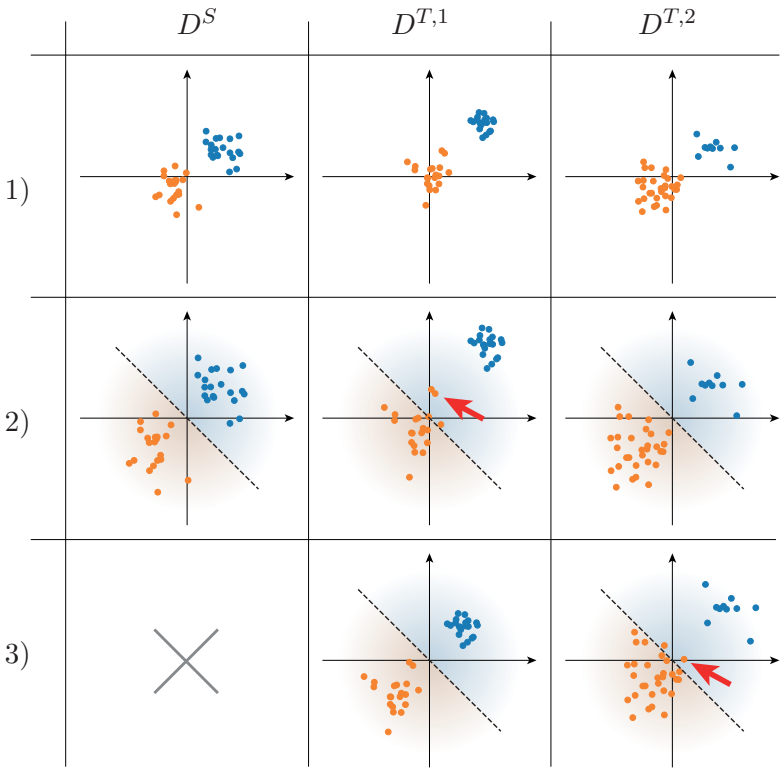

Figure 2.12: Toy example illustrating the working principle of adaptive batch normalisation. 1) 2D data points from the domain according to the column. 2) Data normalised using the means and standard deviations from D^S. 3) Data normalised using means and standard deviations from D^T. The broken line illustrates the decision boundary learned in D^S. The colour of the dots illustrates the true class. Red arrows point to misclassified samples.

The figure illustrates a toy example in which 2D feature vectors are to be classified into two classes, *orange* and *blue*. In the example, the classifier applies batch-normalisation to the input before feeding the normalised features to a linear classifier. In the source domain, there are 20 samples for each class, cf. upper-left plot in Figure 2.12. The figure underneath that one shows the data after normalising each feature such that the average is zero and the standard deviation is one. This plot also shows the learned decision boundary that correctly separates the two classes. The first target domain $D^{T,1}$ is assumed to follow the same global label distribution, i.e. there are again 20 samples per cluster. However, the clusters are slightly shifted compared to the data in D^S (cf. upper-middle plot in Figure 2.12). Applying the standard procedure of batch normalisation would mean to normalise the data from $D^{T,1}$ using the running averages of means and standard deviations learned during training in D^S. The result of normalising the data from $D^{T,1}$ using those values is illustrated in the central subfigure in Figure 2.12. The red arrow in this part of the figure points at two samples which would be misclassified using this approach. The middle in the last row shows the result of applying ABN, i.e. the data from $D^{T,1}$ after recalculating the running averages using the data from $D^{T,1}$. In this case, all samples would be classified correctly. Note that in the example,

the means and standard deviations of all samples from $D^{T,1}$ are used for the normalisation. The approach of ABN is, however, susceptible to the global label distribution, which is illustrated using a second target domain $D^{T,2}$. The data points in $D^{T,2}$ are drawn from the same distribution as those in D^S, but this time there are 30 orange samples and 10 blue samples, cf. upper-right panel of Figure 2.12. Applying the standard procedure of batch normalisation (cf. right panel in the second row in Figure 2.12) would not result in a misclassification, but applying ABN does. In particular, if the data points are normalised according using the means and standard deviations over $D^{T,2}$, the features are shifted into the direction to the upper-right corner, which causes some of the orange samples to be misclassified.

Note that in a real FCN there are usually several batch normalisation layers, which makes the outcome less intuitive. However, the example should illustrate the core idea of ABN as well as possible limitations.

3 Related Work

In this chapter, related work relevant to this thesis is presented and discussed with a focus on unsupervised domain adaptation (UDA) for image classification. UDA for image classification has been studied in the context of classical machine learning for quite some time (Pan and Yang, 2009; Tuia et al., 2016). However, in recent years, with Deep Learning (DL) having emerged as the new state of the art in machine learning, research in UDA has mainly focussed on developing methods that can not only be applied to DNNs, but also exploit the new capabilities of DL such as adversarial training and image generation (Wang and Deng, 2018; Xu et al., 2022). Although many approaches to UDA are based on concepts developed before the era of DL, the following review focusses on recent publications that address UDA in the context of DL, in the following also referred to as *deep UDA* (Wang and Deng, 2018; Xu et al., 2022; Csurka, 2020).

Sections 3.1 to 3.4 provide an overview of publications that address UDA in the context of DL, grouped according to the main adaptation strategy. The groups essentially follow those in (Xu et al., 2022), although different names are used here. The first group of methods is based on *instance transfer*, referred to as *self-training methods* in (Xu et al., 2022), and will be discussed in Section 3.1. Section 3.2 deals with methods based on *representation transfer*. A subset of these methods is discussed in the context of *adversarial training methods* in (Xu et al., 2022). In Section 3.3, methods based on *appearance adaptation* are discussed, which are referred to as *generative training methods* in (Xu et al., 2022). The last group of methods, covered in Section 3.4, consists of those that combine different adaptation concepts and therefore are referred to as *hybrid approaches*. Section 3.5 discusses the aspect of parameter selection in the context of deep UDA. Finally, Section 3.6 draws conclusions from the discussion of the state-of-the-art, identifies research gaps to be closed by this thesis, and and discusses the most similar works.

3.1 Instance Transfer

Instance transfer aims to adapt the classifier from the source domain to the target domain by using semi-labelled samples, i.e. target samples that get their class labels from the current state of the classifier, e.g. (Bruzzone et al., 2006). In *explicit instance transfer*, the semi-labels are actually predicted and used as reference to compute a supervised classification loss in D^T, which is minimized to adapt the classifier to D^T. A successful adaptation based on explicit instance transfer requires many of the semi-labels used for re-training the classifier to be correct. In many approaches, this is addressed either by improving the semi-label generation scheme, for example by making redundant class predictions for each pixel to improve the quality of the semi-labels, e.g. (Iqbal and Ali, 2020), or by a subsequent sample selection strategy to select those semi-labels from all predictions in D^T

which have a high chance of being correct, e.g. (Subhani and Ali, 2020). A different but related approach is *implicit instance transfer*. Such approaches are based on the idea of minimising the entropy for predictions in the target domain, e.g. (Vu et al., 2019). Since entropy minimisation is equivalent to increasing the probability of the most likely class, this approach is very similar to minimising a supervised classification loss with semi-labelled samples, as it is done in explicit instance transfer.

3.1.1 Explicit Instance Transfer

Explicit instance transfer is frequently used to perform DA for the pixel-wise classification. For example, it is frequently used for the task of street scene segmentation with CNNs. In particular, the following works address an adaptation scenario in which synthetic images are available in D^S, but real images are to be classified in D^T. Zou et al. (2018) propose class-balanced self-training. They train a network jointly on labelled source domain data and target domain samples with semi-labels. For each class, they select the semi-labelled samples with the highest confidence scores, i.e. measured by the predicted probabilistic class score for the respective class. Selecting the same number of samples per class turns out to be necessary when dealing with an imbalanced class-distribution in D^T. Based on the source domain samples, they also calculate a spatial prior for each class, which is used to regularise the classifier. Although this approach yields good results in street scene classification, the use of a spatial prior does not seem to be applicable when classifying aerial images or satellite images, as objects may be located anywhere in the image. Another approach based on semi-labelled samples is presented in (Iqbal and Ali, 2020). The authors first train a classifier in the source domain. Afterwards, the classifier is re-trained using semi-labels that were predicted with a high confidence score. The semi-labels are acquired by making redundant predictions for each pixel at different image resolutions and the confidence score is measured based on the agreement of the redundant predictions. However, the sampling strategy of Iqbal and Ali (2020) is based on the assumption that the marginal label distributions $P(y)$ are similar in all images and in both domains. This assumption may not hold in remote sensing applications where, depending on the image size, some images may only show objects that correspond to a single class. A similar argument applies when assessing the work of Lian et al. (2019) and Zhang et al. (2021a). In (Lian et al., 2019), a *pyramid curriculum* approach is proposed to aggregate semi-labels at multiple scales, and Zhang et al. (2021a) perform a soft weighting of the semi-labels instead of a hard selection. However, both works rely on constraining the distribution of predicted labels in D^T, which may not be useful in an adaptation scenario where the label distributions in D^S and D^T are very different. Subhani and Ali (2020) use multi-scale predictions to assess the reliability of predictions in D^T, and to select a pre-defined number of best semi-labels per class for re-training. Consequently, this approach requires the initial classifier before UDA to make proper predictions for all classes in D^T, which cannot always be guaranteed, because the size of the initial performance gap is unknown. In (Zhang et al., 2019b), the activation vectors of the penultimate layer of the classifier for each class are clustered based on images from D^S. Using these clusters, the semi-labels are obtained by assigning the activation vectors of the penultimate layer of the classifier for pixels from images from D^T to the closest centroid. This should prevent problems due to label imbalance in D^T. However, the cluster assignment may fail if the domains are very different. In (Hoyer et

al., 2022), an approach to UDA based on explicit instance transfer was proposed, achieving a high performance for the task of adapting a classifier from synthetic images to real images in the context of street scene classification. However, the contribution of that work lies in improving the initial performance of the model before re-training, which highlights the importance of adequate initial performance for successful adaptation. Although the approach of Hoyer et al. (2022) outperforms several previous methods, there remains a considerable performance gap after UDA. Sakaridis et al. (2019) address an adaptation scenario that is also related to street scene segmentation but does not correspond to adapting from synthetic to real images. Instead, they adapt from day-time to night-time images and introduce an intermediate *twilight* domain. They show that adapting from the source domain to the intermediate domain before adapting from the intermediate domain to the target domain results in a better classification performance in the target domain compared to directly adapting from the source to the target domain. However, such an approach is limited to situations where such an intermediate domain exists.

Few works explore explicit instance transfer for RS applications in the context of deep UDA. Tong et al. (2020) and Wang et al. (2022) propose schemes for semi-labelling and sample selection to improve the transferability of a FCN for pixel-wise classification. However, both works address cross-sensor adaptation, where deep UDA is primarily used to adapt between different image resolutions. Accordingly, their methods are based on multi-scale predictions to overcome this domain shift. The approach presented in this thesis does not address this adaptation scenario as it is assumed that the ground sampling distance (GSD) is commonly known in RS applications. Therefore, in this thesis, it is proposed to compensate for different GSDs by spatial resampling of the data from the source domain before training instead of addressing this difference using UDA.

3.1.2 Implicit Instance Transfer

An approach for implicit instance transfer was presented in (Vu et al., 2019). Here, the entropy values of the class predictions for each pixel are minimized, which corresponds to increasing the predicted probability for the most likely class. This is conceptually similar to the supervised training on semi-labelled samples because the semi-labels also correspond to the most likely classes per pixels. This means that, in order to correctly predict the semi-labels, the classifier also has to increase the probability for the most likely class per pixel. In addition to direct entropy minimization, Vu et al. (2019) propose an adversarial approach in which the so-called weighted self-information maps of source and target domain samples are aligned by adversarial training using a discriminator network. The weighted self-information maps basically contain the information about the entropy for each pixel as predicted by the classifier. In both cases, the classifier is further regularized with respect to the predicted class distribution of the target domain, which is assumed to be close to the class distribution of the source domain. Vu et al. (2019) report that the weights of the entropy loss terms must be carefully tuned to avoid a classification result that is biased towards some classes. Evaluated on the task of street scene segmentation, they show that the adversarial approach slightly outperforms the direct entropy minimization, while both methods achieve results that are comparable to those of recent methods based on representation transfer. As the adversarial approach of entropy minimization basically aligns the local distributions of the entropy maps from

both domains, it is required that both domains actually have a similar local distributions of the label maps (cf. Section 4.5). This may be the case in the context of street scene segmentation, but it may not be the case in a RS adaptation scenario. Chen et al. (2019a) modify the non-adversarial variant of entropy minimization by replacing the average entropy loss with a novel loss, referred to as maximum squares loss. This loss is defined as the negative square sum of the predicted probabilities. They argue that this loss is advantageous because the gradient of this loss with respect to the predicted probabilities changes linearly with the predicted probabilities and, thus, does not focus too much on very well classified samples. They show that their variant is less affected by class imbalance in D^T and outperforms adversarial and non-adversarial entropy minimization on a street scene classification benchmark. It remains unclear if this approach would yield satisfactory results in a RS scenario. Huang et al. (2020) implement adversarial entropy minimization using a gradient reversal layer. In this way, they force the classifier to reproduce patterns in the entropy maps predicted for images from D^T that are typical for D^S. Although this seems reasonable in street scene classification maps, where the scenes are quite similar with respect to the local label map distributions of both domains, this approach might fail in a RS scenario where the domains have different local label map distributions.

In our previous work (Wittich, 2020), an implicit approach for instance transfer was proposed and evaluated in the context of a RS scenario, namely the pixel-wise classification of aerial images. The approach is also based on entropy minimization, but in order to avoid a classification bias towards over-represented classes in D^T, the loss is weighted on a pixel-level using weights derived from the semi-labels. In particular, pixels that are close to a object boundary in the predicted label maps and pixels that are assigned to an overrepresented class will receive a lower weight and, thus, contribute less to the overall loss. It was shown that a small positive transfer could be achieved, but it was also found that in most cases entropy minimization results in a negative transfer if the minimization is performed for too many iterations. This is problematic, because tuning the number of iterations is difficult, as the ideal number of iterations is different from scenario to scenario.

3.1.3 Hybrid Instance Transfer

Michieli et al. (2020) use a variant of adversarial entropy minimization. They do not only match the distributions of the entropy maps, but they also use the entropy as a measure for the confidence of the predictions in D^T for each pixel. The resulting confidence maps are then used to select semi-labels to be used for re-training. Li et al. (2020a) combine implicit instance transfer by entropy minimization with an explicit re-training approach. For the sample selection, they use an approach similar to (Zhang et al., 2019b) to assess the structural similarity of the local neighbourhood in image patches from both domains. Pan et al. (2020) propose a two-step procedure for UDA. In the first step, adversarial entropy minimization is used to roughly align the domains. Based on the average entropy for images from D^T, the images are split into two sets corresponding to easy and hard samples. In the second stage, adversarial entropy minimization is used for the hard samples, while simultaneously a variant of explicit instance transfer based on the easy samples is applied. It remains questionable if these approaches based on hybrid instance transfer can be transferred to RS applications, as they rely explicitly or implicitly on a similarity in the local distributions of the

label maps in both domains, which may not be the case in RS. Besides, directly minimising the entropy in D^T only works well if the initial predictions in the target domain are largely correct, which is difficult to guarantee.

3.1.4 Discussion

Methods for UDA based on instance transfer have been shown to work quite well some applications of pixel-wise classification. However, these methods heavily depend on a good initial performance of the classifier in D^T after training in D^S. This can be problematic if the domains are very different, especially because commonly used DNNs tend to make wrong predictions with a low entropy for samples that are far away from the training data in the feature space (Hein et al., 2019). Furthermore, Arazo et al. (2020) point out that naïve instance transfer is likely to fail as the classifier tends to overfit to the semi-labels. This can be solved by restricting the label distribution in D^T, e.g. requiring it to be similar to the label distributions in D^S. However, such an approach may fail if the respective label distributions are different. Methods that rely on adversarial entropy minimization basically align the distributions of patterns in the entropy maps of images from both domains. This is reasonable if the underlying label maps are similar with respect to these patterns. However, if this is not the case, such an approach is likely to lead to a negative transfer, as the classifier is forced to generate patterns that do not match the image data in D^T.

Based to these arguments, the approach presented in this thesis is not based on instance transfer. However, the strategy of entropy minimization proposed in Vu et al. (2019) is used as a baseline to which the method proposed in this thesis is compared in the experiments.

3.2 Representation Transfer

The second approach for UDA is referred to as *representation transfer*. The main idea is to map images from both domains to a shared representation space in which a shared classifier can be applied. This is either achieved by a shared mapping function that extracts domain-agnostic features, or by using two separate mapping functions for D^S and D^T, respectively. In the context of DL, the single mapping function or the domain specific mapping functions are implemented as DNNs, in this section referred to as *representation encoders*. The output of the representation encoders are referred to as *representations*. Representation transfer can be grouped according to the way in which the representations from both domains are aligned. In *non-adversarial representation transfer* the representations are aligned by minimizing a predefined similarity metric, while in *adversarial representation transfer*, the alignment is achieved by adversarial training.

3.2.1 Non-adversarial Representation Transfer

Non-adversarial representation transfer is often done by finding representations that minimizes a statistical distance between the domains, e.g. the maximum mean discrepancy (MMD) (Matasci et al., 2015) or variants of it (Liu and Qin, 2020). This approach was transferred to CNNs in (Long et al., 2015b) for the task of assigning a single class label to an image (images categorization). A

similar approach was proposed by Sun et al. (2016), who maximise the correlation of representations by minimising an according loss function. In (Sun and Saenko, 2016) it was extended for end-to-end training. Several follow-up works propose further variants of correlation alignment (Morerio et al., 2018; Zhang et al., 2018b; Cheng et al., 2021) for UDA in image categorization. Guo et al. (2016) propose to project features from both domains to a common representation space by transforming the representations with a linear kernel, and Haeusser et al. (2017) align latent representations for images from D^S and D^T by minimizing an association loss. Zhang et al. (2019c) introduce the margin disparity discrepancy (MDD), a theoretically founded metric to measure the distance between representations, and propose a loss to minimize this metric to align the domains.

Although these methods were quite successful in the context of image categorization, they may not work well for the task of pixel-wise classification, as the representations for neighbouring pixels are highly correlated and therefore cannot be considered to be independently and identically distributed, which is assumed, for example, in MMD based alignment (Matasci et al., 2015). This potential problem was empirically confirmed e.g. in (Zhang et al., 2021b). The authors used a variant of correlation alignment for the pixel-wise classification of aerial images and found this approach to be inferior to other approaches such as adversarial representation transfer (cf. Section 3.2.2).

Nevertheless, some successful approaches for UDA in pixel-wise classification can be categorized as being based on non-adversarial representation transfer. However, such methods perform the representation matching in a more implicit way. One non-adversarial approach to UDA for pixel-wise classification was proposed by Kang et al. (2020). The authors suggest to find representations of pixels from images coming from the source and target domains that are similar with respect to a hand-crafted metric and to make these representations even more similar by adjusting the encoder of the network. They empirically show that this approach can reduce the performance gap to some extent in a street scene classification scenario, but it remains unclear whether this approach can be transferred to RS applications. Another non-adversarial method is *adaptive batch normalization* (ABN) (Li et al., 2018), cf. also Section 2.6.1. The authors use batch normalization layers in a FCN to perform the adaptation of a pre-trained classifier to D^T. In particular, they recalculate the batch normalization parameters using statistics derived from target domain images, which aims at aligning the distributions of output activation maps of the batch normalization layers. The approach is evaluated for the pixel-wise detection of clouds in satellite imagery and shows much better performance than other non-adversarial representation transfer methods such as correlation alignment. ABN has further practical advantages. On the one side, it is very fast as it does not require any training beyond the initial training in D^S. On the other hand, as mentioned by Li et al. (2018), ABN can easily be combined with other methods for deep UDA. In this thesis, ABN is used in two ways. First, it is used as a baseline to compare the method proposed in this work against. Second, as ABN follows a different strategy of UDA compared to the proposed method for appearance adaptation, it is also examined whether the two methods can be combined to achieve better performance in the target domain.

3.2.2 Adversarial Representation Transfer

The second way to perform representation transfer is based on adversarial training of a domain discriminator and either a shared representation encoder or a representation encoder for the target domain. In the latter case, representations for images from D^S are obtained by feeding them to a representation encoder with fixed weights obtained by supervised training in D^S. In both cases, the domain discriminator network is trained to distinguish between representations for images from the two domains by predicting the correct domain label. Simultaneously, the shared representation encoder (or the one for the target domain) is trained to fool the discriminator by predicting representations for images from D^T such that they are classified by the discriminator as coming from D^S. This concept was first introduced for image categorization (Ganin et al., 2016; Tzeng et al., 2017). Ganin et al. (2016) implemented the adversarial training using a gradient reversal layer and a shared representation encoder and showed that adversarial training outperformed the non-adversarial approach of minimizing the MMD. Tzeng et al. (2017) used two separate representation encoders for the two domains, which outperformed the approach by Ganin et al. (2016).

In recent years, some methods for UDA for image categorisation based on adversarial representation transfer have been developed that are difficult to be transferred to the task of pixel-wise classification. For example, Wei et al. (2021) proposed an extension of adversarial representation matching for image categorization by optimising some of the hyper-parameters related to the adaptation in training. To this end, they use concepts from the field of meta learning. Particularly, they treat the domain alignment as meta training task and the classification on data from D^S as meta test task. Transferring this approach to pixel-wise classification may be not practical, because the meta optimization step would lead to a massive computational overhead as in each meta training step the classifier needs to be trained. Hu et al. (2018) and Wang et al. (2020a) use a multi-class discriminator in adversarial training. Here, instead of predicting only the domain from which the input image originates, the discriminator should also predict the class label of the image. As the discriminator has to learn to distinguish the classes, it becomes class-aware, which, according to the authors, has positive effects on the domain adaptation. Transferring this approach to pixel-wise classification would be equivalent to using a pixel-wise classifier as discriminator, which would lead to a very large memory footprint of this DNN. Moreover, adversarial training aligns the marginal distribution of the representations, which can be problematic if the label distributions in D^S and D^T are very different. Making the discriminator explicitly class-aware could amplify this problem, because the discriminator has to consider the class structure and thus could easily make predictions based on the label distributions.

Besides the developments in the field of adversarial representation transfer for image categorization, this concept has been well studied in the field of pixel-wise classification, often for the pixel-wise classification of street scenes. Note that here, the representations usually correspond to 2D activation maps. The first publication in this group was (Hoffman et al., 2016). Similarly to (Ganin et al., 2016), a shared representation encoder is trained in a domain-adversarial manner, but instead of predicting a single class label for each representation, the representations are processed by a single transposed convolutional layer in order to make pixel-wise class predictions. Huang et al. (2018a) use separate representation encoders for D^S and D^T and align the representations

simultaneously at multiple stages of a FCN. Particularly, they feed the respective activation maps to several domain discriminators and perform a joint adversarial training of all discriminators and the representation encoder in D^T, while the parameters of the representation encoder for the source domain are not changed. In the adaptation process, both approaches explicitly restrict the distribution of labels predicted by the current model in D^T to be similar to the label distribution in D^S. This implies the assumption that the respective underlying distributions are indeed similar, which may not be true in an adaptation scenario in RS.

Many follow-up works based on adversarial representation transfer deal specifically with the task of street scene classification. Similarly to Huang et al. (2018a), Tsai et al. (2018) perform adversarial adaptation at two layers of the FCN, namely at an intermediate layer of the network and at the last layer of the network. Unlike Huang et al. (2018a), they use a shared encoder for both domains. Hong et al. (2018) align the representations by jointly training the domain discriminator and a network that modifies the representations of images from D^T, such that the modified representations are similar to those for images from D^S. In particular, the latter network predicts a residual, which is added to the initial representations for images from D^T. Luo et al. (2019a) also apply adversarial representation transfer but restrict the representations to contain only necessary task-specific information, i.e. the information required to perform the classification, to make the task more difficult for the discriminator. In particular, they propose two loss terms that regularise the distributions of the representations for images of both domains. Another approach, conceptually very similar to (Huang et al., 2018a), was proposed in (Shan et al., 2020). The authors perform representation transfer at multiple layers of a network, but instead of using multiple discriminators, they concatenate the activation maps from different layers and process them by a single discriminator. Besides these variations of basic adversarial representation transfer, there are more sophisticated approaches, but they were not evaluated for a RS application. Luo et al. (2019b) propose to weight the adversarial loss at a pixel level, deriving the respective weights from measuring the agreement of two classification networks that are trained in parallel. In this way, pixels that are more difficult to classify should receive a larger weight. According to the authors, this should avoid an incorrect alignment of source and target domain representations of classes that are initially aligned poorly. In (Tsai et al., 2019), the representations are first clustered in D^S to obtain a so called cluster space to which each feature can be mapped. Then, domain adversarial training is performed, in which each representation is first transformed into the cluster space to reduce the dimensionality of the input space of the discriminator. Tsai et al. (2019) argue that adversarial alignment of lower dimensional representations has positive effects on the optimisation. However, it does not solve the problems related to different label distributions in the two domains. Du et al. (2019) perform a class-wise adversarial matching, relying on semi-labels in D^T. Thus, this method inherits the possible problems of instance transfer i.e. the requirement of a very good initial performance of the model in D^T after pre-training in D^S. The above-mentioned methods have in common that they solely address UDA for street scene classification in which, both, the local distributions of label maps and the marginal label distributions in D^S and D^T are rather similar. Many approaches exploit this similarity in the form of loss terms (Hoffman et al., 2016; Ganin et al., 2016) or in terms of the main adaptation principle (Tsai et al., 2019). This is seen problematic

when such methods are applied to adaptation scenarios in which such assumptions may not always apply.

There exist several works that address UDA based on adversarial representation transfer for pixel-wise classification outside the domain of street scene classification. Mei et al. (2020) transfer the concept to medical image segmentation. Three discriminators are used to match the distribution of activation maps at different layers of a joint FCN, which is conceptually very similar to (Tsai et al., 2018). Although not addressing street scene classification, a high similarity of the distributions of label maps in the two domains can be expected in this adaptation scenario, as the images of both domains depict the same objects and differ only in acquisition time, lightning conditions and the method for medical image acquisition.

Several publications address adversarial representation transfer for the pixel-wise classification of remotely sensed imagery. An example for UDA based on representation transfer in RS is (Riz et al., 2016). Here, a domain-independent feature representation from images of two geographical areas is obtained by training a stacked auto-encoder using images from both domains that learns to reconstruct the input image via a lower-dimensional feature space. This seems to work well for domains that are rather similar, but it remains unclear whether it would still be sufficient in the presence of larger domain differences. Gritzner and Ostermann (2020) perform representation transfer based on a domain distance for the pixel-wise classification of aerial images. Their results show that the adaptation performance strongly decreases if the class distributions are very different in the two domains. To improve the adaptation performance, they align the representations using target images found to be semantically similar according to the label maps predicted by the classifier trained on source domain data, but this only leads to an improvement in half of the experiments presented. Liu et al. (2020) aim at representation transfer by matching so-called feature curves from both domains using adversarial training. However, the domain gap that could be bridged by this method was rather limited. This indicates that adversarial representation transfer is difficult if the domain gap is large, e.g. when adapting between two different cities where the objects have a different appearance or where the label distributions are dissimilar, both of which is the case for the public benchmark dataset used in (Liu et al., 2020). This approach is used as a further baseline in the experiments of this thesis. In (Wittich and Rottensteiner, 2019) representation transfer based on adversarial training for UDA was also used. A small but stable improvement of the classification results could be achieved if an early network layer was chosen for transfer. However, the results strongly depended on the hyper-parameters used in training, which makes this approach difficult to tune. Noa et al. (2021) extended such an approach for the bi-temporal deforestation detection based on satellite imagery by an advanced constraint between the encoders for $D^S$and D^T, which resulted in a positive transfer in all evaluated scenarios. However, in some scenarios a rather large performance gap remained after UDA. In (Lee et al., 2021), adversarial representation transfer is applied to binary pixel-wise building classification in aerial images. After pre-training a classifier in the source domain, the representations obtained from a layer in the middle of the network are matched using adversarial training. The authors propose to restrict the encoder for D^T so that the representations can be used to reconstruct the samples from D^T. In particular, a decoder is introduced that should reconstruct the original images from D^Tbased on the reconstruction, while simultaneously performing adversarial representation alignment. This approach has been successful

for building classification, but in these adaptation scenarios the structural similarity is quite large and the marginal label distributions are quite similar. For example, the authors consider the Inria Aerial Image Labeling Dataset (Maggiori et al., 2017) and the Massachusetts Buildings Dataset (Mnih, 2013) in their adaptation scenarios, which contain 15.8 % and 13.2 % of building pixels, respectively. It remains doubtful whether such an approach can work in a more difficult multi-class adaptation scenario like land cover classification, especially when the marginal label distributions and the local label distributions are very different in D^S and D^T.

3.2.3 Discussion

In the literature reviewed, a frequently used strategy for UDA is to align the distributions of representations of samples from D^S and D^T, respectively, such that a shared classifier can be applied to them. Adversarial training has been shown to outperform previous approaches that minimise hand-crafted metrics for measuring the dissimilarity between representations for images from both domains. Especially in the field of street scene classification, adversarial representation transfer has been extensively studied and several variants and modifications have been proposed. However, the success of such methods in RS applications was rather limited. Tsai et al. (2018) state that adversarial representation matching using layer in the middle of a FCN as representations is inherently difficult due to high dimensionality of the feature space. As pointed out in (Zhao et al., 2019), having similar label distributions in D^S and D^T is an important factor when trying to learn invariant representations. In their work, they conclude that having a high feature similarity between the two domains and a low classification loss in D^S is not necessarily sufficient for a good performance in D^T. This is consistent with commonly made assumptions about the domains, such as a similar label distributions in D^S and D^T (Hoffman et al., 2016; Huang et al., 2018a). However, such assumptions are not generally justified in RS applications. Some methods make no explicit assumptions about a high similarity label distributions, but as they were evaluated only in adaptation scenarios in which the label distributions are actually high, it remains unclear whether these methods would work in adaptation scenarios with different characteristics. In particular, it is supposed that in adaptation scenarios with unequal distributions of labels in D^S and D^T adversarial training leads to an incorrect alignment of representations, so that the shared classifier does not perform well in D^T. Although similar arguments hold for appearance adaptation (cf. Section 3.3), the latter strategy is preferred in this work because the adapted images can be assessed visually, which is advantageous when developing a method and tuning its hyper-parameters. Consequently, the main approach presented in this work is not based on representation transfer but on appearance adaptation. Furthermore, the appearance adaptation based approach is combined with adaptive batch normalization (Li et al., 2018), which was shown to perform well in a RS application. This method for representation transfer is favoured, as it is very fast, has very few hyper-parameters and does not require any training with gradient descent.

To assess the performance of adversarial representation matching, an UDA approach based on adversarial domain adaptation similar to (Tsai et al., 2018) is evaluated in the experiments, too. Furthermore, adaptive batch normalisation (Li et al., 2018), an approach for UDA based on rep-

resentation transfer that addresses an RS application is used as another baseline to compare the proposed method to in the experiments.

3.3 Appearance Adaptation

The third group of methods for UDA apply the domain adaptation to the original images. They make use of methods for appearance adaptation, which aim to create modified versions of images from either D^S or D^T that look similar to the images from the respective other domain (cf. Section 2.5). The appearance adaptation is performed using FCNs, which are referred to as *appearance adaptation networks*. In the literature, this technique is used in two variants. In the first variant, images from D^T are adapted to match the appearance of images from D^Sbefore being fed to a classifier that was trained in D^S. In this thesis, this strategy is referred to as target-to-source adaptation, and methods based on this strategy are discussed in Section 3.3.1. In the second variant (cf. Section 3.3.2), images are adapted from D^Ssuch that they look like images from D^T. The adapted images are then used in combination with the original label maps to train or fine-tune a classifier to perform well in D^T. This strategy is referred to as source-to-target adaptation.

3.3.1 Target-to-Source Appearance Adaptation

This group of methods can be seen as a special case of representation transfer, where the shared representation space is the original feature space of the images of the source domain. Soto et al. (2020) use this concept for bi-temporal deforestation detection based on satellite imagery (cf. 2.1). Particularly, they train a so-called CycleGAN (Zhu et al., 2017a) which makes use of cycle consistency (cf. Section 2.5) to perform the appearance adaptation in both directions. Note that here, the input and output of the appearance adaptation networks are composite images that contain image pairs. Palladino et al. (2020) propose the same approach for medical image segmentation, i.e. for the detection of white matter in brain scans. While Palladino et al. (2020) achieved quite good results for medical image segmentation, Soto et al. (2020) found out that cycle consistency is not sufficient for achieving semantic consistency in the addressed RS application, as the adapted images from D^T tend to show patterns that appear more frequently in D^S. This phenomenon is referred to as *hallucination of structures* (Cohen et al., 2018). To avoid this problem, Soto et al. (2020) propose to add an identity regularization term ensuring the source-to-target appearance adaptation network to perform an identity mapping when fed with an image from the source domain. In the same way, the target-to-source appearance adaptation network should perform an identity mapping when fed with an image from the target domain. When evaluating this method on a single adaptation scenario, the resulting approach achieved a small positive transfer on average, but with a rather high standard deviation due to random influences such as the sampling order of patches and the random initialisation of the models.

In (Soto et al., 2021), the approach of (Soto et al., 2020) is extended by an additional loss term to constrain the appearance adaptation networks in the CycleGAN architecture. Soto et al. (2021) exploit that in bi-temporal image classification the input and output of the appearance adaptation networks is a composite of two images and consider the difference between the earlier and the later

image in the composite image. In particular, the proposed loss restricts the change of the differences due to the appearance adaptation. Using this extension, the authors could slightly outperform the previous variant. However, this extension is restricted to applications in which the images to be classified are such composite images of the same region. In the experiments, the method proposed by (Soto et al., 2021) will serve as another baseline to compare the proposed method to.

Pandey et al. (2020) address the task of pixel-wise classification of images of persons to differentiate skin and background. In the application, the target domain consists of unlabelled infrared images and the source domain contains labelled red-channel images. They propose to generate an artificial red channel image from the infrared image by latent space optimization of a variational auto encoder, pre-trained on images from D^S, with respect to a structural similarity loss. This approach might be reasonable in the application addressed, but it requires a high similarity with respect to the scene contents in both domains. Otherwise it is likely to fail as the content of images from D^T may not be properly represented by a variational auto encoder.

3.3.2 Source-to-Target Appearance Adaptation

Again, using concepts from appearance adaptation, approaches from this group of methods take an image from D^S and adapt it to look like an image from D^T. As the training labels for images from D^S are available in UDA, a classifier can be trained in a supervised way based on the adapted images. Maintaining *semantic consistency* as defined in Section 2.5 is crucial for the success of this strategy. Some authors addressing street scene classification try to achieve this goal in the frequency domain, where the adaptation is applied only to the amplitude component of the images, either based on learning a corresponding mapping (Yang et al., 2020c) or by swapping the low frequency coefficients between the source and target images (Yang and Soatto, 2020). In RS, however, it may sometimes also be required to consider modifications of higher frequencies, e.g. when transferring between domains corresponding to images acquired at different seasons, in which deciduous trees look completely different.

Hoffman et al. (2018) first proposed to use a CycleGAN architecture for the appearance adaptation from $D^T$to D^S. As they propose a combination of appearance adaptation and adversarial representation transfer, this contribution is discussed in Section 3.4. Gong et al. (2019) also use a CycleGAN to perform UDA solely based on appearance adaptation. They extend the method by using a continuous domain space for the domain discrimination instead of a binary space. In particular, instead of considering discrete domain labels, for example zero for $D^S$and one for D^T, they consider the domain label to be a real value between zero and one, where zero would correspond to the source domain and one corresponds to the target domain. A value between zero and one would indicate an intermediate domain in which the images have an appearance that corresponds to a mix of the appearance in the source and target domains. This aims at improved appearance adaptations, but it is limited to settings in which using a continuous domain space is actually meaningful.

In RS, Benjdira et al. (2019) used CycleGAN to adapt images from one city to another one, each of them being considered a domain. Their main goal is to learn a semantically consistent image

adaptation between the two domains by incorporating the cycle consistency loss. The method leads to quite large improvements in the classification performance for two out of six classes due to UDA, but the performance of the other classes could hardly be improved. Zhao et al. (2023) also use CycleGAN to adapt images from D^S to D^T, but extend the method by an auxiliary task, which is to predict the height map. This auxiliary task aims at improving the semantic consistency. They consider only the multi-spectral images as input to the classifier and, thus, require the height information only in the training phase but not for inference. Those two methods serve as further baselines in the evaluation of the method proposed in this thesis.

In other works on UDA for RS applications, modifications of CycleGAN are presented to improve the performance of the adaptation. Soto et al. (2020), already mentioned in Section 3.3.1, also evaluate a variant of appearance adaptation in which they use CycleGAN to adapt images from $D^S$to $D^T$and use the adapted images to train the classifier. In their experiments, the variant of using a network for appearance adaptation from the target to the source domain performed significantly worse than the variant of training on source-to-target adapted images from D^S. Gritzner and Ostermann (2020) address the pixel-wise land cover classification based on aerial images. The authors observed that using CycleGAN without modifications resulted in negative transfer in 50 % of their experiments. The authors tried to improve the adaptation by training on semantically paired images (cf. Section 3.2.2), but this did not result in a significant improvement compared to using random images.

There are also strategies to achieve semantic consistency that do not require cycle consistency. Tasar et al. (2020a) learn a colour mapping to perform the image adaptation from the source to the target domain. However, this approach cannot adapt the texture of objects and may therefore be too limited to perform well in more complex UDA scenarios, for example, adapting the appearance of images from different seasons to each other. Tasar et al. (2020b) use a bidirectional image adaptation based on an alternative to cycle consistency called cross-cycle consistency (Lee et al., 2018), and align the gradients of images before and after the adaptation to achieve semantic consistency. However, this may be a too strong regularization when trying to apply UDA to imagery from different seasons, as gradient maps can change significantly in vegetated areas.

3.3.3 Discussion

Comparing the two variants for appearance adaptation, the target-to-source adaptation is less frequently used in literature than the source-to-target adaptation. It is assumed that the target-to-source adaptation is less robust to errors in the image adaptation, as such errors directly affect the classification result. In particular, when classifying an image that was adapted from D^T to D^S using the source domain classifier, areas that correspond to one class in D^T but look like another class in D^S after adaptation will most likely be classified wrongly. However, if images adapted from the source domain to the target domain are used for training, possible errors in the appearance adaptation process may only occur occasionally and, thus, only slightly affect the parameters of the resulting classifier, which can still yield a satisfactory performance in D^T. A further advantage of source-to-target adaptation is that, after adaptation, only the adapted classifier is required to make predictions for images from D^T, while in target-to-source adaptation, both the appearance

adaptation network and the source classifier are required. Thus, source-to-target adaptation reduces the memory footprint during inference and also the inference time.

The method proposed in this thesis is also based on appearance adaptation, particularly on the variant in which the classifier is trained using images from D^S that are adapted to D^T, following the arguments just mentioned. Compared to the publications cited in Section 3.3.2, a different strategy to achieve semantic consistency is proposed. Instead of relying on cycle consistency or cross-cycle consistency, only a single adaptation network that adapts images from the source to the target domain is trained. Conceptually, this differs from all approaches based on CycleGAN, as only a single adaptation network is used. This leads to a smaller memory footprint and also to a reduced number of hyper-parameters, making the approach easier to tune. The proposed method achieves semantic consistency by not only enforcing the adapted source images to look like those from D^S after adaptation, but by also requiring them to be classified correctly after the adaptation.

Furthermore, this thesis proposes two new methods aiming to further improve the semantic consistency in difficult adaptation scenarios, i.e. where the appearance adaptation network tends to hallucinate structures. Existing approaches try to constrain the appearance adaptation networks either by architectural choices, e.g. by only allowing a change of the colour of the images in the adaptation process, or by constraining the adaptation networks with respect to the predictions, e.g. by applying the cycle consistency constraint. In contrast, the methods presented in this work do not regularise the appearance adaptation network, but rather aim to mitigate the reasons for semantically inconsistent appearance adaptations.

3.4 Hybrid Approaches

Some methods proposed in the literature combine instance transfer with representation transfer. Yang et al. (2020b) jointly use entropy minimization and adversarial representation transfer. Wang et al. (2020b) differentiate between *stuff* and *things*, assuming that *stuff* looks rather similar in different domains while *things* look different. They combine adversarial representation transfer with explicit instance transfer, making use of multi-scale predictions to improve the semi-labels for *things*. Both methods were evaluated solely for street scene classification and it remains unclear if the approaches can be transferred to RS. Further, as they rely on instance transfer they require a rather good initial performance of the classifier after source-training.

Several hybrid approaches combine instance transfer and appearance adaptation. Yang et al. (2020a) integrate the appearance adaptation from the target to the source domain (cf. Section 3.3.1) into an hybrid two-step procedure. First, they train a CycleGAN to adapt images from D^T to D^S. In the second stage, they apply a domain discriminator to the output of the classifier to match the representations of images from D^S and images from D^T adapted to D^S. Simultaneously, they train a reconstruction network that takes the label maps as input and reconstructs the input to the classifier. The corresponding reconstruction loss is also used to train the classifier, thus, the classifier has to predicted label maps that allow the reconstruction network to reconstruct the original image. According to the authors, this leads to improved classification results for target

images adapted to D^S. It can be argued that this is not generally applicable, because reconstructing the original input of a classifier based on the predicted label map is an ill-posed problem for which there are several possible solutions. Furthermore, the approach strongly relies on the performance of the target-to-source adaptation which can be problematic as discussed in Section 3.3.3. Previous work has shown that a CycleGAN based approach for appearance adaptation easily fails in RS applications (Soto et al., 2021; Gritzner and Ostermann, 2020).

Pizzati et al. (2020) combine explicit instance transfer with the second variant of appearance adaptation, training the classifier on images adapted from D^S to D^T. Pizzati et al. (2020) use a frozen, pre-trained MUNIT network (Huang et al., 2018b) trained in a fully unsupervised way on images from D^S and D^T, and they extend the datasets by web-crawled data to support the adaptation process from rainy to sunny street scenes. They obtain the final classifier by training on the adapted images from D^T while simultaneously performing explicit instance transfer. Using web-crawled data to overcome a specific domain difference is a meaningful approach in principle, but it requires to explicitly know the reasons for differences between the domains. If these reasons are unknown or even a combination of several aspects the approach is no longer applicable. For example, in RS applications, the domain difference is related to regional differences, which are difficult to define precisely.

Several approaches combine representation matching and appearance adaptation. For example, Zhang et al. (2018a) use gradient-based style transfer (Gatys et al., 2016) to reduce the visual difference between the two domains before feeding the images to the classification network, where adversarial representation transfer is applied. However, this approach leads to high computation times and requires a considerable amount of hyper-parameter tuning to achieve good results. One of the first hybrid approaches combining CycleGAN for appearance adaptation and adversarial representation transfer was CyCADA (Hoffman et al., 2018), applied to street scene classification. It is based on a rather complex network that, according to the authors, cannot be trained end-to-end on a consumer GPU due to very high memory requirements. Musto and Zinelli (2020) extend CyCADA by feeding the predicted label map to the appearance adaptation network along with the images and enforcing consistency between the label maps predicted for both, the original and the adapted source images. Chen et al. (2019b) additionally enforce consistency between the predictions of original target images and adapted target images, whereas Chang et al. (2019) replace cycle consistency with cross-cycle consistency. All these methods are very complex due to the high number of networks, parameters and loss terms. This often requires to train parts of the architecture in different steps, which could be the reason why none of theses methods directly use the classification loss for the adapted source images jointly with the losses related to the adaptation network to enforce semantic consistency.

An architecture very similar to CyCADA was trained end-to-end in (Murez et al., 2018a). The authors train two encoders which embed images from both domains in a shared feature space by adversarial training. Simultaneously, two decoders are trained, which recover images based on the embeddings. To enforce semantic consistency the authors use an identity loss that enforces recovered representations to look like the original inputs. Further, they perform appearance adaptation by decoding representations from each domain to the corresponding other one. The second as-

pect is again achieved via adversarial training, requiring two additional discriminators. The actual classifier is optimized to correctly classify embeddings for the source domain and embeddings for images adapted from D^S to D^T. The approach achieves good results for the pixel-wise classification of street scenes. However, it is unclear if it is transferable to UDA in RS, where domain differences related to the distribution of labels in both domains pose additional challenges (Wittich and Rottensteiner, 2019). Another architecture very similar to CyCADA is suggested in (Li et al., 2019). The authors separate the appearance adaptation (CycleGAN) and the classification part in the sense that the corresponding parameters are updated in an alternating way in the training procedure. To update the parameters of the CycleGAN, the frozen classifier is considered in the form of an additional loss term that enforces similarity of representations obtained for original source images and the adapted source images. On the other hand, when updating the parameters of the classifier, the adapted source images are used for supervised training but also to match the respective representations to those obtained for real samples from D^T. Assessing this approach, it can be argued that it is evaluated only for the classification of street scenes and it remains unclear if the approach can cope with the additional difficulties encountered in RS adaptation scenarios.

Whereas there are many hybrid methods for street scene classification, there are fewer such approaches addressing the pixel-wise classification of remotely sensed images. Ji et al. (2020) combine appearance adaptation and representation transfer, using adversarial training in both cases. For the appearance adaptation they also rely on cycle consistency and adversarial representation transfer is applied in the last layer of the network. As already discussed in Section 3.3, cycle consistency may not be sufficient if the domains are very different. However, since the authors report their results on a publicly available dataset, this approach is used as another baseline for the evaluation of the method proposed in this thesis. In (Zhang et al., 2021b), the authors combine correlation alignment as a method based on non-adversarial representation transfer with explicit instance transfer. They evaluate this combination on a single adaptation scenario addressing aerial image classification, which however only leads to an improvement for some classes. Kwak and Park (2022) combine adversarial representation transfer with explicit instance transfer for crop classification. Their main contribution is in a semi-label selection method that is tailored to crop classification. They could reduce the domain gap by quite a large amount, but the semi-label selection approach is based on an existing parcel boundary map and, thus, not transferable to tasks such as pixel-wise land cover classification. Peng et al. (2022) propose an approach that combines appearance adaptation with both, instance transfer and representation transfer. The authors approximate appearance adaptation by performing a locally adaptive contrast enhancement using a so-called Wallis filter. Furthermore, they use adversarial representation transfer to align the distributions of representations in D^S and D^T. Finally, they use explicit instance transfer by re-training the classifier using semi-labels. The authors obtain good results for the task of building detection. However, as they do not address differences in the label distributions between the domains, it remains unclear if the method would perform well in a more difficult adaptation scenario involving multiple classes.

3.4.1 Discussion

The combination of different approaches for UDA has been frequently studied in recent years. The underlying assumption is that different variants can handle different types of domain shifts and thus complement each other, leading to improved adaptation performance. This assumption was empirically demonstrated, e.g. for applications such as street scene classification. However, the approach proposed in this thesis is not based on a hybrid method including instance transfer or adversarial representation transfer for the following reasons: Instance transfer has the disadvantage that it requires a fairly good initial performance of the classifier in the target domain and can lead to a negative transfer if not strongly regularised (Arazo et al., 2020). In particular, if too many semi-labels are wrong while at the same time they have been predicted with a low entropy, instance transfer is likely to fail as the parameter update is strongly affected by such wrong semi-labels. Adversarial representation transfer is based on aligning the distributions of the representations for images from both domains. As it was shown in the literature, this approach is likely to fail in difficult adaptation scenarios where, for example, the label distributions of the two domains are very different (Tanwani, 2020). Furthermore, combining adversarial representation matching and appearance adaptation usually requires very complex architectures that are difficult to train and to tune. This is often solved by pre-training a CycleGAN for the appearance adaptation, which, however, has been shown to be problematic if there are large differences in the label distributions of the domains (cf. Section 3.3). These disadvantages of adversarial representation transfer and instance transfer probably also apply in a hybrid adaptation variant.

The main method proposed in this thesis is only based on appearance adaptation. However, in a variant it is also combined with adaptive batch normalisation (ABN), a non-adversarial approach for representation transfer. Thus, the extended variant can be considered as a hybrid method. However, the two strategies are combined in a two-step adaptation process, where the first step corresponds to the joint training of the appearance adaptation and the classifier. The representation transfer is then done in the second step by applying ABN, which does not require any training procedure in terms of gradient descent. This approach is assumed to combine the two strategies without drastically increasing the complexity of the architecture and the training procedure.

3.5 Parameter Selection in Unsupervised Domain Adaptation

A problem that is barely addressed in research on UDA is the stopping criterion for the adaptation. In supervised training, it is common practice to monitor the performance of the classifier on a validation set. If this metric stops increasing, the training can be stopped, which is called *early stopping*. Furthermore, the parameter values that lead to the best validation performance are commonly used as the final outcome of the training process (Prechelt, 1998). This strategy is the standard parameter selection method and can be used in combination with early stopping or as an alternative to it. In this thesis, the concept of deciding which parameter set to use is referred to as *parameter selection*. However, in the UDA scenario considered in this thesis, neither early stopping nor the standard parameter selection method can be used because of the lack of labelled samples that could be used for validation in the target domain. Unfortunately, in UDA it is particularly

important to decide for how long the adaptation process is performed. For example, Gritzner and Ostermann (2020) show that the performance on a test set from the target domain decreases after some time if the adaptation is carried out for too long. Benaim et al. (2018) discuss this problem for unsupervised appearance adaptation. However, they do not propose a solution but derive a bound to predict the success of such methods. The common strategy in UDA is to specify the number of epochs for the adaptation and to use the very last parameter set for inference (Tasar et al., 2020b,a; Benjdira et al., 2019; Musto and Zinelli, 2020), which means that this hyper-parameter must be tuned with care. Many publication do not even tell for how many epochs they train their model, e.g. (Murez et al., 2018a; Liu et al., 2020; Hoffman et al., 2018; Chen et al., 2019b; Yang et al., 2020b; Zhang et al., 2021b). This makes it difficult to assess the transferability of such methods to new adaptation scenarios, as the used hyper-parameters regarding the number of training iterations might not be suitable for other domains.

To conclude, the parameter selection has been identified as a substantial factor regarding the performance of UDA. However, to the best of the author's knowledge, the parameter selection is not addressed in existing works for UDA, and consequently, no solution has been proposed in the literature. Therefore, the proposed approach for unsupervised parameter selection in UDA is believed to be the first of its kind in the literature.

3.6 Discussion

In this section, conclusions of the literature review are drawn. In Section 3.6.1, the research gap to be filled by this thesis is identified and it is discussed how the contributions proposed in this thesis address this gap. In Section 3.6.2 the proposed method is compared to the most similar works in the literature.

3.6.1 Research Gap

In the literature, many methods for deep UDA for the pixel-wise classification of images have been proposed that make use of different adaptation strategies. All reviewed methods have in common that they cannot fully compensate for the domain gap, meaning that the classifier that was trained in the source domain and adapted to the target domain performs worse compared to a classifier that was directly trained in the target domain. For example, (Hoyer et al., 2022), which is considered to be the currently best performing method for the adaptation of a pixel-wise classifier from synthetic to real images of street scenes, can reduce the performance gap from 30.8 % in the mean intersection over union to 18.2 %. Thus, a rather large performance gap remains. Similarly, recent methods for UDA addressing the pixel-wise classification of remotely sensed images, e.g. (Zhao et al., 2023), a recent publication addressing the task of land cover classification, report an improvement from 44.3 % to 65.8 % with respect to the mean F1-score. Unfortunately, they do not report the performance of a classifier trained in D^T. However, based on experiments performed by the author of this thesis, the expected performance in D^T is about 80 % in the mean F1-score. The remaining performance gap leaves room for improving methods for deep UDA.

Strategy for Unsupervised Domain Adaptation: Methods based on instance transfer have been shown to be able to reduce the performance gap for some applications to some extent, e.g. (Vu et al., 2019). However, they have the disadvantage that they strongly rely on the initial performance in the target domain of a classifier that was trained in the source domain in order to produce useful semi-labels (cf. Section 3.1). This can be problematic in classification tasks in which the initial performance in D^T is rather poor, which is the reason why the method proposed in this thesis does not use instance transfer.

A commonly used approach in both appearance adaptation and representation transfer is the concept of domain adversarial training (Hoffman et al., 2018; Huang et al., 2018a; Soto et al., 2021; Gritzner and Ostermann, 2020). This training scheme has successfully been used to adapt the appearance of images or to align representations for images from both domains in order to perform the domain adaptation. Although adversarial representation transfer was shown to outperform non-adversarial representation transfer for image categorisation (Ganin et al., 2016; Tzeng et al., 2017) and pixel-wise classification (Huang et al., 2018a) it has some disadvantages. On the one hand, it is rather difficult to tune, on the other hand it is likely to fail if the domains have large differences in the distributions of labels (cf. Section 3.2) (Wittich and Rottensteiner, 2019; Gritzner and Ostermann, 2020).

The strategy of adversarial appearance adaptation has high potential in the context of deep UDA because the appearance adaptation networks can in principle compensate for major differences in the appearance of objects in the two domains. However, there is the major problem of performing the appearance adaptation in a semantically consistent way (Soto et al., 2021; Tasar et al., 2020b), and methods that do not consider semantic consistency result in a poor classification performance (Benjdira et al., 2019). In the literature, many approaches have been proposed to achieve semantically consistent adaptations, as described in Section 3.3. However, they often perform too strong a regularisation of the appearance adaptation network (Tasar et al., 2020a; Soto et al., 2021; Yang and Soatto, 2020; Tasar et al., 2020b), such that adapted source images do not properly represent the style of images in D^T. Thus, achieving a semantically consistent appearance adaptation remains an unsolved problem for scenarios in which the label distributions in the source and target domains are very different.

Consequently, the method proposed in this thesis is based on appearance adaptation and has a major focus on achieving semantic consistency. In particular, in this thesis, UDA is achieved by source-to-target appearance adaptation, i.e. by training on images from D^S that were adapted to D^T. In order to achieve semantic consistency, the source-to-target adaptation network is trained jointly with the classifier, which allows to use the classification loss of the adapted images to constrain the adaptation network. This approach is assumed to already result in semantically consistent appearance adaptation in adaptation scenarios in which there is only a small difference in the label distributions between the domains. As this method requires only a single source-to-target appearance adaptation network and a single discriminator, it is inherently simpler than many existing methods that require a second target-to-source adaptation network, e.g. methods that are based on CycleGAN, such as (Hoffman et al., 2018; Benjdira et al., 2019; Soto et al., 2021; Gritzner and Ostermann, 2020). On the other hand, the appearance adaptation is not limited to

very simple colour mappings, such as in (Tasar et al., 2020a; Yang and Soatto, 2020), because a FCN is used to perform the appearance adaptation. The approach of training a single appearance adaptation network and the classification network jointly is also adopted in existing works (Chen et al., 2019c; Choi et al., 2019), but using more complex architectures with multiple decoders, additional appearance adaptation networks or auxiliary tasks. The proposed method is assumed to achieve semantic consistency with fewer networks and loss terms, thus being easier to tune and having a smaller memory footprint.

In the literature, several works have suggested hybrid methods that combine appearance adaptation and representation transfer, e.g. (Zhang et al., 2018c; Hoffman et al., 2018; Chang et al., 2019). It has been shown that the combination of the two strategies is superior to using only a single approach. However, existing hybrid methods are often very complex in terms of the number of loss terms and hyper-parameters, which makes them difficult to tune. On the other hand, to the best of the author's knowledge, there is no work that tries to combine another method for UDA with adaptive batch normalisation (Li et al., 2018). In this thesis, such a combination is investigated, assuming that the benefits of combining different adaptation strategies can be achieved without making the training scheme considerably more complex. In particular, the proposed method based on appearance adaptation is combined with adaptive batch normalisation to evaluate if this leads to a higher performance of the classifier after adaptation.

Improving Semantic Consistency: In the literature, there are several works that aim to achieve semantic consistency in adaptation scenarios with very different label distributions. However, such methods (Tasar et al., 2020a; Soto et al., 2021; Yang and Soatto, 2020; Tasar et al., 2020b) regularise the adaptation network by introducing loss terms or constraints which usually cannot achieve semantic consistency without restricting the appearance adaptation too much. This thesis proposes joint training of the source-to-target adaptation network and the classifier to solve this problem at least to some extent, because the classification loss can be used to constrain the adaptation network. However, it is also assumed that regularizing the adaptation network alone does not fully solve the problem of semantically inconsistent appearance adaptation. Therefore, unlike in existing works (Benjdira et al., 2019; Yang and Soatto, 2020; Tasar et al., 2020b; Soto et al., 2021), the actual reasons resulting in inconsistent adaptations are addressed. In particular, this thesis presents two methods that aim to prevent the discriminator from learning patterns that result in semantically inconsistent appearance adaptation instead of regularizing the appearance adaptation network.

The first variant is to restrict the variability of the predictions of the discriminator. To the best of the author's knowledge, the idea of limiting the variability of discriminator outputs to achieve semantic consistency has not been proposed in the literature except for his own publications. The second variant proposed is to train an auxiliary generator to produce images that are fed to the discriminator together with the adapted images from D^S. The idea guiding this strategy is that the images produced by the auxiliary generator can compensate for differences in the underlying distributions of label maps of the two domains and, thus, the appearance adaptation network will not adapt the images in a semantically inconsistent way. After an exhaustive literature research,

no work was found in which image generation from noise is used to achieve semantically consistent adaptations.

Parameter Selection: Another research gap resulting from the literature research is related to the parameter selection criterion. In particular, several works have shown that the number of training iterations during deep UDA is a very important parameter that can strongly affect the performance of the domain adaptation (Gritzner and Ostermann, 2020; Wittich, 2020). Simultaneously, as discussed in Section 3.5, this topic is hardly addressed in literature and, to the author's best knowledge, there is no work that proposes a method to solve this problem. Furthermore, many works (Murez et al., 2018a; Liu et al., 2020; Hoffman et al., 2018; Chen et al., 2019b; Yang et al., 2020b; Zhang et al., 2021b) do not tell for how long they train during deep UDA, whereas choosing the number of training epochs is considered very problematic when trying to transfer such methods to new applications.

Addressing this research gap, in this thesis, a method for parameter selection in deep UDA is proposed. In particular, it is suggested to consider the average entropy in the target domain as a metric to assess the performance in the target domain and to perform the parameter selection based on this metric.

3.6.2 Comparison to Most Similar Works

To further underline the contributions of this thesis, the three works from the literature which are considered to be most similar to the method proposed in this thesis are discussed in detail.

The first similar work is Murez et al. (2018a). The authors propose an hybrid approach for UDA addressing the adaptation from synthetic images to real images for the application of street scene classification. The approach combines source-to-target appearance adaptation and adversarial representation transfer. Similar to the approach proposed in this thesis, the authors train the classifier and the appearance adaptation network jointly using the classification loss of adapted source images as supervision for both, the classifier and the respective appearance adaptation network. However, they also apply a cycle consistency constraint based on the bi-directional CycleGAN architecture and an identity constrained similar to (Soto et al., 2020). The approach presented in this work only performs a source-to-target adaptation, which reduces the number of hyper-parameters to tune and also reduces the memory footprint as less networks are required. Also, instead of achieving semantic consistency solely by constraining the appearance adaptation network, in this work the discriminator is regularized, too, suggesting that this is necessary to achieve semantic consistency in difficult adaptation scenarios, i.e. with large differences in the label distributions.

A source-to-target appearance adaptation that does not build upon CycleGAN is introduced in (Choi et al., 2019). The authors train the appearance adaptation network jointly with the classifier, which follows the same principle as the approach presented in this thesis. Choi et al. (2019) achieve semantic consistency by a semantic constraint forcing the appearance adaptation network to produce images that are classified correctly. This strategy is also used in this thesis, but Choi et al. (2019) use a pre-trained network to apply the semantic constraint, whereas in this

thesis the actual classifier is used to perform such a regularization. It is assumed that using the actual classifier instead of a classifier with fixed parameters is superior to using a classifier with fixed parameters that was trained in the source domain, because it can adapt to the target domain and, thus, regularise the appearance adaptation network in a more meaningful way.

Lastly, the approach by Chen et al. (2019c) is also considered to be very similar to the one proposed in this thesis, because it also performs joint training of a single source-to target adaptation network and the classifier. Chen et al. (2019c) also deal with street scene classification and assume depth information to be available for each image. The authors perform the update of the parameters of the classifier by training on the source images after having them adapted to the target domain for both, pixel-wise classification but also for the auxiliary task of depth estimation. In addition, they apply a sort of adversarial representation transfer using the predictions of the classifier (i.e. the classification result and the estimated depth maps) for adapted images from $D^S$and for the original images from D^T. The respective objectives are used to train all networks jointly. Conceptually, this is very similar to the approach presented in this thesis. However, unlike in (Chen et al., 2019c), neither adversarial representation transfer, nor auxiliary tasks are used in this thesis. Instead, for more challenging adaptation scenarios, new regularisation strategies for the domain discriminator are introduced.

It is noted that none of these methods addresses the problem of parameter selection. Murez et al. (2018a) do not tell at all for how many iterations they train the networks during UDA. Choi et al. (2019) provide adaptation results after 10K and 56K training iterations without giving any argument for the selection of these numbers. Chen et al. (2019c) train the networks for 10 epochs in the adaptation phase, also without providing any reason for the selection of this number. Thus, the fact that in this thesis a method for parameter selection is proposed is another difference to these methods.

4 Methodology

In this chapter, a new approach for unsupervised domain adaptation (UDA) is presented. The setting of UDA follows the definition given in Section 2.6. The prerequisites for the proposed method and the underlying assumptions about the data in the domains are presented in Section 4.1. Section 4.2 provides an overview of the proposed method for UDA. Section 4.3 presents the architectures of all neural networks used. The strategies for training the networks used for UDA is presented in Section 4.4. The novel approaches to regularise the discriminator are presented in Section 4.5, and the proposed parameter selection technique in Section 4.6. Section 4.7 introduces the second, optional adaptation step, which is based on a method from the literature. Section 4.8 describes how the UDA method deals with different spatial resolutions in the source and target domains.

4.1 Prerequisites and Assumptions

The overall scenario addressed in this work corresponds to the setting of homogeneous deep UDA as described in Section 2.6. The addressed task is the pixel-wise classification of images using a FCN as classifier. In principle, the method is agnostic to the application, but the method is described and evaluated on the basis on the two RS applications: land cover classification and bi-temporal deforestation detection (cf. Section 2.1). Depending on the application, the input images to be classified by the FCN can contain different information. To avoid confusion, the variable n_C is introduced, which describes the number of channels of the input image to the classifier, while d denotes the number of channels in the original multispectral images (MSIs). For the application of land cover classification, the input consists of a single georeferenced MSI ($n_C = d$) or the composite of georeferenced MSIs and a rasterised normalised digital surface models (nDSM) ($n_C = d + 1$) which either comes from a photogrammetric 3D reconstruction of the surface or from LiDAR measurements. Thus, if height information is available, the first d channels in the input data to be classified correspond to the orthorectified multi-spectral image and the last channel contains the metric height information for each pixel in the form of a nDSM, which contains the height above terrain for every pixel. For the application of bi-temporal deforestation detection based on satellite imagery, the input consists of a composite image that contains two georeferenced MSIs, thus $n_C = 2 \cdot d$. Here, the first d channels correspond to the earlier MSI and the last d channels correspond to the later MSI. Note that both MSIs have to show the exact same region.

In principle, there is no restriction regarding the number of channels, the spectral resolution or the spatial resolution of the images. However, the spatial resolution should fit the size of the objects which are to be classified.

In this thesis, the variables h_j and w_j are used to describe height and width, respectively, of the j^{th} image in a set of images. However, as it is common practice, the FCN is trained on square patches with a side length p that are classified by the FCN (cf. Section 2.4). During inference, a sliding window approach is used to classify images with an arbitrary size, which will be described in detail in Section 5.2. Based on this notation, the common input feature space of both domains is $\mathcal{X} = \mathbb{R}^{p \times p \times n_C}$. The common output space of both domains is $\mathcal{Y} = \mathbb{L}^{p \times p}$, where $\mathbb{L} = \{L_1, \ldots, L_{n_L}\}$ is the pre-defined class structure with n_L classes.

It is again pointed out that the focus on the work is on the homogenous setting of UDA. For example, an adaptation scenario in which the image channels in D^S correspond to the bands red, green and blue, and those in D^T correspond to the bands infrared, red and green is not considered to be homogenous and is therefore not primarily addressed by the method proposed in this thesis. Instead the images from both domains are assumed to come from comparable sensors and the respective MSI to cover similar spectral bands.

Regarding the availability of data, it is assumed that in the source domain D^S a training set $T^S = \{X_j^S, Y_j^S\}_{j=1}^{n_T}$ of n_T data samples X_j^S and corresponding reference label maps Y_j^S is available. In the target domain D^T, only the set $U^T = \{X_k^T\}_{k=1}^{n_U}$ of n_U unlabelled images is available. The resolution of the imagery from both domains is assumed to be known in the addressed applications, in which the images to be classified are georectified. Unlike many existing UDA methods, the proposed method does not assume the label distributions to be similar in the source and target domains. However, following (Tuia et al., 2016), it is assumed that the knowledge of the classifier trained in D^S is sufficient, although not perfect when applied to D^T.

4.2 Adaptation Overview

In this thesis, a new method for deep UDA is proposed, aiming to adapt a classifier $\mathcal{C}$ from a source domain D^S to a target domain D^T. The base variant of the method is based on appearance adaptation using a new joint training scheme of the classifier and an appearance adaptation network $\mathcal{A}$. In a further variant, this approach is extended by *adaptive batch normalization* (ABN), an existing method for representation transfer from the literature (Li et al., 2018), which is applied as a second adaptation step.

It is assumed that combining the two methods can result in an improved performance of the classifier after adaptation because the methods follow different strategies and address different sets of parameters. In particular, there are two sets of parameters in the classifier. The first set $\Theta_{C,S}$ consists of those parameters that are updated by minimising a loss function. The second set $\Theta_{C,B}$ consists of the running averages in the 2D batch normalisation layers that approximate the batch statistics of the activation maps (cf. Section 2.3.3). The proposed method for appearance adaptation aims at adapting the parameters $\Theta_{C,S}$ of the classifier to perform well in the target domain. In principle, the parameters $\Theta_{C,B}$ could be updated in this adaptation step as well using the adapted images. However, updating these parameters using ABN may result in a better performance, because ABN directly uses the images in the target domain.

Figure 4.1 shows an overview of the processing steps. Note that the figure uses exemplary images and references from the application of land cover classification with aerial imagery, but the processing steps would be equivalent for other applications. The first step is to pre-scale the data in the source domain to the resolution of the target domain (cf. Section 4.8). This step is optional, because it is only required if the data in the two domains have a different spatial resolution. The second step is the training in D^S, which is referred to as *source training* (cf. Section 4.4.1). In this step, a supervised loss is minimised for the training data in D^S. After source training, the new approach for appearance adaptation (cf. Section 4.4.2) is used to adapt the classifier to the target domain, which corresponds to step three in Figure 4.1. In the extended version of the method, another adaptation step is performed, namely the adaptive batch normalisation. This step is skipped in the basic variant. Finally, the adapted classifier is used to predict the probabilistic class scores for the images from the target domain.

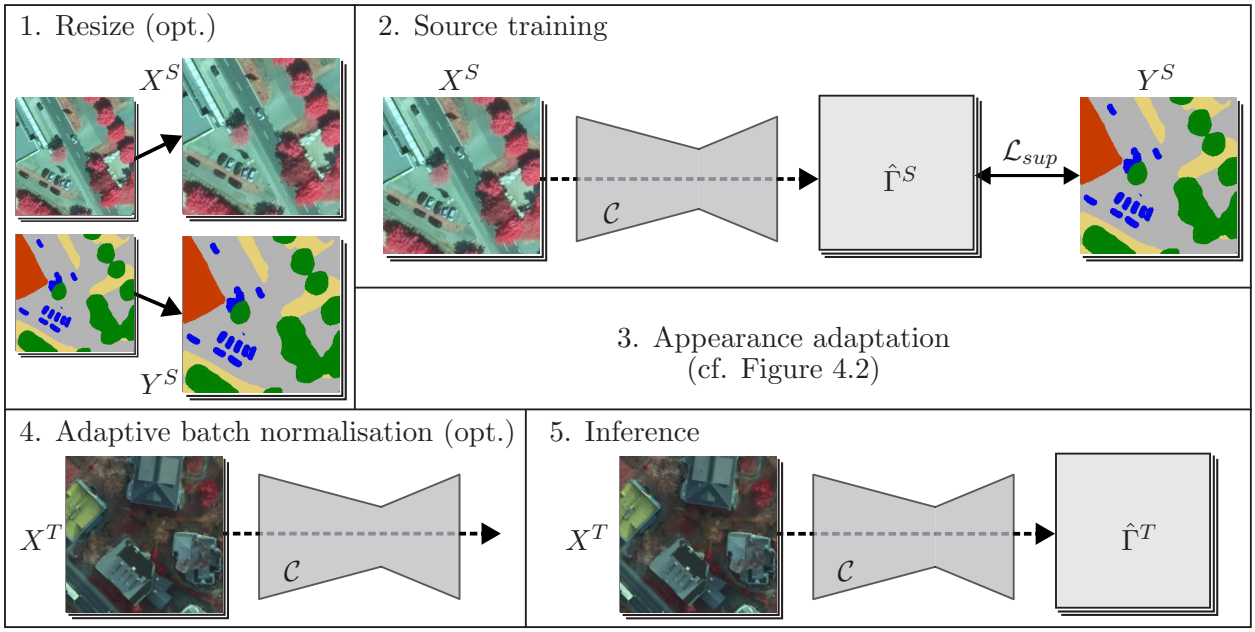

Figure 4.1: Overview of the processing steps of the proposed method. X^S and Y^S denote the images and reference label maps in the source domain and X^T are the images in the target domain. In the first step, the data from D^S are pre-scaled to match the resolution from D^T. Then, in the source training, the classifier $\mathcal{C}$ predicts the maps of probabilistic class scores $\hat{\Gamma}^S$ for X^S and the supervised loss $\mathcal{L}_{sup}$ is minimised. After the two steps for domain adaptation, the classifier $\mathcal{C}$ is used to predict the maps of probabilistic class scores $\hat{\Gamma}^T$ for X^T based on which the final class predictions are computed. opt.: optional processing step.

The main contribution of this thesis is related to the new method for UDA based on appearance adaptation, i.e. to step three in Figure 4.1. An overview of the concept of the basic variant of joint training for appearance adaptation, including the main loss terms, is shown in Figure 4.2. As in many UDA methods based on appearance adaptation, the core idea is to substitute missing label information in D^T by labelled images from D^S that were adapted such that they have an appearance similar to images from D^T. The adaptation of a source image X^S to its adapted version X^{ST} is achieved by an appearance adaptation network $\mathcal{A}$. The classification network $\mathcal{C}$ should be trained in a supervised way so that it performs well for images X^T from D^T; due to the lack of

labelled samples in this domain, the adapted source images X^{ST} with the corresponding label maps Y^S are the main source of supervision available to achieve this goal.

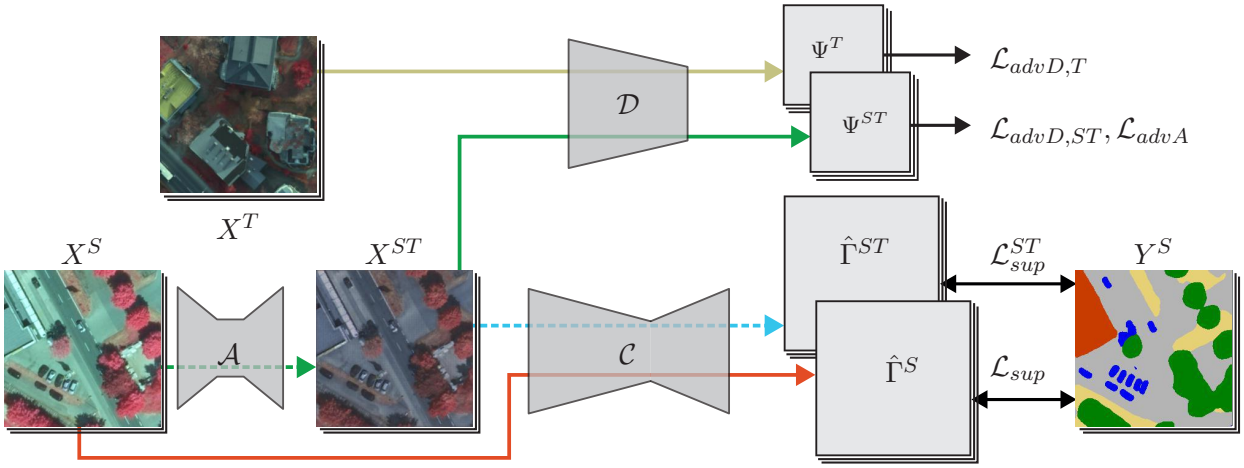

Figure 4.2: Illustration of the concept of joint training for deep UDA (step 3 in Figure 4.1); the coloured arrows indicate the processing flow in the UDA phase and the black arrows correspond to loss terms. In each training iteration in UDA, the appearance adaptation network $\mathcal{A}$ adapts the labelled images X^S from D^S such that they look like images from D^T (green dotted line). The adapted images X^{ST} and the original ones X^S are processed by the classification network $\mathcal{C}$ (red and dotted blue arrows), resulting in the maps of class probabilities $\hat{\Gamma}^S$ and $\hat{\Gamma}^{ST}$, respectively, which are compared to the reference label maps Y^S to determine the loss terms $\mathcal{L}_{sup}$ and $\mathcal{L}_{sup}^{ST}$. Both are minimized in the training of $\mathcal{A}$ and $\mathcal{C}$, but only $\mathcal{L}_{sup}^{ST}$ affects the parameters of $\mathcal{A}$. Within the same training iteration, X^{ST} and the images X^T from D^T are also processed by $\mathcal{D}$, which delivers probability maps Ψ^{ST} and Ψ^T for the corresponding images to belong to D^T (green and yellow arrows, respectively). These maps are considered in the adversarial loss terms $\mathcal{L}_{advA}$, $\mathcal{L}_{advD,T}$ and $\mathcal{L}_{advD,ST}$. While the latter two are used to train $\mathcal{D}$, the first term is considered in the parameter update of $\mathcal{A}$.

To make images that were adapted by $\mathcal{A}$ look like images from the target domain, adversarial training of $\mathcal{A}$ and a domain discriminator $\mathcal{D}$ is performed. Many approaches for appearance adaptation, e.g. (Soto et al., 2020; Benjdira et al., 2019; Gritzner and Ostermann, 2020), start by adversarial training to learn how to adapt images from D^S to D^T and use the adapted images along with the known labels for training the classifier $\mathcal{C}$ in a separate step. However, for the appearance adaptation to be successful, the adapted images should not only look like images from the target domain, but the transformation also has to be semantically consistent in the way defined in Section 2.5, which is difficult to achieve in a two-step approach. Thus, in the method presented in this thesis, $\mathcal{A}$ and $\mathcal{C}$ are not trained separately, but jointly. The core idea of this approach is that $\mathcal{A}$ learns to adapt input images from D^S to D^T such that they look like coming from D^T, which is achieved by minimizing an adversarial loss $\mathcal{L}_{advA}$, while at the same time they are classified correctly by $\mathcal{C}$, which is achieved by minimizing a supervised loss $\mathcal{L}_{sup}^{ST}$. At the same time, the classifier $\mathcal{C}$ is trained so that it classifies transformed samples X_j^{ST} correctly, which is also achieved by minimizing $\mathcal{L}_{sup}^{ST}$. This is required to achieve the main goal of UDA, i.e. a good performance of $\mathcal{C}$

in D^T. By simultaneously minimizing another supervised loss $\mathcal{L}_{sup}$, $\mathcal{C}$ also learns to classify images X^S from the source domain, which acts as a kind of regularization to avoid a drift of the parameters (cf. Section 4.4.2). Besides this theoretical consideration, it was also empirically shown in (Soto et al., 2020) that training on X^S and X^{ST} is superior to training only using X^{ST} with respect to the performance in the target domain after UDA. As usual in adversarial training, the discriminator $\mathcal{D}$ is trained to make the adaptation task difficult for $\mathcal{A}$, which is achieved by minimizing the two adversarial loss terms $\mathcal{L}_{advD,T}$ and $\mathcal{L}_{advD,ST}$. Note that besides the basic variant of joint training, two methods are introduced in this thesis that aim to improve the semantic consistency of the appearance adaptation (cf. Section 4.5).

4.3 Network Architecture

The overall network architecture consists of the three sub-networks (cf. Figure 4.2): the classification network $\mathcal{C}$, the appearance adaptation network $\mathcal{A}$ and the domain discriminator $\mathcal{D}$, with corresponding sets Θ_C, Θ_A, and Θ_D of trainable parameters. The parameters Θ_C of $\mathcal{C}$ consist of two subsets $\Theta_{C,S}$ and $\Theta_{C,B}$. $\Theta_{C,B}$ contains the running averages from all batch normalisation layers (cf. Section 2.3.3) and $\Theta_{C,S}$ are the remaining trainable parameters of the network (cf. Section 4.2).

4.3.1 Classification Network $\mathcal{C}$

The pixel-wise classification of images is performed by a FCN $\mathcal{C}$. The input to $\mathcal{C}$ corresponds to a square patch with side length p and n_C channels (cf. Section 4.1), and the output corresponds to the pixel-wise probabilistic class scores. The probabilistic class scores for each pixel are obtained by normalising the unnormalised class scores using the softmax function (cf. Equation 2.7). The probabilistic class scores for every pixel are arranged in maps $\hat{\Gamma}^S$ for images from D^S and $\hat{\Gamma}^{ST}$ for adapted images X^{ST}. This corresponds to the red and blue paths in Figure 4.2, respectively. During inference, the maps of probabilistic class scores $\hat{\Gamma}^T$ are predicted for the images X^T from D^T (cf. Figure 4.1). The actual class predictions are obtained by selecting the class with the highest probability for each pixel. In this work, a U-Net-like architecture (Ronneberger et al., 2015) is used, having an Xception backbone (Chollet, 2017) pre-trained on ImageNet (Deng et al., 2009). In preliminary experiments, this architecture in combination with initialisation of the encoder by pre-training on ImageNet was compared to a residual network with completely random initialization similar to the one used in (Wittich, 2020). It was observed that a comparable performance can be achieved, but the pre-training results in a noticeable reduction of the training time.

Figure 4.3 shows an overview of the classification network and Table 4.1 provides an overview of all layers. The encoder of the classification network (layers 1-17 in Table 4.1 and Figure 4.3) corresponds to the layers 1-17 of the Xception network that was explained in detail in Section 2.3.6.2. In the decoder of the network, nearest neighbour interpolation is used for upsampling. All convolutions in the decoder use 3×3 kernels and zero padding with $1\,px$. The only exception is the very last convolution in layer 33 that uses 1×1 kernels and no padding. The layers that correspond to the Xception network are pre-trained on the ImageNet dataset. The implementation of the Xception

U-Net and the pre-trained weights are taken from the *Segmentation Models Pytorch* repository[1] (Iakubovskii, 2019).

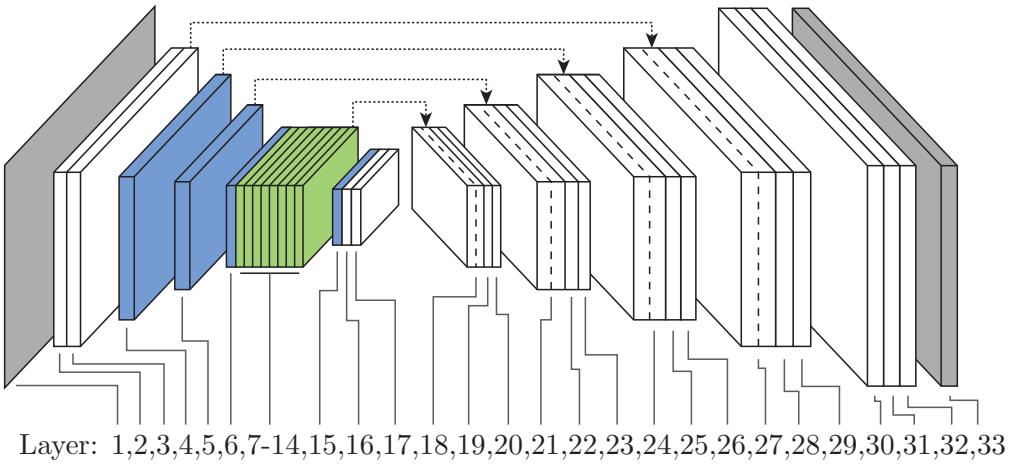

Figure 4.3: Illustration of the Xception U-Net architecture. The layer numbers correspond to those in Table 2.3. Blue and green layers correspond to Xception blocks of type A and B, respectively. The dotted lines represent skip connections.

	Layer(s)	Layer type	h, w	Depth	Simpl.
Encoder	1	Input layer	p	n_C	y
	2	StrConv, BN, ReLU	$p/2$	32	y
	3	Conv, BN, ReLU	$p/2$	64	y
	4	Xception block A	$p/4$	128	y
	5	Xception block A	$p/8$	256	y
	6	Xception block A	$p/16$	728	y
	7-14	8× Xception block B	$p/16$	728	y
	15	Xception block A	$p/32$	1024	n
	16	SepConv, BN, ReLU	$p/32$	1536	n
	17	SepConv, BN, ReLU	$p/32$	2048	n
Decoder	18	Upsample, Concat(14)	$p/16$	2776	n
	19, 20	2×{Conv, BN, ReLU}	$p/16$	256	n
	21	Upsample, Concat(5)	$p/8$	984	y
	22, 23	2×{Conv, BN, ReLU}	$p/8$	128	y
	24	Upsample, Concat(4)	$p/4$	256	y
	25, 26	2×{Conv, BN, ReLU}	$p/4$	64	y
	27	Upsample, Concat(3)	$p/2$	128	y
	28,29	2×{Conv, BN, ReLU}	$p/2$	32	y
	30	Upsample	p	32	y
	31,32	2×{Conv, BN, ReLU}	p	16	y
	33	Conv, Softmax	p	n_L	y

Table 4.1: Layers of the architecture of $\mathcal{C}$. Conv: Convolutional layer. StrConv: Strided convolutional layer. SepConv: Separable convolutional layer. BN: 2D batch normalisation; ReLU: Rectified linear unit. Concat(L_X): Depth-wise concatenation of the output of layer L_X and the current layer. h, w, depth: Output dimensions. p is the predefined patch size. Xception blocks are described in Section 2.3.6.2. The column *Simpl.* denotes whether this layer or group of layers is used in the simplified variant (y) or not (n).

[1]URL: https://github.com/qubvel/segmentation_models.pytorch (last accessed on 09/12/2022)

Because the backbone is pre-trained on three-channel RGB-images, the first layer of the network cannot be used if the number n_C of input channels is different from three. In that case, the first layer is replaced by a convolution layer with n_C input channels, the parameters of which are initialized randomly. The parameters of the decoder are also initialized randomly; all random initializations are based on (He et al., 2015). Overall, this network has about 28.8 M parameters. This rather large network was chosen because it has to learn to classify images from D^S and from D^T (cf. Section 4.4), which is assumed to be a more complex task than classifying only images from one domain, due to the higher variability of the data. Using a larger network reduces the risk that the learning capacity of $\mathcal{C}$ becomes a limiting factor of the method. The network also has a very large theoretical receptive field of $1179\,px$. Practically, if the the patch size p is smaller than $590\,px$, every pixel in the input image can potentially contribute to the predicted label for every pixel in the output.

Note that in this architecture the number of layers and parameters can be reduced by removing the last layers of the encoder and the first layers of the decoder. Whether or not this is advantageous depends on the particular application. In this work, a smaller variant of this network is used in the context of bi-temporal deforestation. Particularly, in that variant, the layers 15-20 are omitted, resulting in a smaller network with $15.5M$ parameters and a theoretical receptive field of $907\,px$. This network consists of the layers that are marked with y in the column *Simpl.* in Table 4.1.

4.3.2 Appearance Adaptation Network

The appearance adaptation network $\mathcal{A}$ takes a square image patch with side length p with n_C channels extracted from an image X^S from the source domain as input and delivers an adapted image patch X^{ST}. This corresponds to the dotted green line in Figure 4.2. For this task a residual FCN with about 5 M parameters is used that is a simplified version of the one used in (Wittich, 2020). Table 4.2 lists all layers of the network.

Layer(s)	Layer type	h, w	Depth
1	Input layer	p	n_C
2	StrConv, ReLU	$p/4$	256
3-18	15× Residual block	$p/4$	256
19	T-Conv, ReLU	$p/2$	128
20	T-Conv	p	n_C

Table 4.2: Residual network $\mathcal{A}$ for appearance adaptation. T-Conv: Transposed convolution. For other abbreviations, cf. Table 4.1. The structure of the residual blocks used in this architecture is given in Table 4.3.

An initial 6×6 strided convolution with a stride of four pixels and zero padding of $1\,px$ downsamples the image patch spatially by a factor of 4. It is followed by 15 residual blocks at the reduced scale. Each of them consists of two subsequent 3×3 convolutions with 64 and 256 filters, respectively, replicate-padding with $1\,px$ and ReLU activation. The result of each residual block is added to its input. The layers of a residual block are defined in Table 4.3.

After the residual blocks, two transposed 4×4 convolutions, each performing an upsampling by a factor of two, are used to enable predictions at the original spatial resolution of the input,

corresponding to the spatial extent $p \times p$ of the patch. Commonly, non-linearities with a bounded range of values such as the hyperbolic tangent function are applied to the output of the final layer of networks for image generation (Goodfellow et al., 2014). However, as in some cases the last channel of the output of $\mathcal{A}$ corresponds to the adapted nDSM, the values are not restricted to a fixed range. Consequently, no activation function is applied to the output of the last convolutional layer in the network $\mathcal{A}$.

Layer	Layer type	h, w	Depth
1	Input layer	$s^{(in)}$	256
2	Conv, ReLU	$s^{(in)}$	64
3	Conv, ReLU	$s^{(in)}$	256
5	Add(1,3)	$s^{(in)}$	256

Table 4.3: Layers of a residual block. Conv: Convolutional layer. $s^{(in)}$: Height and width of the input to the block. Remaining abbreviations and symbols as in the caption of Table 4.1. Add(1,3): the outputs of layer 1 and layer 3 are element-wise added.

A residual FCN is chosen for the appearance adaptation because the optimal solution for this task should not deviate too much from an identity mapping. For example, objects that have the same appearance in both domains should not be changed at all in the adaptation. Following (He et al., 2016), it is assumed that residual networks are well suited to learn such a solution, an assumption that was also supported by preliminary experiments. This network uses replicate-padding instead of zero-padding in the residual blocks. In preliminary experiments, this was found to reduce artefacts in the border regions of the adapted images. The network has a theoretical receptive field of $250 \times 250\,px$, which is considered large enough to consider enough context to perform an adequate image adaptation for the addressed resolutions. In particular, when classifying aerial images with a ground sampling distance (GSD) of $20\,cm$, this receptive fields corresponds to an area of $50 \times 50\,m^2$, which is in most cases larger than objects of interest such as buildings or trees. In the addressed application of bi-temporal deforestation detection, for a GSD of the satellite images of $20\,m$, the size of the receptive field is $5000 \times 5000\,m^2$, which again is in most cases larger than the objects which are to be classified in that application, namely deforestation areas.

4.3.3 Domain Discriminator

The domain discriminator network $\mathcal{D}$ is required for the training of the appearance adaptation network in an adversarial way (cf. Section 4.4.2). It takes either an adapted image X^{ST} from D^S or a target domain image X^T as input and predicts from which domain the image originates. This corresponds to the solid green and yellow paths in Figure 4.2. Consequently, the discriminator performs a binary classification with the class labels $y = L_T$ that corresponds to the case in which the input image comes from D^T and $y = L_{ST}$, corresponding to the case in which the input image is an image from D^S that was adapted by the appearance adaptation network. Instead of predicting a single class score per image patch (Goodfellow et al., 2014), the network predicts a map of probabilistic class scores (cf. Section 2.5). In the literature, this concept has been shown to achieve better results for adapting the appearance of images, for instance by Isola et al. (2017), whose discriminator architecture is adapted here.

The predicted probability maps are denoted by Ψ^{ST} and Ψ^{T} for the input images X^{ST} and X^{T}, respectively. Each value ψ_i^{ST} in Ψ^{ST} and ψ_i^{T} in Ψ^{T} correspond to the probability for the corresponding support window in the input image to come from the target domain (cf. Section 2.5). Thus, $\psi_i^{T} = P(y_{W(i)} = L_T|X^{T})$ and $\psi_i^{ST} = P(y_{W(i)} = L_T|X^{ST})$, where $y_{W(i)}$ refers to the label of the support window $W(i)$ that corresponds to the i-th prediction in the probability map predicted by the discriminator. Note that the i-th prediction in the probability map corresponds to the i-th pixel in the output of $\mathcal{D}$, but it does not correspond to a specific pixel in the input.

The discriminator network, which has about 2.8 M parameters, is described in Table 4.4. This architecture follows the architecture proposed by Isola et al. (2017) which is frequently used for appearance adaptation and in the context of appearance based deep UDA, e.g. in (Murez et al., 2018b; Tasar et al., 2020b; Zhao et al., 2023). It consists of five convolutional layers, each using a 4×4 kernel. The first three convolutions are strided convolutions with a stride of $2\,px$; the final two layers do not use striding. Using this sequence of convolutions the support window of each pixel in the output of $\mathcal{D}$ is a $70 \times 70\,px$ area in the original input image. Isola et al. (2017) propose to use leaky ReLU as activation function for all layers but the last one. The output of the last layer is normalised by the sigmoid function (cf. Equation 2.4) such that the output values can be interpreted as probabilities. In this thesis, deviating from (Isola et al., 2017), the 2D batch normalization layers are replaced by a spectral normalization of the weights of the kernel matrices, i.e. such that after normalisation the largest singular value of the weights of each kernel is one, as proposed in (Miyato et al., 2018). In preliminary experiments this was found to lead to a more realistic appearance of the adapted images in a visual evaluation.

Layer	Layer type	h, w	Depth
1	Input layer	254	d
2	StrConv, LReLU	126	64
3	SN-StrConv, LReLU	62	128
4	SN-StrConv, LReLU	30	256
5	SN-Conv, LReLU	27	512
6	SN-Conv, Sigmoid	24	1

Table 4.4: Layers of the discriminator network $\mathcal{D}$. SN-Conv: Convolution with spectral normalization of the weights. SN-StrConv: Strided convolution with spectral normalization of the weights. LReLU: Leaky ReLU with negative slope of $\theta_L = 0.1$ (as in (Wittich, 2020)). For other abbreviations, cf. Table 4.1.

4.4 Training

To determine the parameters of all networks according to the proposed joint training scheme, a three-stage training strategy which consists of the source training and the subsequent UDA is proposed. In the first stage, source training is performed. Here, only the parameters Θ_C of the classification network $\mathcal{C}$ are determined by conventional supervised training using the labelled source domain dataset T^S (cf. Section 4.4.1), resulting in the parameter set $\hat{\Theta}_C^{(src)}$. In the second stage, described in Section 4.4.2, the proposed method for appearance adaptation is carried out using the datasets T^S and U^T. In this process, the parameter set $\hat{\Theta}_C^{(src)}$ is used as initialization for $\mathcal{C}$,

and the parameters of the other networks are randomly initialized according to (He et al., 2015). The second stage could also be carried out starting from a random initialization. However, in preliminary experiments it was observed that performing the UDA from a random initialization increased the training time by a factor of approximately 2 and in a few cases also had a negative effect of the classification performance after UDA. The result of the joint appearance adaptation is the parameter set $\hat{\Theta}_C^{(jaa)} = (\hat{\Theta}_{C,S}^{(jaa)}, \hat{\Theta}_{C,B}^{(jaa)})$ which can be decomposed into the set $\hat{\Theta}_{C,B}^{(jaa)}$ that contains the running averages from all 2D batch normalisation layers after appearance adaptation, and the set $\hat{\Theta}_{C,S}^{(jaa)}$ that are the remaining trainable parameters of the network.

In the last stage, described in Section 4.7, an optional further unsupervised adaptation step is performed that is based on *adaptive batch normalization* (ABN), a method proposed in the literature (Li et al., 2018). ABN starts from the parameter set $\hat{\Theta}_C^{(jaa)}$ and updates only the parameter set $\hat{\Theta}_{C,B}$. This adaptation step results in the parameter set $\hat{\Theta}_C^{(abn)} = (\hat{\Theta}_{C,S}^{(jaa)}, \hat{\Theta}_{C,B}^{(abn)})$, where $\hat{\Theta}_{C,B}^{(abn)}$ are the updated running averages from all 2D batch normalisation layers.

In the base variant, the parameters $\hat{\Theta}_C^{(jaa)}$ are used for the final evaluation and in the extended variant the parameter set $\hat{\Theta}_C^{(abn)}$ is used.

4.4.1 Supervised Source Training

During source training, the parameters $\Theta_C = (\Theta_{C,S}, \Theta_{C,B})$ for the classification network $\mathcal{C}$ are determined using the training data set T^S. The subset $\Theta_{C,B}$, i.e. the running averages in the 2D batch normalisation layers are updated according to Equations 2.30 and 2.31. The remaining parameters $\Theta_{C,S}$ are initialised as described in Section 4.3.1 and iteratively updated using mini-batch stochastic gradient descent with momentum (cf. Section 2.2.2.1), minimising the loss

$$\mathcal{L}_C^{(src)}(\Theta_{C,S}, T^S) = \mathcal{L}_{sup}(\Theta_{C,S}, T^S) + \omega_{L2} \cdot \mathcal{L}_{L2}(\Theta_{C,S}). \tag{4.1}$$

The first loss term $\mathcal{L}_{sup}$ is a supervised loss that measures the discrepancy between the predicted labels and the reference (cf. Section 2.2.2.1). The second loss term corresponds to a L2-regularisation of the parameters $\Theta_{C,S}$ according to Equation 2.19, which is weighted by ω_{L2}.

Commonly, the multi-class cross-entropy loss (cf. Section 2.2.2) is used for a multi-class classification problem. However, this loss can be suboptimal if the class distribution of the training dataset is imbalanced, as it is often the case in RS applications. In such a case, the cross-entropy loss is dominated by the frequent classes, which is the reason why the prediction quality of under-represented classes may be not satisfactory after training.

In this thesis, the adaptive cross entropy, proposed in (Wittich and Rottensteiner, 2021), is used to mitigate the problems due to an imbalanced label distribution. Similarly to the focal loss (Lin et al., 2017; Yang et al., 2019), it also adopts the idea of assigning larger weights to more difficult samples. However, in contrast to the focal loss, the training procedure should not focus on individual pixels whose class labels are difficult to predict, but it should focus on classes which are

predicted with low quality. Thus, one weight per class is determined (and not per pixel as in (Lin et al., 2017)), which depends on the current global prediction quality for this class.

After initialization, training starts with one training epoch in which the standard multi-class cross-entropy loss (cf. Equation 2.36) is minimised. Note that in this thesis, the term epoch is defined as consisting of n_{iter} training iterations. After the first training epoch, training images of a mini-batch are classified using the current state of the classifier and the results are compared to the reference to determine class-wise quality metrics. In particular, the F_1 score is used. The F_1 score $F_{1,k}$ is a class-specific performance indicator for the k-th class L_k:

$$F_{1,k} = \frac{2 \cdot TP_k}{2 \cdot TP_k + FP_k + FN_k}. \tag{4.2}$$

In Equation 4.2, TP_k, FP_k and FN_k are the numbers of true positives, false positives and false negatives, respectively, for class L_k. The F1-scores are used to determine the class-wise weights

$$w_k = (1 - \Delta F_{1,k})^\kappa = (1 - (F_{1,k} - \frac{1}{n_L} \sum_{k^*=1}^{n_L} F_{1,k^*}))^\kappa, \tag{4.3}$$

where $\Delta F_{1,k}$ is the difference between the class-wise F_1 score for the k-th class L_k and the mean F_1 score of all classes, and the hyper-parameter κ modulates the influence of the weight of each class. Starting from the second epoch, the classifier is trained using a weighted cross-entropy loss in which the loss of each pixel is weighted by w_k according to its reference label. After each epoch, the weights are recalculated according to Equations 4.2 and 4.3 and used for the following epoch. For a batch of n_B training images with patch size p, this loss becomes

$$\mathcal{L}_{sup}(\Theta_{C,S}, T^S) = -\frac{1}{n_B \cdot p^2} \sum_{b=1}^{n_B} \sum_{i=1}^{p^2} \sum_{k=1}^{n_L} \gamma_{b,i,k} \cdot log(\hat{\gamma}^S_{b,i,k}) \cdot w_k, \tag{4.4}$$

where $\hat{\gamma}^S_{b,i,k} = P(y_{b,i} = L_k | X^S_b)$ is the predicted probability for the i-th pixel in the b-th image in the batch to belong to class $y_{b,i} = L_k$, w_k is the weight of class L_k according to Equation 4.3 and T^S is the training dataset. Recall that the symbol $\gamma_{b,i,k}$ indicates whether the i-th pixel in the b-th label map belongs to class L_k ($\gamma_{b,i,k} = 1$) or not ($\gamma_{b,i,k} = 0$) (cf. Section 2.2.2). Note that practically, the weights are initialised by ones. By doing so, the loss in Equation 4.4 becomes the regular cross-entropy loss from Equation 2.36, which is used in the first training epoch.

To further prevent the classifier from overfitting, data augmentation is applied. Details will be described in Section 5.4.1. To determine the number of training epochs, the performance of the classifier is evaluated on a validation set after each epoch. Source training is performed for $n_E^{(S)}$ epochs or stopped if the performance on the validation set does not increase for n_{es} epochs, following the strategy of early stopping defined in Section 2.2.3. After training, the parameter set $\hat{\Theta}_C^{(src)}$ that resulted in the best validation performance is used for the further processing steps.

4.4.2 Joint Training for Appearance Adaptation

In the appearance adaptation phase, the parameters of all networks are determined using variants of mini-batch stochastic gradient descent (cf. Section 2.2.2.1). To simplify the notation, a combined

set $\Theta_{A,C} = \{\Theta_A, \Theta_{C,S}\}$ is defined, which consists of the parameters of the adaptation network $\mathcal{A}$ and the classification network $\mathcal{C}$. While the parameters of $\mathcal{C}$ are initialised by those determined in the source training, i.e. $\hat{\Theta}_C^{(src)}$, the parameters of the discriminator are initialized randomly based on (He et al., 2015). As these parameters are to be determined by joint training of the two networks, according to the principles of adversarial training, the gradient of a joint loss function $\mathcal{L}_{A,C}$ with respect to $\Theta_{A,C}$ is first calculated in each iteration. This corresponds to the paths visualized by the red, the blue and the two green arrows in Figure 4.2, based on a batch of n_B image patches from D^S and the corresponding labels. The resulting gradient is used to update the parameter set $\Theta_{A,C}$. Then, within the same training iteration, the gradient of a discriminator loss $\mathcal{L}_D$ with respect to the discriminator parameters Θ_D is calculated and used to update Θ_D. This requires an additional set of n_B unlabelled image patches from D^T, which are processed jointly with the n_B adapted images from D^S that contributed to the update of $\Theta_{A,C}$, and it corresponds to the paths visualized by the yellow and green arrows in Figure 4.2. This work follows the common practice of alternating between updating $\Theta_{A,C}$ and Θ_D (Goodfellow et al., 2014). The two steps are described in detail below. Algorithm 1 summarizes joint appearance adaptation process. The parameter update process (lines 10 and 11) is described in the subsequent sections. Line 5 and lines 15-20 are related to the new parameter selection approach, which will be described in Section 4.6.

4.4.2.1 Joint Update of $\mathcal{A}$ and $\mathcal{C}$

The joint loss $\mathcal{L}_{A,C}$ used to update the parameters of $\mathcal{A}$ and $\mathcal{C}$ in every iteration (line 11 of Algorithm 1) consists of four components:

$$\mathcal{L}_{A,C}(\Theta_{A,C}, \Theta_D, T^S) = \\ \omega_T \cdot \mathcal{L}_{sup}^{ST}(\Theta_{A,C}, T^S) + \omega_A \cdot \mathcal{L}_{advA}(\Theta_A, \Theta_D, T^S) + \mathcal{L}_{sup}(\Theta_{C,S}, T^S) + \omega_{L2} \cdot \mathcal{L}_{L2}(\Theta_{C,S}), \quad (4.5)$$

where ω_T, ω_A and ω_{L2} are weighting factors to control the influence of the corresponding loss terms relative to the influence of the supervised term $\mathcal{L}_{sup}$.

The first term, $\mathcal{L}_{sup}^{ST}$, is related to the main goal of the UDA, namely to achieve a good classification performance on images from the target domain. It is formulated as a supervised classification loss for adapted images similar to Equation 4.4:

$$\mathcal{L}_{sup}^{ST}(\Theta_{A,C}, T^S) = -\frac{1}{n_B \cdot p^2} \sum_{b=1}^{n_B} \sum_{i=1}^{p^2} \sum_{k=1}^{n_L} \gamma_{b,i,k} \cdot log(\hat{\gamma}_{b,i,k}^{ST}) \cdot w_k, \quad (4.6)$$

where $\hat{\gamma}_{b,i,k}^{ST}$ denotes the predicted probabilistic class score for the i-th pixel in the b-th adapted input image from a batch of images from D^S to belong to class $y_{b,i} = L_k$. Thus, $\hat{\gamma}_{b,i,k}^{ST} = P(y_{b,i} = L_k | X_b^{ST})$. The remaining symbols are those already defined in the context of Equation 4.4. Besides adapting $\mathcal{C}$ to the target domain, this loss is also required to guide $\mathcal{A}$ towards performing semantically consistent adaptations, i.e. that images X^{ST} are still correctly classified by $\mathcal{C}$. Note that the weights w_k in Equation 4.6 are those calculated on the basis of samples from D^S.

The second term in Equation 4.6 is the adversarial loss $\mathcal{L}_{advA}$ which only affects $\mathcal{A}$ and realizes the component of adversarial training that influences $\mathcal{A}$. The appearance adaptation network $\mathcal{A}$

Algorithm 1: Joint appearance adaptation $S \rightarrow T$

Function `JointAppearanceAdaptation(`$\hat{\Theta}_C^{(src)}$`,`T^S`,`U^T`,`n_E`,`n_{iter}`,`n_{min}`):`

`//` $\hat{\Theta}_C^{(src)}$`:` `parameters after source training`
`//` T^S`:` `labelled source domain data`
`//` U^T`:` `unlabelled target domain data`
`//` n_E`:` `number of training epochs`
`//` n_{iter}`:` `number of training iterations per epoch`
`//` n_{min}`:` `related to parameter selection`

$\Theta_C \leftarrow \hat{\Theta}_C^{(src)}$ `// start with parameters after source training`
$\Theta_A, \Theta_D \leftarrow$ random initialization
$\hat{\Theta}_C^{(jaa)} \leftarrow \hat{\Theta}_C^{(src)}$ `// initialize optimal parameter sets`
$ent_{min} \leftarrow inf$
$(w_1, w_2, ...) \leftarrow$ initialise class weights with ones
for $e \leftarrow 1$ **to** n_E **do** `//` e`:` `epoch`
  for $i \leftarrow 1$ **to** n_{iter} **do** `//` i`:` `iteration`
    $B^S \leftarrow$ batch from T^S `//` `labelled data from` $\mathcal{D}^S$
    $B^T \leftarrow$ batch from U^T `//` `unlabelled data from` $\mathcal{D}^T$
    update $\Theta_{A,C}$ by minimizing $\mathcal{L}_{A,C}$ using B^S `//` `(eq. 4.5)`
    update running averages $\Theta_{C,B}$ `//` `(eq. 2.30 and 2.31)`
    update Θ_D by minimizing $\mathcal{L}_D$ using B^S and B^T `//` `(eq. 4.8)`
  end
  update each class weight w_l `//` `(eq. 4.3)`
  if $e > n_{min}$ **then**
    `//` `do not evaluate the entropy for the first` n_{min} `epochs`
    $ent^t \leftarrow$ average entropy for images from U^T
    if $ent^t < ent_{min}$ **then**
      $ent_{min} \leftarrow ent^t$
      $\hat{\Theta}_C^{(jaa)} \leftarrow \Theta_C$ `//` `update optimal parameters`
    end
  end
end
return $\hat{\Theta}_C^{(jaa)}$

should learn to adapt images from D^S such that they look like images from D^T by maximising the probabilities ψ_{ST} predicted by $\mathcal{D}$ for the adapted images X^{ST} to be from D^T, which results in the following loss:

$$\mathcal{L}_{advA}(\Theta_A, \Theta_D, T^S) = -\frac{1}{n_B \cdot \tilde{p}^2} \sum_{b=1}^{n_B} \sum_{i=1}^{\tilde{p}^2} log(\psi_{b,i}^{ST}), \tag{4.7}$$

where $\psi_{b,i}^{ST}$ corresponds to the i-th prediction of $\mathcal{D}$ for the b^{th} image in the mini-batch of adapted images, i.e. the predicted probability $\psi_{b,i}^{ST} = P(y_{b,W(i)} = L_T | X^{ST})$ for the corresponding support window $W(i)$ in the input image presented to $\mathcal{D}$ to be from the target domain, which should be large such that $\mathcal{A}$ can learn to fool the discriminator (cf. Section 4.3.3). $\tilde{p}$ represents the height and the width of the discriminator output.

The third term in Equation 4.5, $\mathcal{L}_{sup}(\Theta_{C,S}, T^S)$, is the supervised loss for images from the source domain and is computed according to Equation 4.4. This is a further important component to achieve semantic consistency and to avoid a drifting effect of $\mathcal{A}$ and $\mathcal{C}$. Particularly, if $\mathcal{C}$ were solely trained using the adapted images, it would be possible for the parameters of the appearance adaptation network to converge to a parameter state where $\mathcal{A}$ produces semantically inconsistent results, e.g. images in which after appearance adaptation trees look like buildings in D^T and vice versa. $\mathcal{C}$ could adapt to such a state and still predict the label maps correctly, but it would no longer perform well for real target images. Considering $\mathcal{L}_{sup}$ for source images will constrain the classifier so that it still performs well as a classifier in the source domain, so that this loss acts as a regularization term.

The last term in Equation 4.5, $\mathcal{L}_{L2}(\Theta_C)$, corresponds to a L2-regularisation of the parameters Θ_C as introduced in Section 2.2.3, which is weighted by ω_{L2}.

As stated above, each training iteration starts with a step aiming at minimizing the joint loss $\mathcal{L}_{A,C}$ with respect to Θ_A and $\Theta_{C,S}$ (line 10 of algorithm 1). While $\mathcal{L}_{A,C}$ depends on Θ_D via the adversarial term $\mathcal{L}_{advA}$, these parameters are not updated in this context. Minimizing $\mathcal{L}_{A,C}$ will adapt the adaptation network $\mathcal{A}$ such that it deceives the discriminator $\mathcal{D}$, i.e. such that its output cannot be discriminated from a real target domain image. At the same time the parameters of the classifier $\mathcal{C}$ are updated such that $\mathcal{C}$ performs well for target domain images. The supervised loss $\mathcal{L}_{sup}$ for images from D^S acts as a kind of regularization for $\Theta_{C,S}$.

In one training iteration, a source domain image will contribute to the loss $\mathcal{L}_{A,C}$ twice, namely via $\mathcal{L}_{sup}^{ST}$ (Equation 4.6) and $\mathcal{L}_{sup}$ (Equation 4.4). This has to be considered in the 2D batch normalization layers of $\mathcal{C}$. As pointed out in (Li et al., 2018), the running averages in 2D batch normalization layers capture domain specific feature distributions and can have a major effect on the performance of the final classifier in D^T. In this thesis, it is proposed to address the batch statistics in a subsequent adaptation step, namely by applying adaptive batch normalisation. However, in the basic variant of the proposed method, only the images X^S from D^S are used to update the running averages of these layers. This variant was found to result in a better performance after UDA compared to updating the running averages on adapted images X^{ST} from D^S, but also compared to updating them using both image types X^S and X^{ST}. In the optional subsequent adaptation

step, the running averages are adapted to the target domain, which will be discussed in detail in Section 4.7.

Random Spatial Shifts: As a minor extension, it is proposed to perform a random spatial shifting of the adapted images X^{ST} jointly with the corresponding label maps used to compute $\mathcal{L}_{sup}^{ST}$, before they are fed to the discriminator. This is motivated by the following two considerations.

First, as the output of $\mathcal{A}$ is to be classified by $\mathcal{C}$ and both networks $\mathcal{A}$ and $\mathcal{C}$ are jointly trained to minimize the classification loss $\mathcal{L}_{sup}^{ST}$, the classification task which should be learned by $\mathcal{C}$ could in principle also be learned to some extent by $\mathcal{A}$. Particularly, $\mathcal{A}$ could learn to perform the classification task and to encode the classification result in the adapted images e.g. in terms of superimposed patterns that can easily be recognized by the classifier. In a few cases, such behaviour was observed in preliminary experiments, visible as superimposed high-frequency patterns in the adapted images and leading to a reduced adaptation performance. It is assumed that this behaviour occurs in a setting in which the appearance adaptation leads to a reduction of information in the images. For instance, if the appearance adaptation network has to add dark shadows in order to mimic the style of images from D^T it would be more difficult for $\mathcal{C}$ to perform a correct classification given the adapted images. In order to minimise $\mathcal{L}_{sup}^{ST}$, $\mathcal{A}$ may encode the semantic information by superimposing patterns in X^{ST}. This can prevent the classifier from learning patterns that are relevant for performing a proper classification in D^T. As a countermeasure for this undesirable behaviour, the adapted images are randomly shifted horizontally and vertically by $\pm 1\,px$ with a probability of 50 %, respectively. By performing the random shifting, it is more difficult for the classifier to detect and use the superimposed patterns generated by $\mathcal{A}$. Note that larger shifts would not bring any further benefits because the step size of the first convolutional layer in $\mathcal{C}$ is $2\,px$, which means that shifting the input by $2\,px$ would lead to the same relative alignment of the adapted images and the kernel positions in the first layer of $\mathcal{C}$. However, only shifting the input of $\mathcal{C}$ would break the alignment with the reference label map. This is why the label maps are shifted correspondingly with the images, using the same random values. The shifting operation potentially results in empty border regions which are not considered when computing the overall loss $\mathcal{L}_{sup}^{ST}$. It shall be noted that this proposed counter-measure cannot fully prevent $\mathcal{A}$ from superimposing patterns that encode semantic information in the generated images. However, the proposed shift is computationally inexpensive and assumed to prevent this behaviour at least to some extent, which was confirmed in preliminary experiments.

Second, a rather common problem in adversarial training for image generation is that the generated images are overlayed with high-frequency checkerboard artefacts (Palladino et al., 2020). This was also observed when using the proposed method for appearance adaptation in preliminary experiments. In particular, occasionally, the parameters of the appearance adaptation network converged to a state in which the adapted images showed checkerboard artefacts and the adaptation performance was very poor. In preliminary experiments, it was found that randomly shifting the input images of the discriminator by $\pm 1\,px$ prevents this behaviour, which is the reason why such a random shift is used in the proposed method. In particular, the shifted images are used as input for the domain discriminator instead of the original, non-shifted variants.

4.4.2.2 Update of $\mathcal{D}$

The second step of adversarial training is related to the update of the parameters of the discriminator network $\mathcal{D}$ (line 12 of Algorithm 1). In the following, the default loss for domain adversarial training is explained. Aiming at an improved semantic consistency of the adapted images, an extension of this loss is proposed, which is described in Section 4.5.1. In the base variant, the loss $\mathcal{L}_D$ minimised in the update step of $\mathcal{D}$ consists of two terms

$$\mathcal{L}_D(\Theta_A, \Theta_D, T^S, U^T) = \mathcal{L}_{advD,T}(\Theta_D, U^T) + \mathcal{L}_{advD,ST}(\Theta_A, \Theta_D, T^S). \tag{4.8}$$

This corresponds to the typical loss formulation for adversarial training and aims at training the discriminator network $\mathcal{D}$ to differentiate real images X^T from D^T and adapted source images X^{ST}. The first term aims at maximising the probability for images from D^T to be classified as coming from D^T:

$$\mathcal{L}_{advD,T}(\Theta_D, U^T) = -\frac{1}{n_B \cdot \tilde{p}^2} \sum_{b=1}^{n_B} \sum_{i=1}^{\tilde{p}^2} log(\psi_{b,i}^T), \tag{4.9}$$

where $\psi_{b,i}^T = P(y_{b,W(i)} = L_T | X^T)$ is the probability for the support window $W(i)$ corresponding to the i-th prediction in the discriminator output for the b^{th} image in a batch of image patches from D^T to be obtained from a target domain image. The remaining symbols are identical to those defined in the context of Equation 4.7.

The second term in Equation 4.8 aims at minimising the probability of adapted images from D^S to be classified as coming from D^T:

$$\mathcal{L}_{advD,ST}(\Theta_A, \Theta_D, T^S) = -\frac{1}{n_B \cdot \tilde{p}^2} \sum_{b=1}^{n_B} \sum_{i=1}^{\tilde{p}^2} log(1 - \psi_{b,i}^{ST}), \tag{4.10}$$

with $\psi_{b,i}^{ST} = P(y_{b,W(i)} = L_T | X^{ST})$ as defined in the context of Equation 4.7. The remaining symbols are identical to those defined in the context of Equation 4.7. Note that although this loss depends on the parameters of the appearance adaptation network, these parameters are not changed in this update step.

In order to update the parameters of $\mathcal{D}$, the gradients of $\mathcal{L}_D(\Theta_A, \Theta_D)$ with respect to Θ_D are used, which will train $\mathcal{D}$ such that it can discriminate well between real target images and adapted source images, thus making the task of the adaptation network $\mathcal{A}$ more difficult.

4.5 Improving Semantic Consistency

As extensively discussed in Section 3.3, it is crucial for the appearance adaptation to be done in a semantically consistent way to achieve an improvement of the classifier when trained on the adapted images.

So far, semantic consistency was targeted by training $\mathcal{A}$ to minimize the supervised loss $\mathcal{L}_{sup}^{ST}$ of adapted images. However, this approach cannot solve problems that are related to different distributions of label maps in the two domains. In particular, this refers to the situation in which the

domain gap is due to $P(Y)^S \neq P(Y)^T$ (cf. Section 2.6). Such a scenario can lead to a semantically inconsistent appearance adaptation, in which structures that are more frequent in the target domain are hallucinated and structures that are more frequent in the source domain are removed in the adapted images (Cohen et al., 2018). It is pointed out that $P(Y)^S \neq P(Y)^T$ can either be due to differences in the global label distributions or due to differences in the local distributions of label maps of the two domains. In particular, the global label distribution is the distribution of labels at pixel-level. If the global label distributions of two domains are different, it means that at least one class appears more frequently in one of the domains. On the other side, the local distribution of label maps refers to the arrangement of labels relative to each other in patches with a specific patch size. If the local label map distributions are different, some structures appear more frequently in either domain. Particularly relevant in adversarial training is the local label map distribution with respect to the window size that corresponds to the receptive field of the discriminator. An example for two regions with different local label map distributions is shown in Figure 4.4. Although the global label distributions of both regions are almost identical, there are differences in the local label map distributions; there are structures that appear only in one of the domains, as indicated by the black squares in Figure 4.4, resulting in a differences in the local distributions of label maps. For example, there are groups of very small trees in D^S, a structure that does not appear in D^T. On the other hand, D^T contains large areas of *low vegetation*, a pattern that hardly exists in D^S. Note that the area within a marker corresponds to $70 \times 70\,px$ at a resolution of $20\,cm$ per pixel, which is the receptive field of the discriminator used in this work.

Global label distribution [%]

	Sealed ground	Building	Low Vegetation	Tree	Car
D^S	35	41	11	11	2
D^T	34	42	9	11	4

Figure 4.4: Example of a challenging scenario for appearance adaptation. The two domains have almost identical global label distributions (cf. table to the right), but there are differences in the local distribution of label maps. The black squares, also indicated by the black arrows, indicate examples for structures in the label maps that appear only in one domain.

Such differences are problematic in adversarial training, because the distributions of the labels directly affect the distribution of the features. When using an adversarial training scheme, the

appearance adaptation network aims at producing images X^{ST} that have a distribution similar to the images in the target domain, i.e. such that $P(X)^{ST} = P(X)^{T}$. However, this will result in semantically inconsistent adaptations if $P(Y)^{S} \neq P(Y)^{T}$.

This is explained on the basis of a toy example, illustrated in Figure 4.5. In the example, an image X^S is adapted to different target domains $D^{T,1}$ - $D^{T,4}$. It is assumed that all domains consist of a single image, i.e. the ones shown in Figure 4.5. Thus, each image represents the corresponding feature distribution $P(X)$. Each image is associated with a label map Y that represents the global label distribution and the local distribution in the label map. In the toy example, there is a foreground class and a background class, which can be interpreted e.g. as *trees* in front of *low vegetation*. The classes correspond to the colours dark grey and light grey in the label maps, respectively. Furthermore, there are different kinds of *trees*, i.e. larger ones and groups of smaller ones. Those two structures appear in different frequencies in the domains.

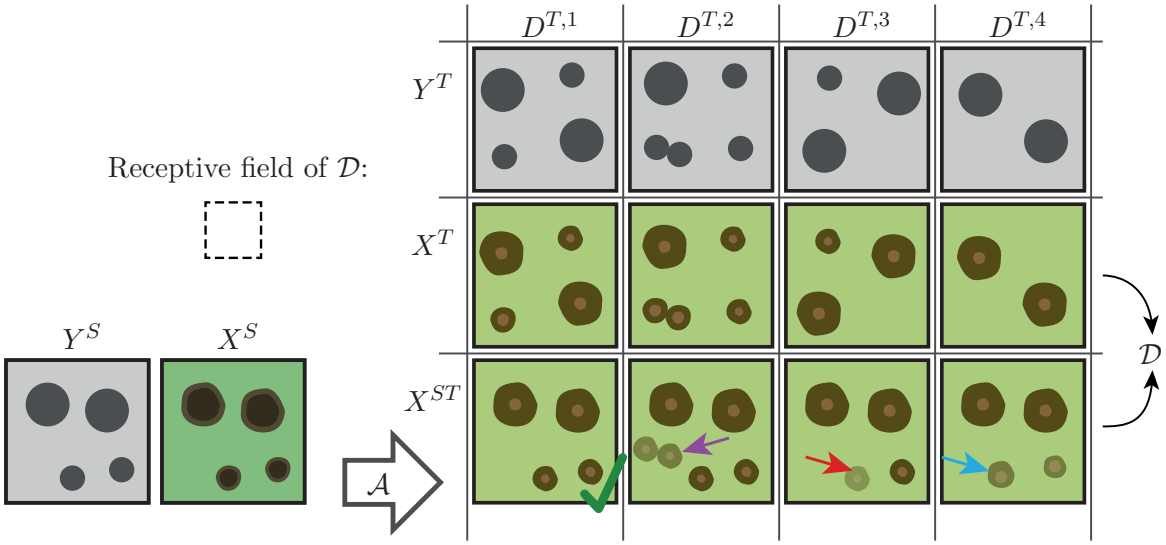

Figure 4.5: Toy example to illustrate the effect of differences in the label distributions of source and target domain with respect to semantic consistency. Here, the task is to classify each pixel into foreground objects, i.e. the blob-like structures, and background. An image X^S is adapted to multiple target domains $D^{T,1}$ - $D^{T,4}$. Each of them is associated with a target image X^T and the underlying label map Y^T. Y^S denotes the underlying label map of X^S and X^{ST} is the output to be expected for the appearance adaptation network $\mathcal{A}$ for the different target domains if the differences in the label distributions are not considered. Colour-codes: foreground (dark-grey), background (light-grey).

The third row in the table in Figure 4.5 illustrates the output to be expected of the appearance adaptation network X^{ST} for the different target domains when applying the proposed joint training scheme for appearance adaptation. In particular, the appearance adaptation network is trained to align the distributions $P(X)^{ST}$ of adapted images X^{ST} and the distribution $P(X)^{T}$ of images X^T, which is the main goal of adversarial training, but also to produce adapted images which are classified correctly by the classifier $\mathcal{C}$ which is adapted simultaneously. The dotted rectangle in Figure 4.5 illustrates the receptive field of the discriminator.

In the first case ($D^{T,1}$), the global and local label distributions of source and target domain are identical. Recall that the local label distribution refers to the receptive field of the discriminator, illustrated by the dotted rectangle in Figure 4.5. Here, a semantically consistent appearance adaptation is to be expected, as the distribution $P(X)^{ST}$ of the adapted image X^{ST} is equivalent to the distribution $P(X)^{T}$ of the images from D^{T} with respect to the receptive field of the discriminator. In the second scenario, in which the image is adapted to $D^{T,2}$, the global label distributions of both domains are still identical, but now the local distributions in the label maps are different, in particular, because in the target domain the structure of a large foreground object appears only once, while at the same time there is another group of smaller structures. Performing the appearance adaptation in a semantically consistent way (cf. X^{ST} when adapting to $D^{T,1}$) would result in $P(X)^{ST} \neq P(X)^{T}$, i.e. the distributions of the features with respect to the receptive field of $\mathcal{D}$ would be different. In this scenario $\mathcal{A}$ is likely to hallucinate the group of smaller objects (cf. purple arrow in Figure 4.5) in order to deceive the discriminator. A similar behaviour is to be expected when adapting to the domains $D^{T,3}$ and $D^{T,4}$, where some structures are under- or overrepresented, respectively, in the target domain. This is likely to result in (partial) removal of structures in X^{ST} (cf. red and blue arrows in Figure 4.5), because $\mathcal{A}$ tries to create an image that follows the distribution of $P(X)^{T}$ while at the same time trying to predict an image that is correctly classified by $\mathcal{C}$.

It is noted that, in principle, the degree of the semantic consistency can in most cases be increased by setting a relatively larger weight to the supervised loss term $\mathcal{L}_{sup}^{ST}$ in the joint training scheme. However, by doing so, the appearance adaptation network is heavily regularised, which may lead to adapted images that are semantically consistent but no longer give the impression of the images in the target domain.

To conclude, without any countermeasures, adversarial appearance adaptation is likely to fail if there are considerable differences in the local distributions of the label maps in both domains, and the adapted images will no longer be semantically consistent with the original images. Training on such images in combination with the original label maps from the source domain is expected to result in a poor classification after adaptation, because the training samples are no longer representative for the target domain. Based on this consideration, two methods are proposed in this thesis which both aim to improve semantic consistency even if there are such differences between the domains. The first method is based on reducing the variability in the probability maps predicted by $\mathcal{D}$. The second variant is based on training an auxiliary generator that predicts images that should compensate for potential differences in the label distributions of the two domains.

4.5.1 Method 1: Reduction of Variability

The first method to improve the semantic consistency corresponds to a reduction of the variability in the output of $\mathcal{D}$. This is motivated by the following line of thought, supported by observations in preliminary experiments. If the discriminator learns to distinguish real images from $D^{T}$and adapted source images based on features that are related to the label distributions, this will lead to semantically inconsistent adaptations, but it will also lead to a high variability in the output of $\mathcal{D}$. For instance, if D^{T} has a higher frequency of pixels corresponding to the class *vegetation* than

D^S, the discriminator will quickly learn to predict the probability for an image patch corresponding to the receptive field of the discriminator to originate from D^T based on the number of pixels that are representative for *vegetation*. Consequently, $\mathcal{D}$ will predict a high probability for such patches to come from D^T. At the same time, for patches that are more difficult to differentiate, i.e. that show a structure that appears in both domains, the predicted probabilities to originate from D^T might be close to 50 %. Such a situation results in a high variance in the predicted probability maps. Consequently, in order to increase semantic consistency, the variability in the output of $\mathcal{D}$ has to be reduced.

To that end, it is proposed to constrain the standard deviation of the output of the discriminator. In this way, $\mathcal{D}$ must also learn to differentiate domains in more difficult areas and thus to learn non-trivial differences between the domains. In practice, large standard deviations of the values of the discriminator output $\Psi^T = \mathcal{D}(X^T)$ for the images from D^T and $\Psi^{ST} = \mathcal{D}(X^{ST})$ for the adapted source images are penalised. This results in the regularization loss

$$\mathcal{L}_{reg}(\Theta_A, \Theta_D, T^S, U^T) = \mathcal{L}_{reg,T}(\Theta_D, U^T) + \mathcal{L}_{reg,ST}(\Theta_A, \Theta_D, T^S), \tag{4.11}$$

that consists of the two terms

$$\mathcal{L}_{reg,T}(\Theta_D, U^T) = \sqrt{\frac{1}{(n_B \cdot \tilde{p}^2) - 1} \sum_{b=1}^{n_B} \sum_{i=1}^{\tilde{p}^2} (\psi_{b,i}^T - \bar{\psi}^T)^2}, \tag{4.12}$$

and

$$\mathcal{L}_{reg,ST}(\Theta_A, \Theta_D, T^S) = \sqrt{\frac{1}{(n_B \cdot \tilde{p}^2) - 1} \sum_{b=1}^{n_B} \sum_{i=1}^{\tilde{p}^2} (\psi_{b,i}^{ST} - \bar{\psi}^{ST})^2}, \tag{4.13}$$

where $\bar{\psi}^T$ denotes the average value in Ψ^T for the batch $\{X_b^T\}_{b=1}^{n_B}$ of images from $D^T$and $\bar{\psi}^{ST}$ denotes the average value in Ψ^{ST} for the batch of adapted images $\{X_b^{ST}\}_{b=1}^{n_B}$. The remaining symbols are as described in the context of Equation 4.7. This regularization loss is weighted by the hyper-parameter ρ and added to the original discriminator loss from Equation 4.8, resulting in the modified loss term for the discriminator:

$$\mathcal{L}_{advD}^{RD}(\Theta_A, \Theta_D, T^S, U^T) = \mathcal{L}_{advD}(\Theta_A, \Theta_D, T^S, U^T) + \rho \cdot \mathcal{L}_{reg}(\Theta_A, \Theta_D, T^S, U^T). \tag{4.14}$$

It is noted that when using this method, the loss for the appearance adaptation network remains unchanged, i.e. as in Equation 4.5.

4.5.2 Method 2: Auxiliary Generator

The second method to achieve semantically consistent appearance adaptation in a scenario in which the domains have different local distributions of label maps corresponds to a modification of the overall architecture of the proposed method. This approach is based on the following line of thought. The problematic scenarios described above result from differences in the local distribution of the underlying label maps of the target images X^T and of the adapted source images X^{ST}, the latter distribution being identical to the one for the original source images X^S. As adversarial training aligns the local distributions of X^{ST} and X^T with respect to the receptive field of the

discriminator, this can result in semantically inconsistent adaptations. This could be prevented by modifying the adversarial training scheme in the following way. Let us assume, that an auxiliary image set $U^A = \{X_i^A\}$ is available in which each auxiliary image X_i^A has the appearance of images from $D^T$and follows a distribution $P(X)^A$ that fulfils the condition

$$\beta \cdot P(X)^A + (1 - \beta) \cdot P(X)^{ST} = P(X)^T. \tag{4.15}$$

If such an image set U^A is available, the local distribution $P(X)^{A,ST}$ of a mixed set $U^{A,ST}$ of images that contains $\beta \cdot 100\,\%$ auxiliary images from U^A and $(1 - \beta) \cdot 100\,\%$ adapted source images X^{ST}, would be identical to $P(X)^T$, i.e. $P(X)^{A,ST} = P(X)^T$.

The benefit of this formulation is that $P(X)^{ST}$ is no longer required to match $P(X)^T$ exactly, but only the distribution $P(X)^{A,ST}$ of the mixed set $U^{A,ST}$ has to match $P(X)^T$. This leaves more freedom to the appearance adaptation network, which is assumed to result in a larger degree of semantic consistency. Following this line of thought, the second method to achieve semantically consistent appearance adaptations consists of a modified adversarial training scheme that does not result in $P(X)^{ST} = P(X)^T$ but instead in $P(X)^{A,ST} = P(X)^T$. In the following, the motivation of this approach is underlined using a toy example in Figure 4.6 before further details about the training scheme will be provided.

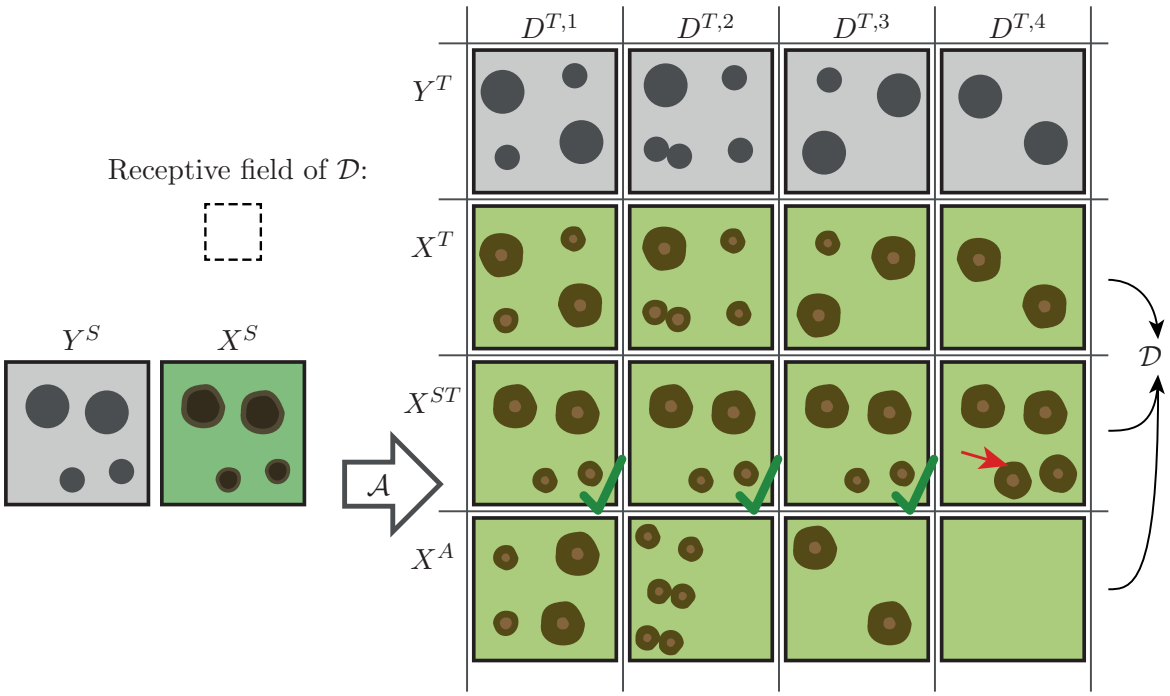

Figure 4.6: Toy example to illustrate how auxiliary images X^A help the appearance adaptation to achieve semantic consistency. An image X^S is adapted to multiple target domains $D^{T,1}$ - $D^{T,4}$, each associated with a target image X^T and an underlying label map Y^T. Y^S denotes the underlying label map of X^S and X^{ST} is the output to be expected for the appearance adaptation network $\mathcal{A}$ for the different target domains. Colour-codes as in the description of Figure 4.5.

Similarly to Figure 4.5, Figure 4.6 illustrates the appearance adaptation to be expected for an image X^S to different target domains $D^{T,1}$ - $D^{T,4}$, considering the auxiliary images X^A. In this

variant, the adversarial training does not aim at the equilibrium $P(X)^{ST} = P(X)^T$ but instead $P(X)^{A,ST} = P(X)^T$. In particular, the adapted images in the third row in Figure 4.5 illustrate the expected output of the appearance adaptation network X^{ST} for the different target domains and the last row illustrates the auxiliary images X^A that are chosen such that Equation 4.15 is fulfilled. Note that in the example, there is one auxiliary image for each adapted image, i.e. $\beta = 0.5$. When adapting to the first target domain, in which $P(X)^T = P(X)^S$, $\mathcal{A}$ can perform the appearance adaptation in a semantically consistent way. The auxiliary image is expected to be a random image that fulfils $P(X)^A = P(X)^T$. In the second and third case, i.e. when adapting to $D^{T,2}$ and $D^{T,3}$, the label distribution in D^T is different from the one in D^S. This resulted in semantically inconsistent adaptations when using the default adversarial training scheme (cf. Figure 4.5). However, when considering the auxiliary images and the modified adversarial training scheme, X^{ST} can be semantically consistent with X^S, as Equation 4.15 is still fulfilled. In particular, the local feature distribution in the mixed set $U^{A,ST}$ is equivalent to the one in X^T. For example, in $D^{T,3}$ there are two large objects and one small object in one patch, which results in the same local feature distribution than in $U^{A,ST}$, in which four large objects and two small objects appear in two patches.

Yet, this approach has some limitations, i.e. if the difference in the local distributions of the label maps between the two domains is too large, the auxiliary images may not compensate for this difference. For example, when adapting to $D^{T,4}$, a non-consistent transformation of the structures is to be expected (cf. red arrow in Figure 4.6), because the small objects do not appear in the target domain.

In conclusion, auxiliary images can help to achieve semantically consistent appearance adaptation in scenarios in which the regular adversarial training scheme would fail. In the following, the method resulting from this consideration is explained. The general idea is to learn to generate images X^A jointly with performing the appearance adaptation and to use these images to improve the semantic consistency of the adapted source images. This results in a modified architecture that is depicted in Figure 4.7.

An auxiliary generator $\mathcal{G}$ is introduced. In each training iteration, $\mathcal{G}$ generates a batch of n_G images X^A based on n_G random noise vectors z. They should look like images from D^T, so that $\mathcal{G}$ is trained to maximise the probability predicted by D for X^A to originate from D^T, i.e. $P(y = L_T|X^A)$ (cf. purple arrow in Figure 4.7). The main purpose for considering these images is that they compensate for the differences in the label distributions between the domains. To that end the adversarial training scheme is extended as described in detail in Section 4.5.2.2. Using the extended adversarial training scheme, $\mathcal{G}$ and $\mathcal{A}$ are both trained to deceive the discriminator, i.e. to generate images that look like coming from the target domain. Simultaneously, $\mathcal{D}$ is trained to correctly predict whether an input image patch comes from the target domain or not. In the extended training scheme, the input images for training the discriminator are either images from X^T, adapted images X^{ST} or auxiliary images X^A. Image patches from X^{ST} and from X^A are to be classified as not coming from the target domain by the discriminator. As the adversarial training scheme will align the distribution of images in D^T and the distribution of the remaining images, it will eventually reach an equilibrium in which Equation 4.15 is fulfilled.

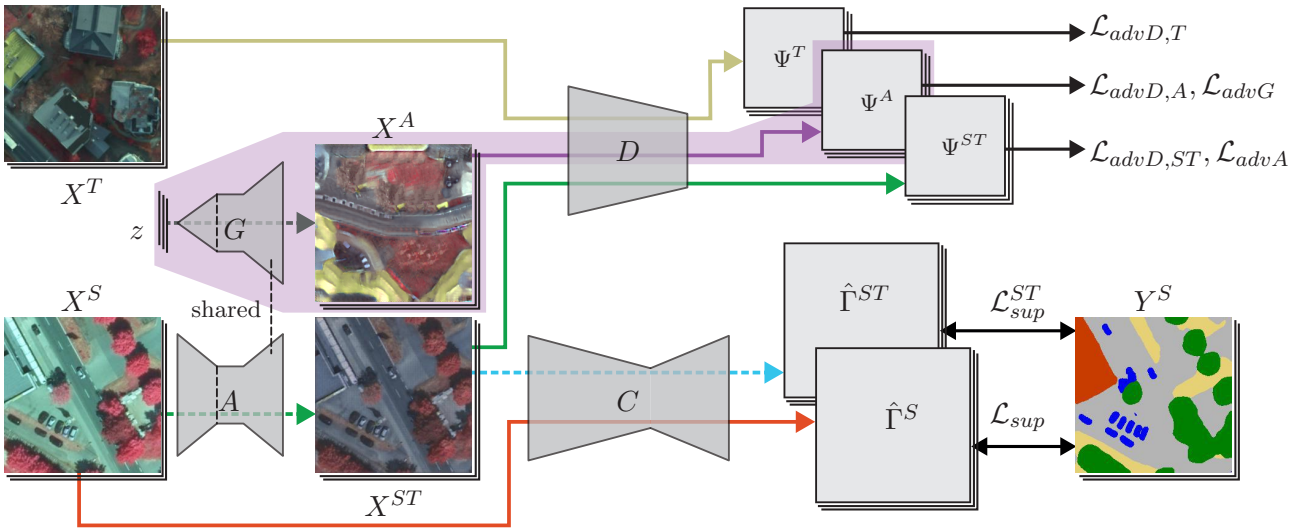

Figure 4.7: Proposed extension of the training scheme for joint appearance adaptation by an auxiliary generator $\mathcal{G}$ aiming to increase the semantic consistency of the adapted image X^{ST} with the image X^S; the coloured arrows indicate the processing flow in the UDA phase and the solid black arrows correspond to loss terms. The main parts remain unchanged (cf. Figure 4.2); the differences are highlighted with purple background. In this variant the auxiliary generator $\mathcal{G}$ is introduced. This network takes a batch of random vectors z and predicts one auxiliary image X^A per vector. The last layers of $\mathcal{G}$ are shared with the network $\mathcal{A}$ (cf. Section 4.5.2.1). In the extended variant, $\mathcal{D}$ also predicts the origin of the auxiliary images, i.e. whether they come from D^T or not. The corresponding probability maps Ψ^A predicted by $\mathcal{D}$ for the X^A are considered in the additional loss terms $\mathcal{L}_{advD,A}$ and $\mathcal{L}_{advG}$ (cf. Section 4.5.2.2).

As a last remark, aiming to reach an equilibrium in which Equation 4.15 is fulfilled does not guarantee the images X^{ST} to be semantically consistent with X^S. In particular, the equilibrium would also be fulfilled if the outputs of $\mathcal{A}$ and $\mathcal{G}$ were swapped. Considering the toy example from Figure 4.6, this would mean that when swapping the images X^{ST} with the corresponding auxiliary images X^A, Equation 4.15 would still be fulfilled, but the adapted images would no longer be semantically consistent. However, it has to be considered that the images X^{ST} are further constrained by the supervised loss term $\mathcal{L}_{sup}^{ST}$, which also aims at semantic consistency. On the other hand, the auxiliary images are not further constrained but only affect the adversarial loss. Thus, it is assumed that when using the extended adversarial training scheme, the appearance adaptation network will converge to a state in which the images are adapted in a semantically consistent way, while the auxiliary generator will be trained to predict images such that Equation 4.15 is fulfilled.

In Sections 4.5.2.1 and 4.5.2.2, the architecture of $\mathcal{G}$ and the modified loss formulations are presented, respectively.

4.5.2.1 Architecture of $\mathcal{G}$

The auxiliary generator $\mathcal{G}$ takes a random Gaussian noise vector z of length n_z as input and predicts an image patch X^A with n_C channels and a spatial size of $p \times p\,px$, where p is the patch size that

is used for training the remaining networks (cf. Section 4.1). Generating images at high resolution is a rather difficult task and usually requires progressive learning, which means that training is performed in multiple phases, where in each phase the resolution of the images is increased (Karras et al., 2018). As such a strategy is rather slow and requires a unhandy training schedule it is not used in this thesis. Instead, the presented approach exploits the fact that both networks $\mathcal{A}$ and $\mathcal{G}$ should produce images that look like coming from D^T, which is the reason why these networks share the n_{SH} last layers. In this work, n_{SH} is empirically set to 17. Consequently, the auxiliary generator consist of two parts. The first layers start from a $1 \times 1 \times n_z$ random noise vector and process this vector by several transposed convolutional layers until the spatial size is $p/4$. The second part corresponds to the last n_{SH} layers of $\mathcal{A}$, the weights of which are shared between $\mathcal{A}$ and $\mathcal{G}$. Table 4.5 provides the full architecture of $\mathcal{G}$.

Layer(s)	Layer type	h, w	Depth	Shared
1	Input layer	1	n_z	n
2	T-Conv, ReLU, BN	$p/32$	256	n
3	Conv, ReLU, BN, DROP	$p/32$	512	n
4	T-Conv, ReLU, BN	$p/16$	256	n
5	Conv, ReLU, BN	$p/16$	256	n
6	T-Conv, ReLU, BN	$p/8$	256	n
7	Conv, ReLU, BN	$p/8$	256	n
8	T-Conv, ReLU, BN	$p/4$	256	n
9	Conv, ReLU, BN	$p/4$	256	n
10-25	15× Residual block	$p/4$	256	y
26	T-Conv, ReLU	$p/2$	128	y
27	T-Conv	p	d	y

Table 4.5: Architecture of the auxiliary generator $\mathcal{G}$. T-Conv: transposed convolution. DROP: Dropout layer with a dropout probability of 10 %. For other abbreviations, cf. Table 4.1. The structure of a residual block is given in Table 4.3. Shared: this column indicates whether the layer is shared with appearance adaptation network $\mathcal{A}$. The shared layers correspond to layers 3-20 in Table 4.2.

All convolutional layers in the architecture use 3×3 kernels and zero-padding by $1\,px$. All transposed convolutional layers except for the first one in layer 2 use 4×4 kernels and perform upsampling by a factor of 2. The first transposed convolutional layer is applied to the $1 \times 1 \times n_z$ random noise vector and predicts an activation map with the extent $p/32 \times p/32 \times 256$. Thus, the kernel size is $p/32 \times p/32$ and the upsampling factor is also $p/32$. The parameters of all layers of the network are initialised randomly based on (He et al., 2015). This architecture was chosen based on a limited number of preliminary experiments in which different variants were compared.

4.5.2.2 Modifications of Adversarial Loss Terms

The extension of the architecture with the auxiliary generator $\mathcal{G}$ results in a few minor modifications of the loss terms. First, the joint loss of $\mathcal{A}$ and $\mathcal{C}$ (cf. Equation 4.5) is replaced by the joint loss $\mathcal{L}_{A,C,G}$ for $\mathcal{A}$, $\mathcal{C}$ and $\mathcal{G}$ which us used to update the respective parameters:

$$\mathcal{L}_{A,C,G}(\Theta_{A,C}, \Theta_G, \Theta_D, T^S) = \mathcal{L}_{A,C}(\Theta_{A,C}, \Theta_D, T^S) + \omega_G \cdot \mathcal{L}_{advG}(\Theta_D, \Theta_G), \tag{4.16}$$

where Θ_G is the parameter set of $\mathcal{G}$. Practically, the original loss $\mathcal{L}_{A,C}$ (cf. Equation 4.5) is extended by an additional loss term $\mathcal{L}_{advG}$, weighted by ω_G. This loss term aims at maximising the probability predicted by $\mathcal{D}$ for the generated images X^A to originate from D^T. Thus, it is formulated similar to the adversarial loss term $\mathcal{L}_{advA}$ for the appearance adaptation network (cf. Equation 4.7):

$$\mathcal{L}_{advG}(\Theta_D, \Theta_G) = -\frac{1}{n_G \cdot \tilde{p}^2} \sum_{b=1}^{n_G} \sum_{i=1}^{\tilde{p}^2} log(\psi_{b,i}^A), \tag{4.17}$$

where $\psi_{b,i}^A$ corresponds to the probability $P(y_{b,W(i)} = L_T | X_b^A)$ for the support window $W(i)$ corresponding to the i-th pixel in the probability map predicted by $\mathcal{D}$ for the b-th image in the batch of n_G auxiliary images $\{X_b^A\}_{b=1}^{n_G}$ to originate from D^T. The remaining symbols are identical to those described in the context of Equation 4.7. Note that the ratio n_G/n_B corresponds to the parameter β in Equation 4.15. If the difference of the label distributions of both domains is higher, n_G and, thus, β has to be larger in order to compensate for such differences. Of course, the difference is unknown in a practical application, thus, n_G has to be tuned as a hyper-parameter of the method. Furthermore, if n_G is set to a value that is larger than required, the positive effects would remain, but only the memory footprint would be increased, which is considered less critical.

The second loss modification in this extended method affects the discriminator loss $\mathcal{L}_D$ from Equation 4.8, which becomes the extended discriminator loss

$$\mathcal{L}_{D+}(\Theta_A, \Theta_D, \Theta_G, T^S, U^T) = \mathcal{L}_{advD,T}(\Theta_D, U^T) + \mathcal{L}_{advD,ST}(\Theta_A, \Theta_D, T^S) + \mathcal{L}_{advD,A}(\Theta_G, \Theta_D). \tag{4.18}$$

The original loss $\mathcal{L}_D$ from Equation 4.8 is extended by a third term that considers the auxiliary images X^A. In order to correctly predict images X^A produced by the auxiliary generator as not originating from D^T the loss is formulated as

$$\mathcal{L}_{advD,A}(\Theta_G, \Theta_D) = -\frac{1}{n_G \cdot \tilde{p}^2} \sum_{b=1}^{n_G} \sum_{i=1}^{\tilde{p}^2} log(1 - \psi_{b,i}^A), \tag{4.19}$$

with the symbols being identical to those introduced in the context of Equations 4.17 and 4.7.

The overall training scheme of appearance adaptation remains the same. In each training iteration, first, the parameters of $\mathcal{A}$, $\mathcal{C}$ and $\mathcal{G}$ are updated by minimising $\mathcal{L}_{A,C,G}$ before the parameters of $\mathcal{D}$ are updated by minimising $\mathcal{L}_{D+}$ in the second step.

4.6 Entropy-based Parameter Selection

Many UDA methods rely on iterative training procedures that are repeated for a fixed number of iterations, with the parameter set after the very last iteration being used for inference (Tasar et al., 2020b,a; Benjdira et al., 2019). For the reasons given in Section 3.5, a different strategy is proposed in this thesis, namely to select the final parameter set according to an optimality criterion derived from the data in D^T. As there are no labelled data in D^T, this criterion cannot be based on the validation error in that domain. Instead, the average entropy of the predicted class scores for images X^T from D^T is used as an approximate measure for the validation performance.

The entropy of the class scores can be interpreted as a measure of uncertainty of the predictions (Wittich, 2020; Pan et al., 2020). Based on this interpretation, good parameters can be expected to lead to low uncertainty of the predictions and, thus, to low entropy. After each epoch of adaptation, the classification network $\mathcal{C}$ is used to predict the probabilistic class scores $\hat{\gamma}^T = (\hat{\gamma}_1, \ldots, \hat{\gamma}_{n_L})$ for each pixel of all n_T images from U^T. For the i-th pixel in image X_j^T, the entropy $E_{j,i}$ is computed according to

$$E_{j,i} = -\sum_{k=1}^{n_L} \hat{\gamma}_k \cdot log(\hat{\gamma}_k), \tag{4.20}$$

and the average entropy

$$\bar{E}_j = \frac{1}{n_{px}} \sum_{j=1}^{n_T} \sum_{i=1}^{h_j \cdot w_j} E_{j,i} \tag{4.21}$$

is determined from all n_{px} pixel-wise values, where

$$n_{px} = \sum_{j=1}^{n_T} h_j \cdot w_j \tag{4.22}$$

and h_j, w_j are height and width of image X_j^T. Note that the height and width is not the same for all images. The parameter set $\hat{\Theta}_C^{(jaa)}$ having the lowest average entropy is selected after running the adaptation for a n_E epochs. To save computation time, the mean entropy is not computed for the first n_{min} epochs, because $\mathcal{A}$ and $\mathcal{D}$ are not expected to deliver meaningful results in the beginning of the adaptation phase (cf. lines 5 and 15-20 in Algorithm 1).

4.7 Adaptive Batch Normalization

The common procedure when using 2D batch normalisation layers is to track running averages of the mean and standard deviation of the activations in each channel (cf. Section 2.3.3). The set of all running averages $\Theta_{C,B}$ approximates the *domain statistics*, i.e. the average values and standard deviations of the activation values, when computed for a whole dataset (domain). In a domain adaptation scenario, it may not be reasonable to use the statistics that were calculated based on the images of the source domain, because the statistics of the target domain may be different.

In this thesis, it is proposed to address this issue by an optional adaptation step, which is to apply ABN (cf. Section 2.6.1) after having adapted the classifier using the proposed method based on appearance adaptation. The input of ABN consists of the images from D^T and the classifier $\mathcal{C}$ with the parameter set $\hat{\Theta}_C^{(jaa)} = (\hat{\Theta}_{C,S}^{(jaa)}, \hat{\Theta}_{C,B}^{(jaa)})$ resulting from the joint appearance adaptation. Instead of using the running averages $\hat{\Theta}_{C,B}^{(jaa)}$ for the inference, the running averages are updated to approximate the statistics of the target domain. Note that the statistics are practically not calculated exactly on the entire target domain dataset, as this would require too much memory. Instead, an iterative approximation is used (Li et al., 2018). In particular, to approximate the statistics from the target domain, $n_{I,ABN}$ mini-batches with a batch size of $n_{B,ABN}$ that contain patches from target domain images are presented to the classification network. In each forward pass, the running averages in the 2D batch normalisation layers in the network are updated as described in Section 2.3.3. After having processed all mini-batches the updated parameters $\hat{\Theta}_{C,B}^{(abn)}$ are obtained. The final parameter set used for the inference is $\hat{\Theta}_C^{(abn)} = (\hat{\Theta}_{C,S}^{(jaa)}, \hat{\Theta}_{C,B}^{(abn)})$.

4.8 Resolution Adaptation

In adaptation scenarios related to RS applications it might happen that the ground sampling distance (GSD) of the source domain (GSD^S) is different from the one of the target domain (GSD^T). In such a case, objects appear at a different scale in both domains, which can lead to differences in the local distribution of labels in the label maps in the two domains. As discussed in Section 4.5, this poses additional challenges for the appearance adaptation. Furthermore, having different resolutions affects the appearance of objects in the two domains. This can strongly increase the domain gap, which may no longer be bridged with methods for UDA.

Obviously, the larger the difference in the GSD of both domains is, the larger the domain gap becomes and, thus, the more difficult it becomes to achieve a successful adaptation of a classifier. This was empirically shown in literature. For example, Liu et al. (2020) and Benjdira et al. (2019) try to adapt a classifier from a source domain with a GSD of 9 cm to a target domain with 5 cm and vice versa using methods for deep UDA. In both scenarios, the initial target domain performance in of a classifier that was trained in D^S is very poor. It was shown that even after domain adaptation the performance in the target domain remained at a rather low level.

In this thesis, a possible difference of the GSD in the source domain and the one in the target domain is not addressed by deep UDA. Instead, to overcome this problem, the data of the source domain are pre-scaled using the information about the GSD of both domains, which is usually available in RS applications if the images are georectified.

Practically, if $GSD^T \neq GSD^S$, i.e. if the data in D^S have a different geometrical resolution than the data in D^T, the data from D^S are either downsampled or upsampled to GSD^T. This includes resampling of both, the image data (using bilinear interpolation) and the reference (using nearest neighbour interpolation). Note that if the image data are downsampled to GSD^T, Gaussian smoothing is applied first to avoid aliasing effects. Then, source training and UDA are performed using the resampled data from D^S and the original data from D^T.

5 Experimental Setup

The proposed method for UDA is evaluated based on two applications using multiple datasets. These datasets are introduced in Section 5.1. Section 5.2 provides the evaluation protocol and quality metrics which are used to assess the performance of the classifiers after source training and after domain adaptation. Furthermore, an evaluation method for assessing semantic similarity is proposed. This is followed by an overview of the experiments in Section 5.3. Section 5.4 describes the hyper-parameters used in the experiments and describes how methods from literature are adapted to allow a comparison to the state of the art.

5.1 Datasets

The main goal of the experiments is to assess the performance of the proposed method for UDA for the pixel-wise classification of remotely sensed imagery. To this end, two groups of datasets are used that address two different applications. The first group of datasets, introduced in Section 5.1.1, addresses the application of land-cover classification based on airborne imagery and co-registered, rasterised height information. Here, each pixel in the images is to be classified according to typical land-cover classes such as *vegetation*, *building* or *road*. The datasets in the second group, introduced in Section 5.1.2, address the application of deforestation detection in the Brazilian Amazon region using satellite imagery. Here, the goal is to detect recent deforestation at pixel-level based on a pair of satellite images showing the same region but acquired at different dates. This application is related to the work by the National Institute for Space Research (INPE) in Brazil, which provides manually generated references for new deforestation areas on a yearly basis (Assis et al., 2019).

5.1.1 Data for Land-cover Classification using Aerial Imagery

For the evaluation of the proposed method with respect to the first application, seven datasets are used, each showing a different German city and each being treated as a single domain. The first group of datasets consists of orthorectified multi-spectral images (MSI), height data and label maps for the cities of Schleswig (S), Hameln (Hm), Buxtehude (B), Hannover (H) and Nienburg (N) with a GSD of 20 cm (Wittich, 2020). All of these datasets include `RGBI` images (red, green, blue, near infrared) and a normalised digital surface model (nDSM) which contains the height above ground for each pixel. The height information comes from image based 3D reconstruction. It is noted that the data were pre-processed by the providers of the dataset and there are some differences between the datasets related to the preprocessing, i.e. to the georectification and the generation of the nDSMs. In the experiments, the blue channel is not used because it is not available for another

dataset (Vaihingen) introduced below. A reference was generated at pixel level by manual labelling according to the class structure shown in Table 5.1.

Further, the Potsdam (P) and Vaihingen (V) datasets of the ISPRS labelling benchmark are used (Wegner et al., 2017). They consist of orthorectified MSI, nDSMs and label maps with 6 classes as shown in Table 5.1. P consists of `RGBI` images, captured with a GSD of 5 cm, whereas the imagery from V has a GSD of 9 cm and does not include a blue channel. Both datasets were split into training and testing areas by the benchmark organizers. In some experiments, resampled versions of the datasets P and V are used, which is indicated by a subscript showing the GSD, e.g. V_{20} refers to data from V resampled to a GSD of 20 cm. To that end, bilinear interpolation is used for resampling the image and height data, and nearest neighbour interpolation is used for the label maps. In order to align the class structure of P and V to the one of the other cities, the class *Clutter* is ignored, as suggested in (Liu et al., 2020). The corresponding regions in the reference do not contribute to the average loss during training, and at test time they are not considered in the evaluation procedure. Whenever *Clutter* is not ignored, the datasets for Potsdam and Vaihingen are denoted by P' and V', respectively.

City		P'_5	V'_9	P_{20}	V_{20}	S	Hm	B	H	N
Capturing season		A	S	A	S	S	A	S	S	S
Size [M px]										
Training		648	56.9	40.5	11.5	13	18	55.6	55.6	55.6
Validation		216	21.2	13.5	4.3	4	7	11.1	11.1	11.1
Testing		504	90.2	31.5	18.2	9	12	33.3	33.3	33.3
All		1368	168.3	85.5	34	26	37	100	100	100
Class distribution [%]										
Sealed ground	(SG)	29.6	27.7	31.1	27.9	14.1	18.8	22.1	33.6	22.8
Building	(BU)	25.7	26.0	27.0	26.2	14.7	19.1	19.7	36.7	18.4
Low vegetation	(LV)	22.6	21.6	23.8	21.8	38.9	36.2	36.9	7.5	40.3
High vegetation	(HV)	15.5	22.7	16.2	22.8	31.5	24.5	20.3	20.6	17.8
Vehicle	(VH)	1.8	1.3	1.9	1.3	0.8	1.3	1.0	1.6	0.7
Clutter	(CL)	4.8	0.7	-	-	-	-	-	-	-

Table 5.1: Dataset overview; capturing season is either autumn (A) or summer (S). Class distribution: percentage of pixels assigned to the corresponding class in the corresponding city.

Figure 5.1 gives an overview of the data in all domains and Table 5.1 shows the global label distributions and the overall size of each dataset, along with the capturing season as another possible reason for a domain gap. In all datasets, the class *vehicle* is strongly under-represented. Also, the class distributions are very different between the datasets, which is problematic for appearance adaptation, as discussed in Section 4.5. For example, S has a much larger amount of *high vegetation* (31.5 %) than P (15.5 %), and H has a much smaller amount of *Low Vegetation* (7.5 %) than B (36.9 %). The domain pairs (P_{20}, V_{20}) and (B, N) are rather similar with respect to the marginal label distribution. Besides the differences in the global label distributions of the datasets, there are also differences in the local distribution of labels in the label maps (cf. Section 4.5). In particular,

several structures appear with different frequencies. An obvious example is the structure of a larger area (e.g. a square of size $50 \times 50\,m^2$) that shows only low vegetation. This pattern appears frequently in Nienburg, Buxtehude, Schleswig and Hameln, but barely in Potsdam and never in Vaihingen or Hannover, as those images show more densely built-up areas. Such differences are very challenging when applying image adaptation as discussed in Section 4.5.

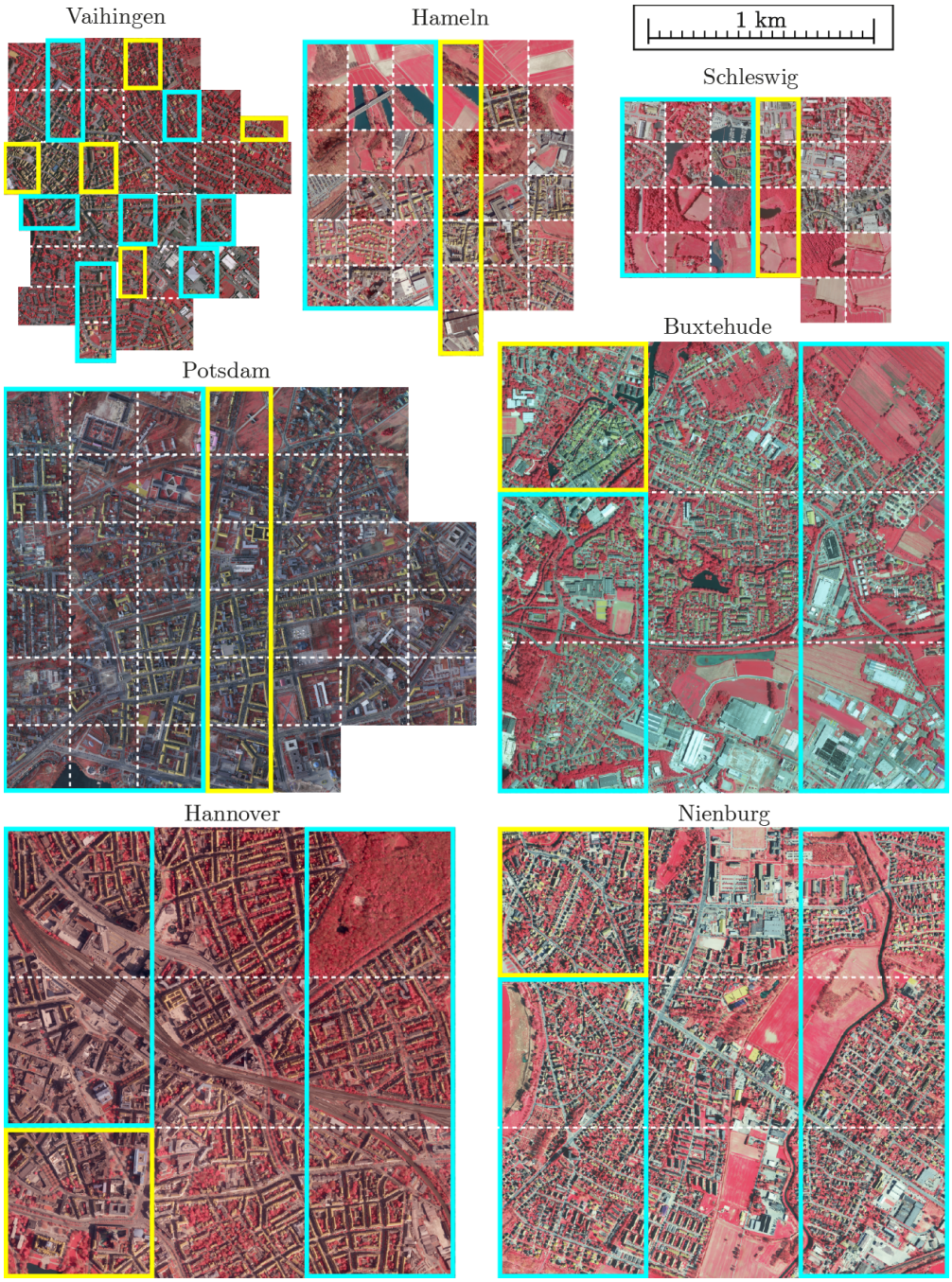

Figure 5.1: Dataset overview; False-colour images (infra red, red, green) for the 7 datasets for land cover classification. Areas with cyan outline are used for training, yellow corresponds to the validation set. The remaining areas are used for testing. The dotted lines show the borders of the individual images.

All datasets are split into subsets for training, validation and testing. Note that the test sets from P and V correspond to those suggested by the organisers of the ISPRS labelling benchmark.

Figure 5.1 illustrates the split of the data into the subsets for training, validation and testing. In the figure, it can also be seen that there are some obvious systematic differences in the image data from different domains, such as different lighting conditions that affect the contrast between areas in the sun and those in the shade. Also, the characteristics regarding colour and brightness are different; for example the images from Potsdam are darker and the images in Buxtehude appear more bluish compared to the other domains. Such differences are to be expected because the appearance of colours is affected e.g. by the weather conditions and the sensor configuration. To slightly reduce these differences, the image data of each dataset is pre-processed so that after normalisation each channel has a mean of zero and a standard deviation of one. In case of the nDSMs, the height values are also shifted to have a mean of zero, but they were divided by a constant value of $30\,m$ (instead of the standard deviation) to preserve the relative metric height information.

Figure 5.2 shows the MSI, the nDSM and the reference for exemplary patches from the different domains. It can be seen that there are further differences in the domains with respect to the height information. In particular, the nDSM from Hannover has a different appearance compared to the other domains. It appears less smooth and height changes are badly aligned with the object boundaries. The height maps from Schleswig and Hameln appear less sharp than those from the other domains. Note that the height information was provided by the respective authors of the datasets and was only available in the form of the rasterized maps as exemplarily presented in Figure 5.2.

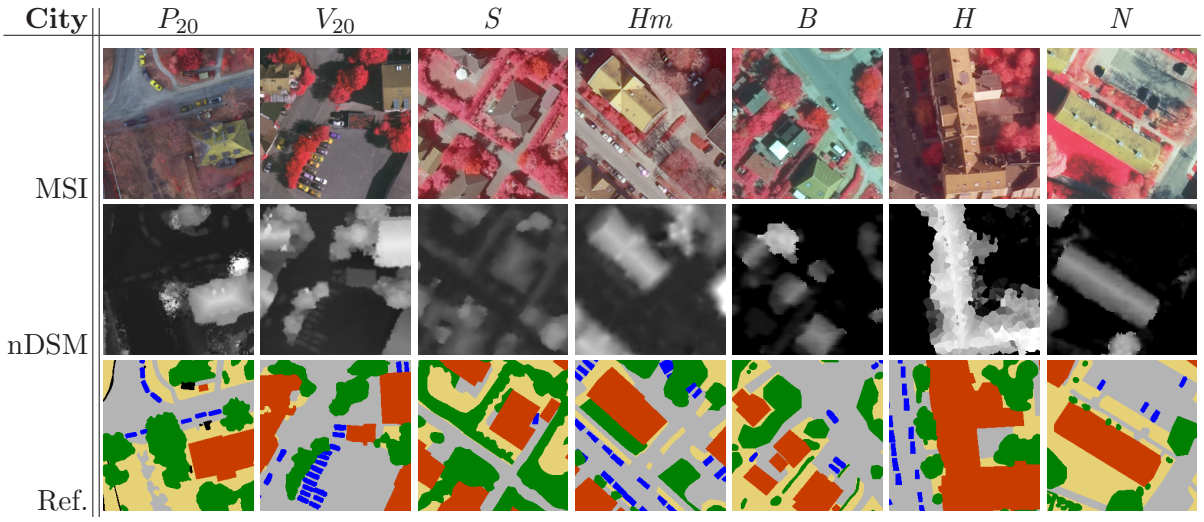

Figure 5.2: Exemplary data samples from the aerial image classification datasets. The side length of the patches is $256 \times 256\,px$ and the GSD is $20\,cm$. MSI: False-colour multi-spectral image (infrared, red, green). nDSM: rasterized normalised digital surface model. Brighter pixels correspond to larger heights. Ref.: Reference label map. Colour-codes: *SG* (grey), *BU* (red), *HV* (green), *LV* (yellow), *VH* (blue), *CL* (black) with class abbreviations according to Table 5.1.

5.1.2 Data for Bi-temporal Deforestation Detection using Satellite Imagery

The presented method for UDA is further evaluated for the application of deforestation detection. This application is driven by the work of INPE, which performs a yearly manual investigation of Brazil to detect legal and illegal deforestation. The institute provides freely available reference data via the TerraBrasilis[1] platform (Assis et al., 2019). INPE performs a yearly manual labelling of newly deforested areas based on different databases, but mainly based on imagery coming from the Landsat program of NASA. Multiple images from different acquisition dates from late May to September are considered in the manual analysis depending on the cloud coverage and data availability (Soto et al., 2021). Areas which were labelled as deforested in the past are not considered in the analysis. The yearly information about new deforestation areas are provided in the form of polygons that describe the outline of new deforestation areas.

This application corresponds to the following classification scenario. The input to the classifier consists of a pair of two satellite images that show the same region, but that were taken at different times, and a binary map that contains the areas that were recognised as deforested in the past. The classifier's task is to predict for each pixel that has not been labelled as deforested in the past whether it shows deforestation. In particular, the class *deforestation* means that there is forest in the earlier image and no forest in the later image.

In (Soto et al., 2021; Soto et al., 2020), three datasets were proposed for evaluating methods for UDA for this application. To enable a comparison of the proposed method to existing ones, the same datasets are also used in this thesis to evaluate the proposed method. The datasets correspond to three different areas, two of which are located in the Amazon biome and one is located in the Cerrado biome in Brazil. The first region (MA) is located in the the state Maranhão in the north-west of the Amazon biome. The second region (PA) lies in the state Pará in the north of the Cerrado biome, and the third region (RO) is in the state of Rondônia in the south of the Amazon biome. Table 5.2 gives the location of the tree test sites.

Test site	North-West	South-East
Maranhão (*MA*)	S4°44'52" W44°1'23"	S5°12'48" W43°37'58"
Pará (*PA*)	S3°8'21" W51°16'12"	S3°26'16" W50°34'4"
Rondônia (*RO*)	S9°36'51" W64°20'51"	S10°18'35" W62°56'41"

Table 5.2: Geographical coordinates (latitude and longitude) of the North-West and South-East corners of the three datasets.

For each region, the dataset consists of two Landsat 8 images (X_e, X_l), both of which show the whole area of the respective region, but were acquired at different dates. Furthermore, each dataset contains the past deforestation maps and the reference for the deforestation that happened between the two dates. The latter two are derived from the manual annotations provided by INPE. The acquisition dates of the two images which define the deforestation period are selected according to the reference, i.e. the images, that were used in the manual labelling process are considered. The

[1]URL: http://terrabrasilis.dpi.inpe.br (last accessed on 23. November 2022)

image acquisition dates are provided in Table 5.3. Each region is treated as a different domain, assuming that the regional differences and differences in the acquisition dates affect the appearance of vegetated and non-vegetated areas in the images (Soto et al., 2021).

Test site	Acq. date of X_e	Acq. date of X_l
Maranhão (*MA*)	18/08/2017	21/08/2018
Pará (*PA*)	02/08/2016	20/07/2017
Rondônia (*RO*)	18/07/2016	21/07/2017

Table 5.3: Acquisition dates of the images for all test sites.

Figure 5.3 provides examples for the image pairs and reference labels coming from the three test sites.

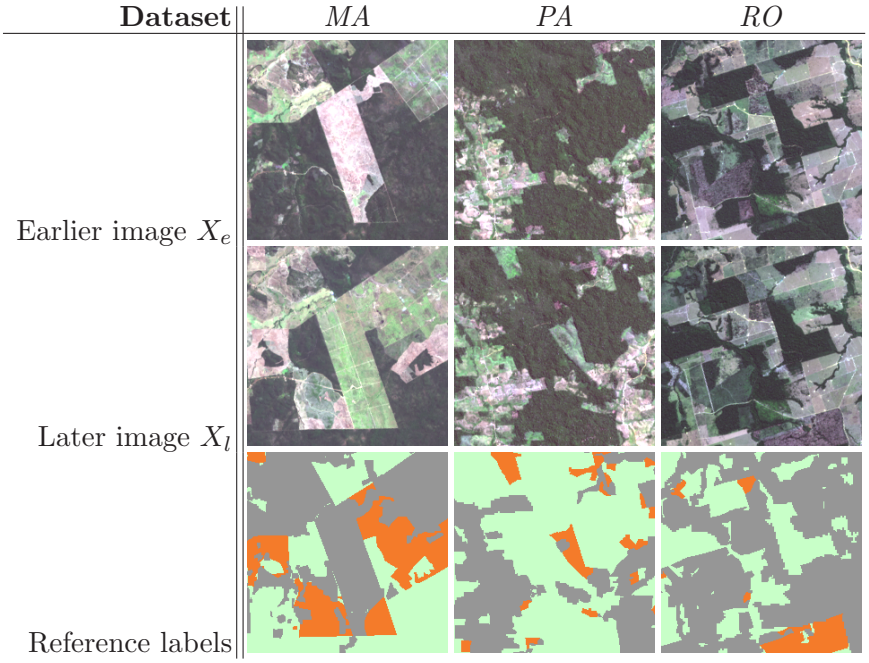

Figure 5.3: Exemplary samples from datasets for deforestation detection; the side length of the patches is $224\,px$ ($6720\,m$). Note that only the red, green and blue channels are visualised. Colour-codes of the reference: *Unknown* (grey), *deforestation* (orange), *no deforestation* (green)

For the automated classification, the first 7 multi-spectral bands of the georeferenced Landsat 8 satellite images are used, having a GSD of $30\,m$. All images were normalised such that after normalisation each band has a mean value of zero and a standard deviation of one calculated over the respective image.

In pre-processing, the reference data for yearly deforestation increments and past deforestation were rasterised with a resolution of $30\,m$, resulting in maps that are aligned with with the images of the test sites. Furthermore, following (Soto et al., 2021), a padded version of the reference was created, to be used for supervised training, validation and testing. Here, pixels that have a Euclidean distance of less than $2\,px$ to the closest boundary of an area that is labelled as *deforestation* in the reference are marked as *unknown*.

The data from each domain is tiled and split into three subsets for training, validation and testing according to the scheme that was used in (Soto et al., 2021). An overview of the three test sites and of the split into the subsets is provided in Figure 5.4.

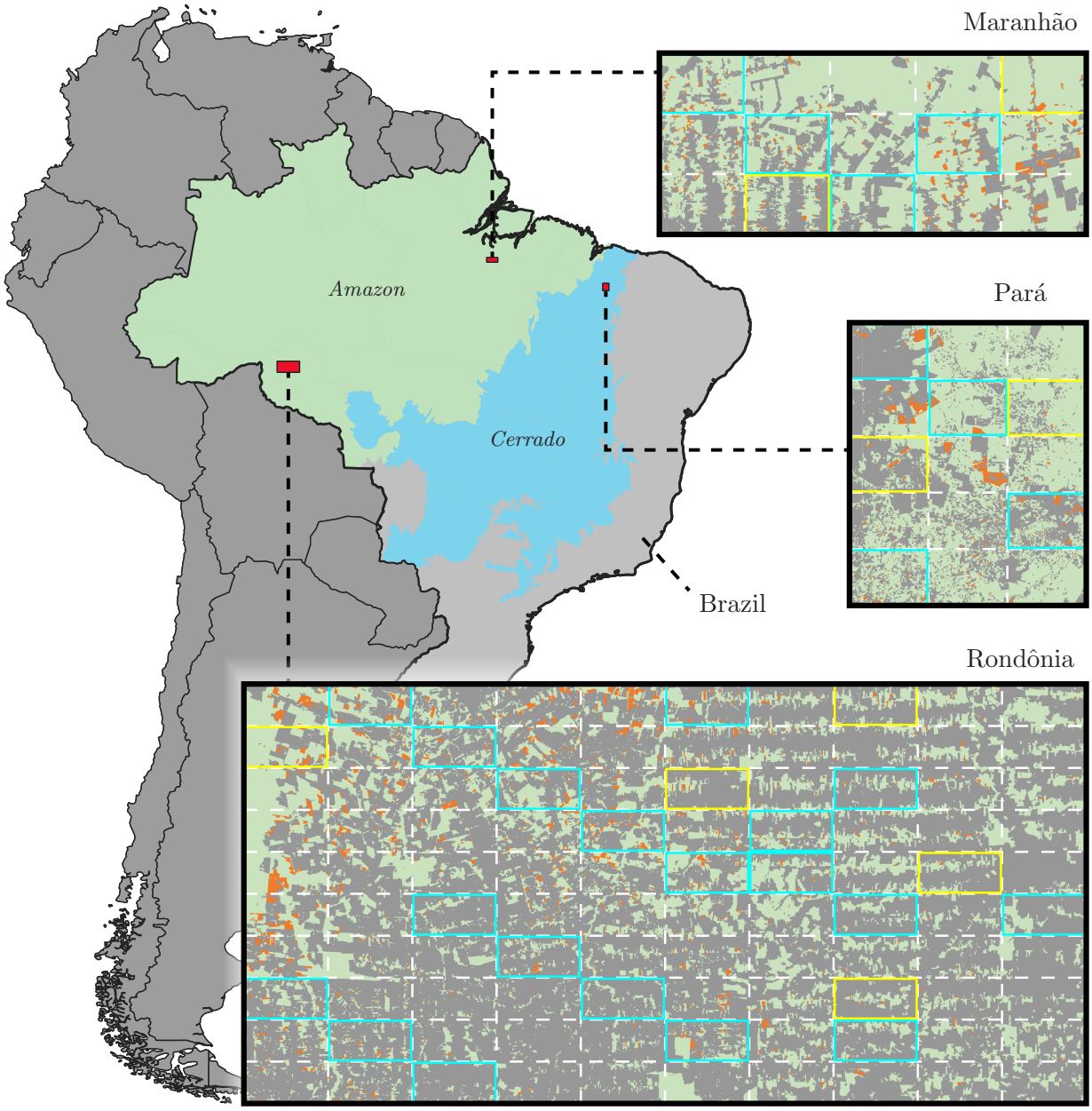

Figure 5.4: Test sites for deforestation detection; the figure shows the locations of the three test sites and an overview of the rasterized reference data. Colour code for the reference: *no deforestation* (green), *past deforestation* (grey), *deforestation*(orange). The broken lines show outlines of the tiles that were created to split the images into the subsets for training, validation and testing. Colour-code for tiling: Training (cyan), validation (yellow), testing (no colour).

The original version of the dataset for the area of Rondônia used in (Soto et al., 2021) has badly aligned reference data due to a mistake in the rasterisation process. To enable a fair comparison of the two methods, this version of the dataset, referred to as RO_0 is used in the corresponding

experiments. However, in the remaining experiments, a corrected version, referred to as *RO* is used. Table 5.4 provides information about the size of each dataset and about the corresponding class distribution.

Dataset		*MA*	*PA*	RO_0	*RO*
Size [M px]					
Training		0.7	0.8	2.5	2.5
Validation		0.3	0.4	0.7	0.7
Testing		1.5	1.7	9.9	9.9
All		2.5	2.9	13.1	13.1
Class distribution [%]					
Deforestation	(DF)	1.4	1.3	1.1	1.1
No deforestation	(ND)	55.4	64.2	26.1	28.3
Unknown	(UK)	43.2	34.6	72.8	70.6

Table 5.4: Dataset overview; Class distribution: percentage of pixels assigned to the corresponding class in the corresponding region in the padded reference.

5.2 Evaluation and Quality Metrics

To assess the performance of a classifier on a set of images, the predictions are compared to the reference labels. During inference, the predictions for images that are larger than the patch size are obtained in a sliding-window based approach. This means that the images are processed by slicing them into horizontally and vertically overlapping patches of size $p \times p$ and processing the patches by the network. It is pointed out that the patch size p used in the inference is the same that is used during training. Using the sliding window based approach, one obtains redundant predictions for the pixels in the overlapping areas. The probabilistic class scores predicted for different patches that correspond to the same pixels are averaged before the class with the largest probability is identified and returned as final class prediction. In all experiments, the overlap of the patches during evaluation is half of the patch size.

Based on the predicted label maps, a confusion matrix is determined by comparing each pixel in the predictions to the each pixel in the reference label maps. From the confusion matrix, the number of pixels which correspond to true positive (TP_k), false positive (FP_k) and false negative (FN_k) predictions are derived for each class $L_k \in \mathbb{L}$.

For each class, the F_1 score $F_{1,k}$ (cf. Equation 4.2) is computed. As global metrics, the overall accuracy OA, i.e. the percentage of correct class assignments, the mean F1-score $\bar{F}_1$ is reported, where the mean is taken over all classes. Some experiments are repeated several times to assess the influence of random components such as the parameter initialisation on the result. In general, only the average performance metric and the corresponding standard deviation are reported.

In order to quantitatively assess the semantic consistency a strategy is proposed that is motivated by the following line of thought. A pair consisting of an input image X^S and the corresponding

adapted image X^{ST} is considered to be semantically consistent if the original label map Y^S still matches the scene depicted in X^{ST} in which the objects imitate the style of the target domain. Thus, when X^{ST} is presented to a perfect classifier for D^T, the predicted label map $\hat{Y}^{ST}$ should correspond to Y^S. A perfect classifier is not available, but a classifier $\mathcal{C}^T$ trained in D^T is assumed to be sufficient in order to assess the consistency. In particular, it is proposed to quantitatively assess the semantic consistency by the agreement of the predictions made by $\mathcal{C}^T$ for adapted images, measured using the quality metrics introduced above.

5.3 Goals and Structure of Experiments

To investigate different aspects of the proposed method several experiments are conducted, grouped into five experiment sets (E1 - E5). The first four sets consider the application of land cover classification and the last one evaluates the proposed approach for deforestation detection. In the following paragraphs, an overview of the experiments is provided. Further details are given in the subsequent sections.

E1) Source Training and Naïve Transfer: In the first experiment, the source training is performed for each domain in two settings optimised either for intra-domain (ID) or for cross-domain (CD) evaluation, respectively. In the ID scenario, classifiers are trained in a supervised fashion on each domain and evaluated on the same domain. The performance of these classifiers serves as a reference to assess any approach for UDA. In particular, it is assumed that supervised training always outperforms UDA, which is the reason why the respective performance is considered to be an upper bound for the performance after UDA. In the CD scenario, classifiers are trained on each (source) domain and evaluated on every other (target) domain without adaptation. This approach is also referred to as *naïve transfer*. The performance of naïve transfer serves as a sort of lower bound to assess the performance of any approach for UDA. In particular, if in later experiments, the performance after UDA is better than the performance of naïve transfer, this will be considered to be a positive transfer, otherwise it will be considered a negative transfer.

E2) Proposed Method for UDA: Next, different variants of the proposed method for UDA are evaluated using the domains from land cover classification to construct multiple scenarios for UDA. This experiment should answer the first scientific question (cf. Section 1.2):

Do the different variants of the proposed method for UDA achieve a stable positive transfer when adapting between different domains? By how much can the performance gap be reduced and where are the limitations?

Furthermore, the proposed methods for improved semantic consistency are evaluated. This comparison aims at answering the second scientific question:

Is the joint training scheme of the classifier and the appearance adaptation network sufficient to achieve semantically consistent adaptations? Do the proposed methods for improving semantic

consistency actually result in a higher semantic consistency and if so, does this also lead to a higher classification performance after UDA?

Lastly, the combination of the proposed strategy for UDA and adaptive batch normalisation is evaluated to answer the third scientific question:

Can the performance of UDA be improved further by combining the proposed method for UDA based on appearance adaptation with adaptive batch normalisation?

Based on the results of this set of experiments the best performing variants are selected and only those are used in the remaining experiments.

E3) Evaluation of Parameter Selection: In this set of experiments, the parameter selection criterion is evaluated for the application of land cover classification by comparing it to the naïve strategy of performing the adaptation for a fixed number of epochs. This aims at answering the fourth scientific question (cf. Section 1.2):

Does the proposed parameter selection method based on the entropy in the target domain lead to better results compared to running the UDA process for a fixed number of epochs?

E4) Comparison to other Strategies and Methods: In the fourth set of experiments, the best performing variants identified in the experiment set E2 are evaluated by comparing them to other adaptation strategies and existing methods from literature. First, the architecture and the training scheme are changed to follow different adaptation strategies, one being instance transfer according to (Vu et al., 2019) and the other being adversarial representation transfer according to (Tsai et al., 2018). The best performing variants resulting from experiment set E2 are compared against these strategies. Second, the proposed variants are compared against three methods from literature which were evaluated on public datasets for aerial image classification, which allows for a direct comparison of the final performance by applying the method to the same adaptation scenarios.

E5) Evaluation of UDA for Bi-temporal Deforestation Detection: In the last set of experiments, the method is evaluated on the task of bi-temporal deforestation detection using satellite imagery. This experiment allows to assess the transferability of the proposed method to other applications. Furthermore, the method will be compared to a method for UDA from literature that was evaluated for deforestation detection using the same datasets.

Consequently, the goal of the experiment sets E4 and E5 is to answer the fifth scientific question posed in Section 1.2:

Does a variant of the proposed method outperform approaches from the literature for UDA when evaluated on the applications of land cover classification and bi-temporal deforestation detection?

5.3.1 Experiment Set E1: Source Training and Naïve Transfer

In the first set of experiments, no UDA is used, but only supervised training in different variants is performed. This type of training in a single domain is referred to as *source training*. In particular, classifiers are (source) trained on the domains P_{20}, V_{20}, B, H, N. The domains S and Hm are not used for evaluation, because they are used for tuning the hyper-parameters (cf. Section 5.4). Two variants of source training (ST) are considered.

In the first variant, classifiers are trained on the training set of one domain and the validation set of the same domain is used for early stopping (cf. Section 5.4.1). The resulting classifiers are evaluated on the test set of the respective domain, resulting in the so called intra-domain performance. The intra-domain performance in a domain is considered to be an upper bound for the performance of a classifier that was trained in another domain and adapted to this domain using UDA. This variant of ST is referred to as *ST for intra-domain evaluation*.

In the second variant, classifiers are trained on the union of training set and test set in each (source) domain, again using the respective validation set for early stopping. However, the classifiers resulting from this setting are not evaluated in the domain they were trained on but instead on the test set of all other (target) domains. The reason for using the joint set in D^S for training is that these classifiers should maximise the performance in the respective target domains and it is assumed that using more training data could help in this regard, even more so because some datasets are rather small. This variant of ST is referred to as ST *for cross-domain evaluation*. The strategy of training a classifier in D^S and apply it to D^T without adaptation is referred to as *naïve transfer* (NT) and the respective performance of the classifier in D^T is called *naïve cross-domain performance*. The naïve cross-domain performance for a specific pair of D^S and D^T serves as a lower limit when assessing the performance of UDA in that scenario. In particular, if the performance of a classifier after UDA is worse than the naïve cross-domain performance, this will be considered a negative transfer, otherwise it will be considered a positive transfer. Besides assessing the naïve cross-domain performance, the parameters of the classification network resulting from the cross-domain training scenario are also used for initialisation of $\mathcal{C}$ in the UDA.

All classifiers are trained five times using a different random initialisation of the weights and a different random sampling of the training batches to assess the influence of these random components on the results. Note that the encoder of the segmentation network is always initialised by the weights after pre-training on the ImageNet dataset (cf. Section 4.3.1).

5.3.2 Experiment Set E2: Proposed Method for UDA

In the second set of experiments, the classifiers that were trained in the source domain for cross-domain evaluation in experiment set E1 are adapted to the other domains using several variants of the proposed method for UDA. Again, the domains S and Hm are not used for evaluation in this experiment because they were involved in the hyper-parameter tuning process (cf. Section 5.4.2). In the adaptation process, the full datasets are used for D^S and D^T, but the resulting classifiers are evaluated only on the test set of D^T in order to compare them to the results from the first experiment set.

In the first variant, referred to as V_{SEP}, the loss term $\mathcal{L}_{sup}^{ST}$ is not used to train the appearance adaptation network $\mathcal{A}$. Consequently, $\mathcal{A}$ is only trained by minimising the adversarial loss $\mathcal{L}_{advA}$. Furthermore, none of the methods aiming at an improved semantic consistency are applied. This variant is used as a baseline to investigate the benefits of the joint training scheme which is crucial to use $\mathcal{L}_{sup}^{ST}$ for training $\mathcal{A}$.

The second variant, referred to as V_{BSLN}, uses the proposed joint loss for $\mathcal{A}$ and $\mathcal{C}$. Like in the first variant, none of the methods aiming at an improved semantic consistency are applied. Thus, the difference to V_{SEP} is that $\mathcal{A}$ is also trained to minimise $\mathcal{L}_{sup}^{ST}$. By comparing the performance of the classifiers adapted using this variant to the ones adapted using variant V_{SEP} the benefit of the joint training scheme can be assessed. Further, this variant serves as a baseline for the proposed modifications aiming at an improved semantic consistency.

In the next two variants, the proposed modifications for improved semantic consistency are applied. V_{RD} denotes the method in which the variability in the output of $\mathcal{D}$ is regularised (cf. Section 4.5.1) and V_{AG} corresponds to the variant in which the auxiliary generator $\mathcal{G}$ is used (cf. Section 4.5.2).

Lastly, the combinations of the variants V_{RD} and V_{AG} with a strategy for representation transfer, adaptive batch normalisation (ABN) (Li et al., 2018), are evaluated. These variants are denoted by $V_{RD,ABN}$ and $V_{AG,ABN}$, respectively.

In this experiment set, first a qualitative and quantitative evaluation of the appearance adaptation is presented before the evaluation of the performance after using the different variants of the proposed method for UDA is done. To assess the semantic consistency quantitatively, the images from the validation set of the respective source domains are adapted to the target domain using $\mathcal{A}$. Then, the adapted images are classified using the classifier that was trained in D^T using the variant of source training for cross-domain evaluation. The resulting predictions are compared against the reference, assuming that a semantically consistent appearance adaptation leads to a proper classification (cf. Section 5.2). Afterwards, the performance of the adapted classifiers on the test set of the target domain is evaluated.

Like in the previous set of experiments, all adaptation scenarios are repeated five times using the parameter sets of the classifiers after source-training for cross-domain evaluation from experiment set E1 as initialisation. Based on the results of this experiment set, final variants will be suggested and only those variants will be used in the remaining experiments.

5.3.3 Experiment Set E3: Evaluation of Parameter Selection

This set of experiments focusses on an evaluation of the suggested parameter selection method. In particular, the experiments should help to evaluate whether this method leads to a superior performance compared to the naïve approach of performing the UDA for a fixed number of iterations. To that end, the best variant of the method resulting from the experiment E2 that does not use ABN is used to adapt classifiers that were trained on P_{20}, V_{20}, B, H and N, serving as source domains, to all other domains using a fixed number of epochs. After each training epoch, the cross-domain

performance is evaluated and compared to the cross-domain performance which is achieved using the parameter selection approach proposed in this thesis. Such a comparison will reveal if the parameter selection approach results in a better performance after UDA compared to using a fixed number of epochs. Furthermore, the impact of the number of epochs itself on the performance after UDA can be assessed.

5.3.4 Experiment Set E4: Comparison to other Strategies and Methods

In this set of experiments, the method is compared to several UDA strategies (E4.1) and methods (E4.2) from literature.

5.3.4.1 Experiment Set E4.1: Comparison to other Strategies

In a first set of experiments, the best performing variants of the method selected in experiment set E2 are compared to three different strategies for UDA. To assess the influence of the main adaptation strategy, other aspects such as the architecture for the classification network, the loss function for the supervised loss term and the parameter selection based on the entropy in D^T are not modified. The first two strategies to compare to are adapted from methods from literature, one proposing variants of instance transfer (Vu et al., 2019) and the other one proposing variants of adversarial representation transfer (Tsai et al., 2018). These two methods were selected because they are considered to be rather representative approaches for the respective strategy. The third strategy against which the method proposed in this thesis is compared to corresponds to applying ABN to the classifiers after source training. Details about the three strategies are presented in the next paragraphs.

I) Instance Transfer: Vu et al. (2019) propose a method based on instance transfer. They combine implicit instance transfer by direct entropy minimisation (EM) for images from D^T and indirect EM via adversarial training, in which a discriminator network is trained to discriminate entropy maps into those coming from from D^S from those from D^T. The authors propose three variants. The first one uses only direct EM, the second one uses indirect EM and the third one uses a combination of direct and indirect EM. The three variants are adapted to the proposed architecture by replacing the loss terms accordingly while keeping the remaining aspects unchanged to enable a comparison of the main strategy for UDA. In particular, the pre-training scheme, the architecture of the classification network and the parameter selection criterion correspond to the method proposed in this thesis. Details about how the strategy is adapted are given in Section 5.4.3.

II) Representation Transfer: A method based on representation transfer is presented in (Tsai et al., 2018). The authors apply the basic concept of adversarial representation transfer, in which a discriminator network is trained to distinguish latent representations coming from D^S and those from D^T. In order to match the representations for samples from both domains, the classifier is trained to deceive the discriminator by producing representations for target domain images that look like coming from D^S. Again, the pre-training scheme, the architecture of the classification

network and the parameter selection criterion correspond to the ones of the method proposed in this thesis, but only the loss used during domain adaptation is changed to match (Tsai et al., 2018). Details about how the strategy is adapted are given in Section 5.4.3.

III) Adaptive Batch Normalisation: This strategy corresponds to the application of adaptive batch normalisation (Li et al., 2018) to the classifiers obtained by source training for cross-domain evaluation. This method was described in Section 4.7.

5.3.4.2 Experiment set E4.2: Comparison to other Methods

For a comparison of the proposed method to those from literature, three methods that address the application of land cover classification are selected that were evaluated on the basis of the data of the ISPRS Semantic Labelling challenge, i.e., P'_5 and V'_9, using P'_5 as D^S and V'_9 as D^T. The best performing variants of the method selected in experiment set E2 are compared quantitatively by applying them to the same adaptation scenarios and comparing the resulting quality metrics. The specific settings and evaluation protocols are described in the following two paragraphs.

I) (Benjdira et al., 2019): This method is based on source-to-target appearance adaptation using CycleGAN and was evaluated by adapting from P'_5 to V'_9. Benjdira et al. (2019) use the `RGB` data in P'_5 and `IRG` in V'_9. Note that according to the interpretation of the criterion for homogenous UDA in Section 4.1 this setting is considered to correspond to heterogeneous UDA. To apply the proposed method to this adaptation scenario, the `RGB` data from Potsdam are downsampled to P'_9 to be used as D^S, and V'_9 is used as D^T (cf. Section 4.8).

II) (Ji et al., 2020): This method uses a hybrid approach for UDA and was also evaluated by adapting from P'_5 to V'_9, but the authors use `IRG` data in both domains. Ji et al. (2020) resample the data from P'_5 to P'_{10} before adapting to V'_9, which is similar to the proposed strategy for dealing with different GSDs.

III) (Zhao et al., 2023): Similar to (Benjdira et al., 2019), this method also uses source-to-target appearance adaptation using CycleGAN. However, Zhao et al. (2023) extend the training scheme of CycleGAN by an auxiliary task, which is to predict the height maps during UDA, aiming at improving the semantic consistency. Similarly to the strategy for dealing with different GSDs proposed in this thesis, they perform an resampling of the data from D^S to the GSD from D^T. However, they do not resample the input images to the appearance adaptation network, but instead the adapted images. The method is evaluated in the same adaptation setting that was used in (Ji et al., 2020), i.e. by adapting from P'_5 to V'_9, using `IRG` data in both domains.

5.3.5 Experiment set E5: Evaluation of UDA for Bi-temporal Deforestation Detection

In the last set of experiments, the best performing variants from experiment E2 are evaluated on adaptation scenarios for the task of bi-temporal deforestation detection. Note that the hyper-parameters of the method are slightly changed (cf. Section 5.4).

In the first part, the datasets PA, MA and RO are used. For each of the domains PA, MA and RO, source-training is performed by training on the corresponding training subset and using the corresponding validation set for early stopping and parameter selection (cf. Section 5.4.1). The classifiers are evaluated on the test set of the respective domains they were trained on, but also on the test sets of the respective other domains. Note that in this experiment set no source training optimised for cross-domain adaptation is performed, i.e. no classifiers that were trained on the union of the subset for training and validation are evaluated. Each experiment is repeated five times to asses the influence of random factors such as the random initialisation of the parameters on the performance on the classifier.

Next, the proposed method for UDA using the best performing variants resulting from the experiment set E2 are used to adapt the classifiers to the respective other domains. The performance on the target domains after UDA is compared to the performance in the intra-domain and cross-domain settings to assess the performance of the proposed method for the application of deforestation detection.

In the second part of this experiment set, the variant of the method that performs best in this application is compared to the one proposed in (Soto et al., 2021). To this end, the datasets PA, MA and RO_0 are considered. The results of source-training and UDA are presented and compared to those reported in (Soto et al., 2021).

5.4 Training Details and Hyper-parameters

In this section, the hyper-parameters of the method and further training details are described. The hyper-parameters related to the source training are discussed in Section 5.4.1 and those of the domain adaptation in Section 5.4.2. In Section 5.4.3, details about the implementation of instance transfer and representation transfer against which the proposed method is compared to in the experiment set E4 are provided.

5.4.1 Source Training

Most of the hyper-parameters for source training were tuned empirically by performing source training with different hyper-parameters on the domains S and Hm and searching for the variant that achieves the highest average cross-domain performance. In particular, for each variant, two classifiers were trained on the training sets of both domains, using the corresponding validation sets for early stopping. The resulting classifiers were evaluated on the test set of the respective other

domain, and the performance, here measured by the mean F_1 score, was averaged. Some further hyper-parameters were set on the basis of related work or preliminary experiments.

For both applications, source training is performed for a maximum number of $n_E^{(S)} = 100$ epochs, where an epoch is defined as consisting of $n_{iter} = 2500$ training iterations. Training is stopped earlier if the performance on the respective validation set does not increase for $n_{es} = 25$ epochs. The final parameter set is the one that yields the highest performance on the validation set. For aerial image classification, the mean F_1 score is used to assess the performance and for the deforestation detection, the F_1 score of the class *deforestation* is used. Mini-batch stochastic gradient descent with a friction of $\beta_0 = 0.9$ is used as optimisation strategy and the weight ω_{L2} of weight decay is set to $1e^{-5}$. The hyper-parameter κ in Equation 4.3 of the adaptive cross entropy loss (cf. Section 4.4.1) is set to 4. The training batches consist of patches that were randomly cropped from the images and the reference of the training set after applying a random rotation for data augmentation. In particular, the rotation angle is drawn from a uniform distribution in the range $[0°, 360°]$. The image data is interpolated using bi-linear interpolation and the corresponding label maps using nearest neighbour interpolation. Further the extracted patches are flipped along the main diagonal with a probability of 50 %. For additional data augmentation, each channel in the training images is multiplied by a random value drawn from a normal distribution centred at $\mu = 1.0$ and having a standard-deviation of $\sigma = 0.1$, and each channel is shifted by adding a random value drawn from a normal distribution with $\mu = 0$ and $\sigma = 0.1$.

The remaining hyper-parameters and configurations depend on the application. For the application of land cover classification the learning rate λ is set to 0.01, the mini-batch size n_B is set to 4 and the patch size is set to $p = 256\,px$. This corresponds to the optimisation parameters that were used to pre-train the encoder of the classifier on the ImageNet dataset (Chollet, 2017).

For the bi-temporal deforestation detection the smaller classification architecture is used (cf. Section 4.3.1). In comparison to the version used for land cover classification, the central layers were removed. This is motivated by preliminary experiments in which only a very minor drop of performance was observed when using the smaller network for a similar task, the pixel-wise classification of the vitality loss based on a pair of satellite images (Wittich et al., 2022). Following (Soto et al., 2021), the patch size is set to $p = 128\,px$. To have the same number of pixels in a batch, the batch size is increased to 16. The learning rate is decreased to 0.002, assuming that this partially compensates for the potential optimisation problems due to the sparsity of the usable reference pixels during training. In particular, less pixels contribute to the loss which is likely to result in more noisy gradients. The strategy for the extraction of patches for training and data augmentation are the same as those for land cover classification. An overview of all hyper-parameters is given in Table 5.5.

Parameter	Symbol	Value (LCC)	Value (BDD)	cf.
Early stopping patience	n_{es}	25 epochs		Sec. 4.4.1
Max. number of epochs	$n_E^{(S)}$	100 epochs		
Param. of adaptive CE loss	κ	4		Eq. 4.3
Optimiser for $\mathcal{C}$	-	MB-SGD-M		Sec. 2.2.2.1
Learning rate	λ	.01	.002	Eq. 2.14
Friction parameter	β_0	0.9		Eq. 2.13
Weight of L2-regularisation	ω_{L2}	10^{-5}		Eq. 4.1
Batch size	n_B	4	16	Sec. 2.2.2.1
Patch size	p	256	128	Sec. 2.4
Simplified network	-	No	Yes	Sec. 4.3.1
(Network parameters)	-	28.8 M	15.5 M	
Strength of radiometric augmentation	σ	0.1		This section

Table 5.5: Hyper-parameters for source training for the two applications tested in the experiments. LCC: Land cover classification. BDD: Bi-temporal deforestation detection.

5.4.2 Unsupervised Domain Adaptation

The hyper-parameters for the domain adaptation were tuned by adapting classifiers from S to Hm and vice-versa. To this end, classifiers were first trained on S and Hm as D^S and the final parameters after source training were used for the initialisation of the classifiers in the UDA phase. The classifiers were then adapted to the corresponding other domain, using different hyper-parameters for the adaptation. The performance of the models after UDA was compared to the initial cross-domain performance without UDA, using the mean F_1 score as quality metric. The set of hyper-parameters resulting in the best performance of UDA, averaged over the two scenarios, was maintained.

In the tuning process, first, the variant V_{RD}, using the regularisation of the discriminator output, is considered. In a subsequent step, the variant V_{AG}, using the auxiliary generator, was evaluated to tune parameters related to the optimisation of $\mathcal{G}$ and the hyper-parameter n_G (cf. Section 4.5.2). In a last step the hyper-parameters related to the ABN were tuned.

The patches for D^S and D^T are sampled from all available data in the respective domains. After each epoch starting from epoch $n_{min} = 3$, the average entropy in the test set of the target entropy is tracked. The first epochs are not considered because it is assumed that the appearance adaptation network will not deliver any meaningful adapted images in the beginning. After running the adaptation for $n_E = 50$ epochs, the parameter set that results in the smallest mean entropy in $D^T$is kept as final parameter set. When sampling the batch for the images from D^S, the strategy for data augmentation is equivalent to the one used in source training. For the images from D^T, only the geometric augmentation is performed when sampling the patches. Radiometric augmentation is applied to the adapted images before they are presented to the networks $\mathcal{D}$ and $\mathcal{C}$ and, in variant V_{AG}, to the images generated by the auxiliary generator before they are presented to $\mathcal{D}$.

ABN is applied with the following hyper-parameters. Batches of $n_{B,ABN} = 64$ samples from D^T are presented to $\mathcal{C}$ and the running averages of the batch normalisation (BN) layers are updated. The batch size is increased to better approximate the statistics from D^T. Note that using such a large batch size compared to the one used in source-training and adaptation is only possible because during ABN only the network $\mathcal{C}$ is updated and, thus, the other networks, $\mathcal{A}$, $\mathcal{D}$ and possibly $\mathcal{G}$, are no longer required, which drastically reduces the memory footprint. To adapt the batch statistics from D^T, $n_{I,ABN} = 1000$ randomly drawn batches are presented to $\mathcal{C}$. Note that practically ABN may converge after much fewer iterations, but the training time of ABN is comparably short, which is the reason why this conservative approach was chosen.

Table 5.6 lists all hyper-parameters of the adaptation process. All hyper-parameters related to training the classification network $\mathcal{C}$ are equal to those from source training (cf. Section 5.5).

Parameter	**Symbol**	**Value**	cf.
Optimizer for $\mathcal{A}, \mathcal{D}, \mathcal{G}$	-	ADAM	Sec. 2.2.2.1
Learning rate for $\mathcal{A}, \mathcal{D}, \mathcal{G}$	λ	$5 \cdot 10^{-4}$	Eq. 2.18
Parameter 1 for $\mathcal{A}, \mathcal{D}, \mathcal{G}$	β_1	0.5	Eq. 2.15
Parameter 2 for $\mathcal{A}$	β_2	0.99	Eq. 2.16
Parameter 2 for $\mathcal{D}, \mathcal{G}$		0.999	
Weights of loss terms	ω_T, ω_G, ω_A	2, 2, 2	Eq. 4.5, 4.16
Weight of $\mathcal{D}$ regularisation	ρ	4	Eq. 4.14
Batch size for generated images	n_G	8	Eq. 4.17
ABN batch size	$n_{B,ABN}$	64	Sec. 4.7
Number of ABN iterations	$n_{I,ABN}$	1000	This section

Table 5.6: Hyper parameters for UDA for the two applications tested in the experiments.

5.4.3 Implementation Details of Baseline Strategies

In the experiment set E4, the suggested method for UDA is compared to different adaptation strategies for UDA from the literature. In particular, several variants for UDA are evaluated to which the proposed method is compared to. The first variants are adapted from (Vu et al., 2019) and the others from (Tsai et al., 2018).

Instance Transfer: In (Vu et al., 2019), instance transfer by direct and indirect entropy minimisation is proposed. This is realised by formulating the following adaptation loss for $\mathcal{C}$:

$$\mathcal{L}_{Vu} = \mathcal{L}_{sup} + \lambda_{ent} \cdot \mathcal{L}_{ent} + \lambda_{adv} \cdot \mathcal{L}_{adv,T}. \tag{5.1}$$

Vu et al. (2019) use the regular softmax cross-entropy loss to model the supervised loss $\mathcal{L}_{sup}$. In the experiments reported in this thesis the adaptive cross-entropy loss as described in Equation 4.4 is used, because this choice is not related to the main adaptation concept. The second term, $\mathcal{L}_{ent}$, corresponds to the direct entropy loss as proposed in (Vu et al., 2019), thus $\mathcal{L}_{ent} = \bar{E}$, where $\bar{E}$ is the average entropy in the target domain according to Equation 4.21. This loss term is

weighted by λ_{ent}. The third term corresponds to the approach of adversarial entropy minimisation, involving a domain discriminator $\mathcal{D}_{Vu}$. This network takes the so-called weighted self-information maps $M_{SI} = -r \cdot log\, r$ as input, where r is a map of class scores for a sample X predicted by the classification network $\mathcal{C}$. Each pixel in the output of $\mathcal{D}_{Vu}$ is interpreted as the probability for the corresponding window according to the receptive field of $\mathcal{D}_{Vu}$ to come from D^S. Vu et al. (2019) propose to use the common strategy of alternating training $\mathcal{C}$ and $\mathcal{D}_{Vu}$, which is adopted without further changes in this thesis. $\mathcal{D}_{Vu}$ is trained to correctly predict whether a weighted self-information map comes from D^S or not and $\mathcal{C}$ is trained to deceive the discriminator by maximising the probability predicted by $\mathcal{D}_{Vu}$ for a weighted self-information map obtained for a sample from D^T to come from D^S. The architecture of $\mathcal{D}_{Vu}$, the hyper-parameters related to the optimisation of $\mathcal{D}_{Vu}$ and the hyper-parameters $\lambda_{ent} = \lambda_{adv} = 1e^{-3}$ of the method are taken from (Vu et al., 2019) without modifications. As also done in (Vu et al., 2019), two further variants are evaluated, obtained by setting either $\lambda_{ent} = 0$ or $\lambda_{adv} = 0$. Note that in some experiments in (Vu et al., 2019), an additional class ratio prior is used to regularise the classifier. Variants including such a prior are not evaluated in this thesis.

Representation Transfer: Tsai et al. (2018) propose a variant for UDA based on adversarial representation transfer. In particular, they use the following loss for $\mathcal{C}$:

$$\mathcal{L}_{Tsai} = \mathcal{L}_{sup} + \lambda_e \cdot \mathcal{L}_{adv,e} + \lambda_l \cdot \mathcal{L}_{adv,l}. \tag{5.2}$$

Note that in (Tsai et al., 2018), a further variant is evaluated in which an auxiliary classification loss is considered. This variant is not considered in the experiment reported here. Again, the first term $\mathcal{L}_{sup}$ is the supervised loss, which corresponds to the one that is used in the method proposed in this thesis (cf. Equation 4.4). The second and third terms correspond to the adversarial representation alignment and are taken without modification from (Tsai et al., 2018). The terms correspond to two variants of adversarial training using a layer in the middle of the network ($\mathcal{L}_{adv,e}$) and the last layer of the network ($\mathcal{L}_{adv,l}$). These terms are weighted by λ_e and λ_l, respectively. For the adversarial training, two discriminator networks are introduced. The first one, called $\mathcal{D}_{Tsai,e}$, takes the activation maps from the layer 6 in Table 4.1 as input, and the second one, $\mathcal{D}_{Tsai,l}$, takes the activation maps from the last layer, which is layer 32 in Table 4.1. The discriminator architectures, the hyper-parameters related to the training of the discriminator networks and the alternating training strategy of $\mathcal{C}$, $\mathcal{D}_{Tsai,e}$ and $\mathcal{D}_{Tsai,l}$ are used as described in (Tsai et al., 2018). The authors suggest two variants with different weights for the adversarial loss terms in Equation 5.2. In the first variant, the weights are set to $\lambda_l = 1e^{-3}$ and $\lambda_e = 0.0$, which corresponds to applying adversarial representation transfer only to the last layer. In the second variant, the weights are set to $\lambda_l = 1e^{-3}$ and $\lambda_e = 2e^{-4}$, so that the representation transfer is applied to the earlier layer, too.

6 Results and Discussion

In this chapter, the results of the experiments are reported and discussed. The structure of the experiments corresponds to the structure presented in Section 5.3.

In this chapter, the term *adaptation scenario* corresponds to a combination of source and target domains, and the shorthand $D^A \rightarrow D^B$ is used to refer to an adaptation scenario with source domain D^A and target domain D^B. The term *intra-domain scenario* refers to an adaptation scenario in which $D^S = D^T$ and the term *cross-domain scenario* refers to an adaptation scenario in which $D^S \neq D^T$. Lastly, the term *naïve transfer* describes the strategy of training a classifier in D^S and applying that classifier to D^T directly, thus, without applying any method for UDA.

6.1 Results of Experiment Set E1: Source Training and Naïve Transfer

Source Training: First, the results of source training for the intra-domain evaluation are reported. Table 6.1 provides means and standard deviations of the $\bar{F}_1$ score obtained for five training runs on each domain. Note that the standard deviations for the values that correspond to the average performance does not represent the variability of the performance in the different domains, but only the variability due to random factors of training.

D^S	B	H	N	P_{20}	V_{20}	Avg.
$\bar{F}_1$ [%]	87.1±0.2	80.3±0.2	85.5±0.2	89.1±0.1	84.3±0.2	85.3±0.2
OA [%]	88.6±0.1	86.4±0.1	88.7±0.1	89.8±0.1	86.9±0.1	88.1±0.1

Table 6.1: Results of source training for intra-domain evaluation. Performance on the test set of each source domain D^S when training on the training set from D^S and using the validation set from D^S for parameter selection. In the last column, the average performance over all domains is given. The table provides means and standard deviations of the $\bar{F}_1$ scores and overall accuracies resulting from five repetitions of each experiment.

For all domains, $\bar{F}_1$ scores and overall accuracies between 80 % and 90 % are achieved. The performance on H is worst with $\bar{F}_1 = 80.3\,\%$, which is mainly due to the low classification performance of the class *low vegetation* that has a F_1 score of less than 60 %. This has only a minor impact on the overall accuracy achieved for this domain, because only 8.6 % of the pixels in the test set of H belong to the class *low vegetation*. A reason for this relatively poor performance in H could be that in this domain the difference between training and test sets in this domain is larger compared to the other domains or that there is a higher ratio of errors in the reference label maps.

On the other hand, the intra-domain performance is highest for B and P_{20} with $\bar{F}_1$ scores and overall accuracies close to 90 %. The standard deviations over the five runs are only about 0.2 % in $\bar{F}_1$ and about 0.1 % in OA for all domains, which indicates that the influence of random factors on the source training is relatively low. The performance achieved in each domain will be considered an upper bound for any UDA scenario in which a classifier is adapted to that domain.

Naïve Transfer: The results of naïve transfer are reported in Table 6.2. Here only the means and standard deviations of the $\bar{F}_1$ scores are reported. The last row and column in the table correspond to the results averaged over all source domains and all target domains, respectively. The value in the lower right corner corresponds to the average performance over all scenarios.

D^S \ D^T	B	H	N	P_{20}	V_{20}	Avg.
B	-	61.5±1.2 (-18.9)	80.4±0.6 (-5.0)	70.7±1.4 (-18.4)	81.9±0.2 (-2.4)	73.6±0.8 (-11.6)
H	76.8±1.1 (-10.3)	-	77.1±0.9 (-8.3)	78.5±0.3 (-10.6)	77.7±0.6 (-6.6)	77.5±0.6 (-7.7)
N	82.9±0.8 (-4.2)	63.1±1.7 (-17.3)	-	78.0±0.5 (-11.1)	82.6±0.5 (-1.7)	76.6±0.4 (-8.6)
P_{20}	73.7±0.8 (-13.4)	60.8±1.9 (-19.5)	77.6±0.8 (-7.8)	-	77.2±0.4 (-7.1)	72.3±0.6 (-12.9)
V_{20}	80.1±0.5 (-7.0)	55.9±1.1 (-24.5)	79.0±0.7 (-6.4)	76.2±0.4 (-12.9)	-	72.8±0.5 (-12.4)
Avg.	78.4±0.2 (-8.7)	60.3±1.2 (-20.0)	78.6±0.5 (-6.9)	75.9±0.3 (-13.2)	79.9±0.2 (-4.4)	74.6±0.4 (-10.7)

Table 6.2: Results of source training for cross-domain evaluation. Performance on the test set of each target domain D^T when training on the training and test sets from D^S and using the validation set from D^S for parameter selection (naïve transfer). Each cell provides means and standard deviations of the $\bar{F}_1$ scores in % resulting from five repetitions of the experiments. Values in parentheses are the performance drops, i.e. the differences to the intra-domain performance that was achieved when training and testing on the same domain (cf. Table 6.1). The last row and column provide the performances for each target domain averaged over all source domains and vice versa, respectively. The value in the lower right cell corresponds to the performance averaged over all scenarios.

In general, a rather large performance drop can be observed (-10.7 % in the $\bar{F}_1$ score on average). In several scenarios the performance drop is even larger with values up to -24.5 % for the scenario $V_{20} \rightarrow H$. The smallest performance drop is observed for the scenario $N \rightarrow V_{20}$ with -1.7 %, which is still about 3 times larger than the respective standard deviation of the $\bar{F}_1$ in this setting and, thus, considered to be significant. These observations underline the necessity of methods for UDA.

Table 6.2 also reveals that the scenarios in which $D^T = H$ are particularly difficult, indicated by very large performance drops. This also leads to a rather poor performance of the classifiers evaluated in H after being trained in another domain, i.e. of $\bar{F}_1 = 60.3$ % on average. On the other hand, the scenarios in which $D^T = V_{20}$ result in the smallest performance drops, on average only -4.4 % in the $\bar{F}_1$ score. This observation, i.e. that adapting to V_{20} is particularly easy and adapting to H is comparably difficult, becomes even clearer when assessing the symmetry of the performance drops. For the domains B, N and P_{20} the difference between the average performance

drop when using the respective domain as source domain to the average performance drop when using the same domain as target domain are rather small, i.e. less than 3 % in the $\bar{F}_1$ score. On the other hand, the average performance drop when using H as target domain is −20.0 % while the average performance drop when using H as source domain is only −7.7 % in the $\bar{F}_1$ score. For all combinations of H and any other domain, the scenario in which H is the target domain results in a larger performance drop compared to the corresponding scenario in which H is the source domain. For example $B \rightarrow H$ results in a performance drop of −18.9 % while the scenario $H \rightarrow B$ results in a smaller performance drop of −10.3 % with respect to the $\bar{F}_1$ score. The domain V_{20} behaves exactly in the other way round. The performance drops for all domain pairs involving V_{20} are smaller if V_{20} is used as target domain.

From this observation is is concluded that H includes the other domains to a large extent, i.e. it represents the appearance of objects in the other domains rather well. Consequently a classifier trained in H achieves a relatively good performance when applied to the other domains. On the other hand, the domain V_{20} is rather well represented by the other domains, but V_{20} is not particularly representative for the other domains.

In general, it is difficult to predict the performance drop in the naïve transfer setting for a pair of source and target domains. In the visual assessment of the domains, H was assumed to be rather different from the others in terms of the appearance of objects and the overall appearance of the scene, which is in line with the low performance when using H as target domain. However, it is somehow surprising that H works comparably well when it is used as source domain. The seasonal effects seem to have an impact on the performance drop as well. For example, when training in P_{20}, which is the only domain of the ones evaluated in which the deciduous trees have no leaves, and applying that classifier to V_{20}, this results in the largest performance drop and the lowest performance of all scenarios in which $D^T = V_{20}$. Furthermore, the performance drops are quite high when adapting to P_{20}, which is largely related to the performance drop in the class *high vegetation*. For example, in the setting $N \rightarrow P_{20}$, the performance drop for this class is about −20 % in $\bar{F}_1$, while it is only −11.1 % on average. Again, this is to be expected, because the pattern of a tree without leaves was not learned by the classifiers that were trained in the other domains.

To summarise, the scenarios in which $D^T = H$ result in the largest performance drops (−20.0 % in the $\bar{F}_1$) followed by the scenarios in which there are large seasonal differences, i.e. in which P_{20} is source or target domain (−13.1 % in the $\bar{F}_1$). A smaller, yet comparably high, performance drop is observed if $D^S = H$ (−7.7 % in the $\bar{F}_1$). In the remaining scenarios, i.e. those between the domains B, N and V_{20} the average performance drop is only −4.5 % in the $\bar{F}_1$.

6.2 Results of Experiment Set E2: Proposed Method for UDA

In this set of experiments, the variants of the proposed method for UDA introduced in Section 5.3.2 are evaluated and compared to each other. Table 6.3 provides an overview of all variants of UDA compared in this section, for details cf. Section 5.3.2.

Variant	Use $\mathcal{L}_{sup}^{ST}$ for A	Modification for semantic consistency	Apply ABN
V_{SEP}	No	None	No
V_{BSLN}	Yes	None	No
V_{RD}	Yes	Regularise output of D	No
V_{AG}	Yes	Auxiliary generator	No
$V_{RD,ABN}$	Yes	Regularise output of D	Yes
$V_{AG,ABN}$	Yes	Auxiliary generator	Yes

Table 6.3: Variants for UDA to be evaluated for the task of land cover classification. For detailed descriptions of the variants cf. Section 5.3.2

6.2.1 Evaluation of Appearance Adaptation

First, the variants V_{SEP}, V_{BSLN}, V_{RD} and V_{AG} are compared with respect to the quality of the appearance adaptation to assess which modifications lead to an improvement of the appearance adaptation and to assess how large the respective improvement is. To this end, for several adaptation scenarios the appearance adaptation network with the parameters from the epoch selected using the proposed parameter selection criterion is used to adapt images from D^S to D^T. The adapted images are assessed qualitatively, but also quantitatively by predicting labels for the adapted images using the classifier that was trained in the target domain of the respective adaptation scenario. In particular, the images of the validation set of the source domain are adapted to the target domain and classified using the classifier that was trained in D^T using the the cross-domain source training scheme from experiment set E1. The $\bar{F}_1$ score for these predictions is reported, assuming that images that were adapted in a semantically consistent way should be classified correctly by a classifier that was trained in the respective target domain.

Five adaptation scenarios are selected such that each domain is used once as a source domain and once as a target domain. The first scenario is $B \rightarrow H$. Figure 6.1 shows the results of appearance adaptation achieved by the four variants. The figure also shows the predictions that were made by a classifier trained in the target domain. At a first glance, all adapted images look meaningful, because they give an impression that is similar to the appearance of images in the target domain H (cf. last row in Figure 6.1 for an exemplary image from H). In H, the appearance of the nDSM is not very smooth but rather consists of patches with similar height values and steep height changes in-between. This style is imitated well in the adapted images using V_{BSLN} and V_{RD}, but not very well when using the variants V_{SEP} and V_{AG}. In the adapted nDSMs delivered by the latter two variants, high objects like buildings and high vegetation have sharp outlines and a rather homogenous height, which at least partially imitates the style of nDSMs in H. In the nDSMs related to V_{SEP}, some artefacts are visible that look like holes in high objects (cf. position marked by the red arrow in Figure 6.1).

The target domain H has a much higher amount of *sealed ground* and a lower amount of *low vegetation* compared to the source domain B. Thus, without any countermeasures, the appearance adaptation network has to hallucinate structures that look like *low vegetation* to align the

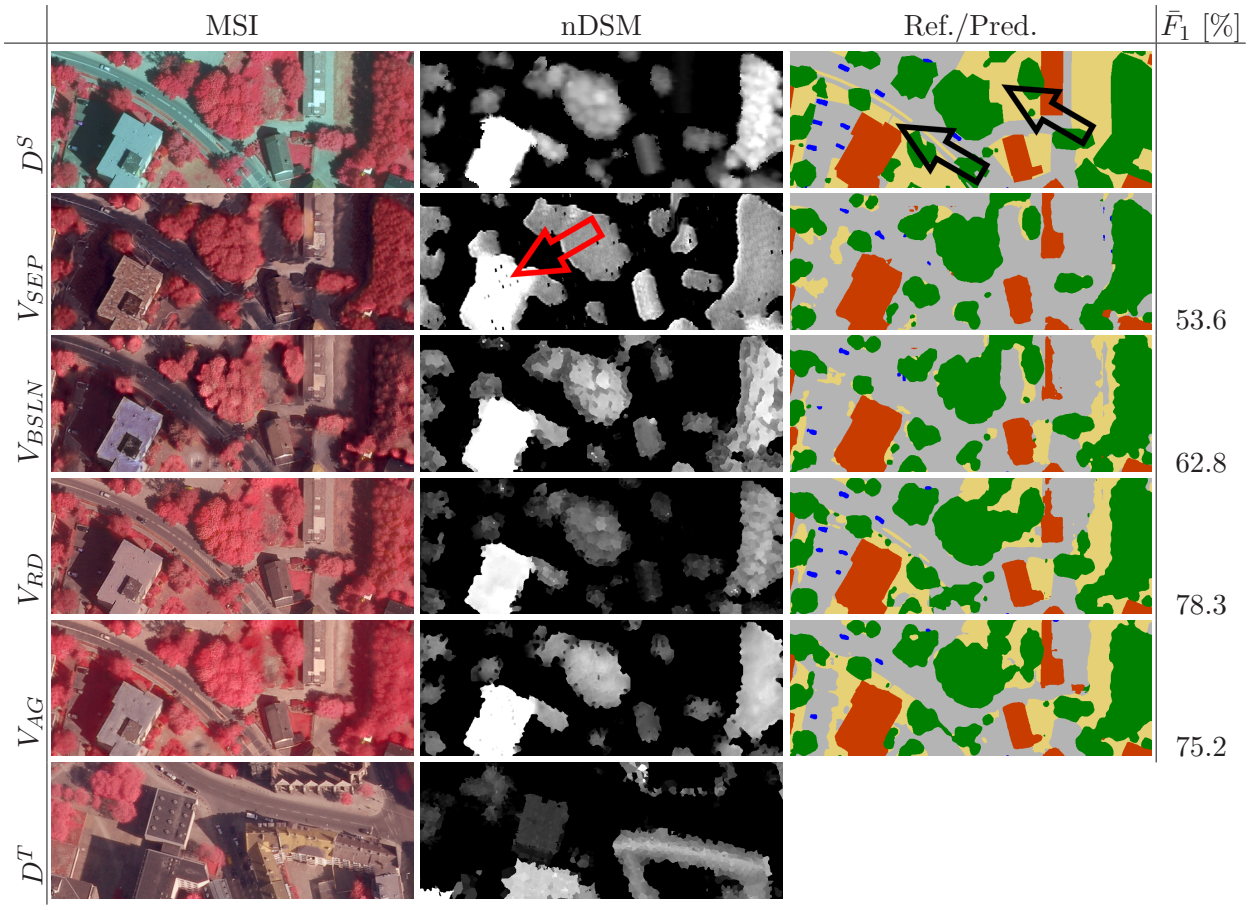

Figure 6.1: Examples for appearance adaptation in scenario $B \rightarrow H$. First row (left to right): MSI and nDSM for a patch from D^S, reference label map. Rows 2-5: The first and second columns show the adapted images using the appearance adaptation network trained with the respective variant of UDA. The third column shows the predictions for the adapted images using a classifier that was trained in D^T and the last column shows the corresponding F_1 score in %. Colour-codes of label maps as in Figure 5.2. The last row shows an example from D^T.

distributions of features (cf. Section 4.5). This could be a reason why the adapted images in the variants V_{SEP} and V_{BSLN} tend to show *sealed ground* where there is *low vegetation* in the reference (cf. the positions marked by black arrows in Figure 6.1). Note that these wrongly adapted areas are in fact classified as *sealed ground* by the classifier trained in D^T. When using the architectural modifications for improved semantic consistency, i.e. variants V_{RD} and V_{AG}, the respective areas are adapted correctly, i.e. in a semantically consistent way, and are classified as *low vegetation*. Although the adapted images using these variants are not classified completely correct, the performance is clearly better compared to the first two variants. The variant V_{RD} achieves the highest performance when classifying all adapted images from the validation set using the classifier trained in D^T, with an $\bar{F}_1$ score of 78.3 %. As expected, the variant V_{SEP}, in which only the adversarial loss term is used to train $\mathcal{A}$, i.e. in which the only constraint is for the adapted images to give an overall impression that is similar to the images from D^T, the semantic consistency is poor. Not only do artefacts appear in the adapted images, but also the performance of a classifier trained in D^T is very low ($\bar{F}_1 = 53.6\,\%$).

The next scenario that is evaluated with respect to the appearance adaptation is $H \rightarrow N$. The results of appearance adaptation and the respective predictions are shown in Figure 6.2.

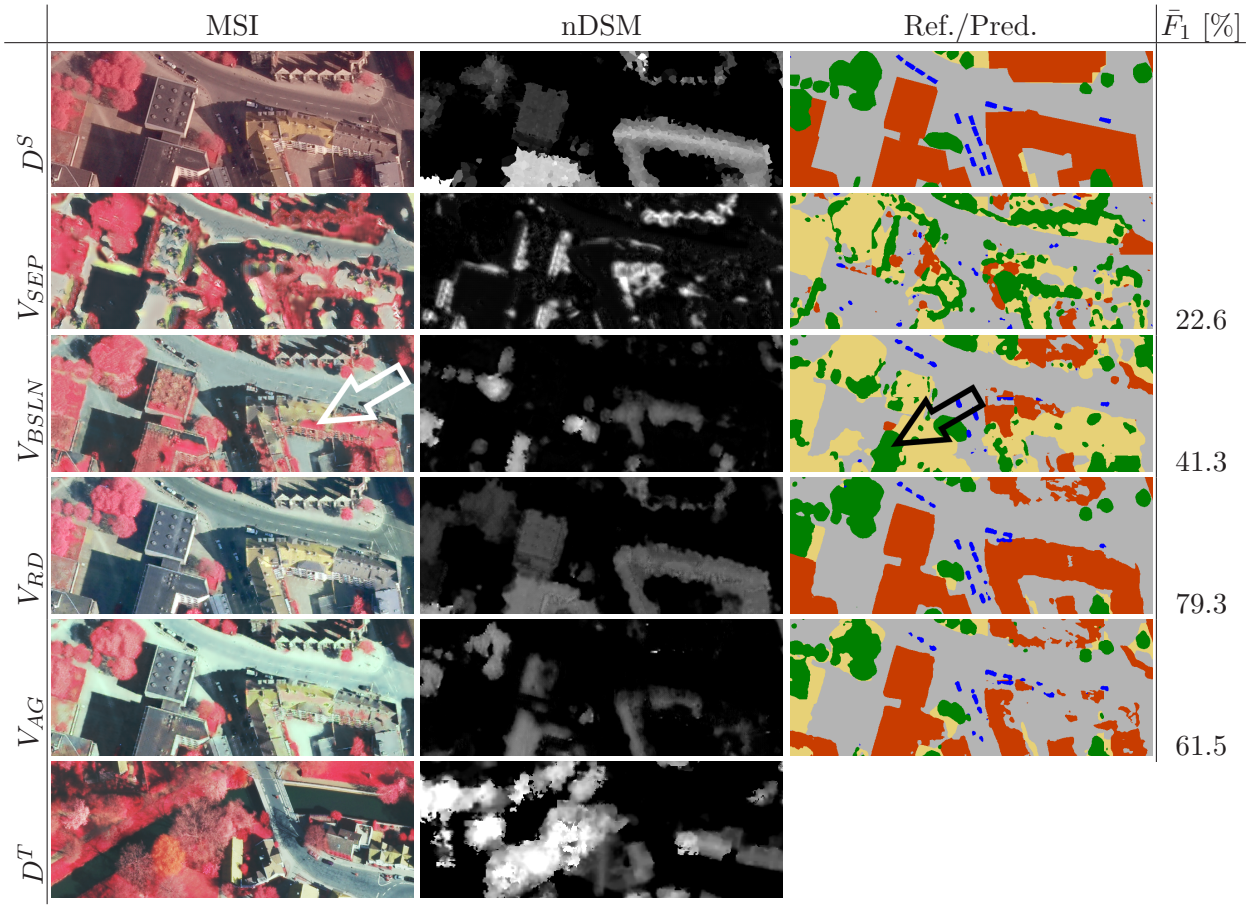

Figure 6.2: Examples for appearance adaptation in scenario $H \rightarrow N$. The structure of the figure follows the one of Figure 6.1.

In this scenario, the results of appearance adaptation are quite heterogeneous for the different variants. The variants V_{SEP}, V_{BSLN} and V_{AG} show inconsistent adaptations of building areas. This is most prominent in the adaptations related to the variants V_{SEP} and V_{BSLN}. Here, building areas were adapted to structures that look like high vegetation and are classified as such by the target domain classifier (cf. black arrow in Figure 6.2). Furthermore, in the variants V_{SEP} and V_{BSLN}, after appearance adaptation, the trees in the upper left area correspond to a structure that looks like low vegetation in D^T. In general, the appearance adaptation networks trained using these two variants hallucinate many structures that look like low vegetation in D^T, which is to be expected because this class is much more frequent in D^T. The building to the right could only be adapted by V_{RD} in a way such it is mostly classified correctly, i.e. in a semantically consistent way. The other image adaptations result in hallucinations of patterns that look like high and low vegetation (cf. white arrow in Figure 6.2) or like sealed ground in D^T. Regarding the performances of the target domain classifier applied to the adapted images, again the variant V_{RD} achieves the highest $\bar{F}_1$ score of 79.3 %, indicating that this variant results in the highest semantic consistency. The lowest performance is related to the variant V_{SEP} with a $\bar{F}_1$ score of 22.6 %. The variant V_{AG} that uses the auxiliary generator results in a higher semantic consistency compared to V_{BSLN}, with the

target domain classifier achieving a $\bar{F}_1$ score of 61.5 %, which nevertheless is still much lower than the one achieved by V_{RD}.

The third scenario that is discussed in detail is $P_{20} \rightarrow V_{20}$; the results of appearance adaptation and the respective predictions for this scenario are shown in Figure 6.3. This scenario is interesting, because the adapted images related to V_{BSLN} already show a quite high semantic consistency. The corresponding $\bar{F}_1$ score is comparable to those that were achieved using the architectural modifications in the variants V_{RD} and V_{AG}. This indicates that the differences with respect to the label distributions are rather small between the domains P_{20} and V_{20}. A small difference in the global label distribution was already confirmed in Section 5.1.1, cf. Table 5.1. However, the variant V_{SEP} still fails to produce adapted images that are semantically consistent and results in a very low performance when applying the target domain classifier. It is concluded that even in adaptation scenarios in which the domains are rather similar with respect to the label distributions, the supervised loss term has to be considered when training the appearance adaptation network in order to achieve semantic consistency.

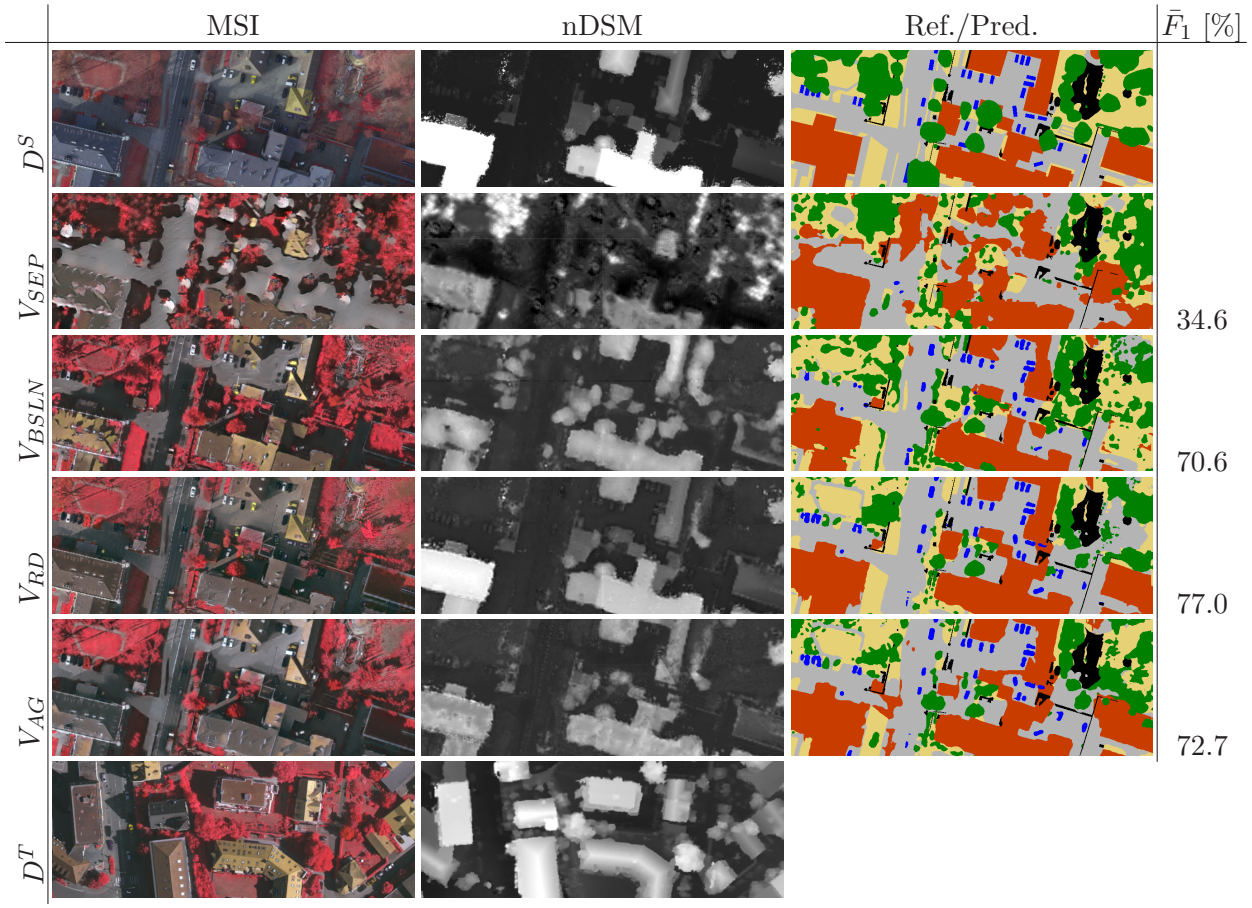

Figure 6.3: Examples for appearance adaptation in scenario $P_{20} \rightarrow V_{20}$. The structure of the figure follows the one of Figure 6.1.

A detailed discussion of the remaining two scenarios $N \rightarrow P_{20}$ and $V_{20} \rightarrow B$ is omitted at this point as the effects that can be observed are very similar to those in the scenarios just discussed and the quantitative analysis also turns out to be similar. Figures showing qualitative results similar to those in Figures 6.1 - 6.3 can be found in Appendix A. An overview of the $\bar{F}_1$ scores achieved by

applying the target domain classifiers to the adapted images is presented in Table 6.4. Note that the numbers for $B \rightarrow H$, $H \rightarrow N$ and $P_{20} \rightarrow V_{20}$ are identical to those in Figures 6.1 - 6.3.

Variant	$B \rightarrow H$	$H \rightarrow N$	$N \rightarrow P_{20}$	$P_{20} \rightarrow V_{20}$	$V_{20} \rightarrow B$
V_{SEP}	53.6	22.6	40.2	34.6	47.7
V_{BSLN}	62.8	41.3	60.7	70.6	66.2
V_{RD}	78.3	79.3	74.8	77.0	77.6
V_{AG}	75.2	62.6	61.5	72.7	72.9

Table 6.4: $\bar{F}_1$ scores in % obtained for the predictions by classifiers trained in D^T for images from D^S that were adapted to D^T using four variants of the proposed method for UDA for five different adaptation scenarios.

In all scenarios, the adapted images related to variant V_{SEP} lead to the lowest performance when classified by a classifier trained in the respective target domain. It is concluded that the variant V_{BSLN} that uses the supervised loss term for training the appearance adaptation network without any further architectural modifications leads to a higher semantic consistency compared to variant V_{SEP}. This underlines the importance of the joint training scheme in order to consider the supervised loss term for training $\mathcal{A}$.

The methods aiming to improve the semantic consistency introduced in the variants V_{RD} and V_{AG}, respectively, lead to an even higher semantic consistency, i.e. to better classification results when applying the classifiers trained in D^T to the adapted images, compared to V_{BSLN}. The performance values using V_{AG} are higher than those related to V_{BSLN}, and the variant V_{RD} outperforms the other variants in all evaluated scenarios by quite a large margin (about $3\,\% - 13\,\%$). It is noted that the difference between the performance of the best and the worst variants is largest in the two scenarios that include H either as source or target domain. This indicates that achieving semantic consistency is particularly difficult in these scenarios, which is in line with the observation that H is not well represented by the other domains (cf. Section 6.1).

To summarise, the semantic consistency is larger when using the supervised loss term for training the appearance adaptation network and it is further improved when one of the additional architectural modifications is used. The highest semantic consistency is achieved using the variant V_{RD}. Due to the low performance of the variant V_{SEP} with respect to the semantic consistency, it is not considered in the remaining experiments. The results of this evaluation can be used to partially answer the second scientific question. In particular, it was found out that using the joint training scheme is not sufficient to achieve a semantically consistent adaptation even though it is better than the variant in which the loss term $\mathcal{L}_{sup}^{ST}$ is not used to train the appearance adaptation network. Both methods aiming at improving the semantically consistency have been shown to actually lead to a higher degree of semantic consistency and are, thus, considered to be successful in that regard. Whether the improved semantic consistency actually leads to a higher performance of UDA will be evaluated in the next set of experiments.

6.2.2 Evaluation of Unsupervised Domain Adaptation

In this section, the performance of the adapted classifiers is evaluated. To this end, the average $\bar{F}_1$ score for the target domain in each adaptation scenario is reported for the variants V_{BSLN}, V_{RD} and V_{AG}. The results for the baseline variant V_{BSLN} are given in Table 6.5. Table 6.6 show the results for the variant V_{RD} and Table 6.7 shows the improvements of that variant compared to V_{BSLN}. Accordingly, Tables 6.8 and 6.9 show the results and improvements for the variant V_{AG}.

Evaluation of V_{BSLN}: Using this variant, the average performance is higher by 1.8 % in the $\bar{F}_1$ score compared to the naïve transfer, thus, on average, a positive transfer was achieved. However, there are several cases of negative transfers; in particular, all scenarios in which $D^S = H$ resulted in a negative transfer with respect to the $\bar{F}_1$ score (-2.3 % on average). This observation is in line with the conclusions drawn for the evaluation of semantic consistency (cf. Section 6.2.1). In particular, as the adapted images in the scenario $B \rightarrow H$ contain a lot of hallucinated structures when using V_{BSLN} (cf. Figure 6.1), the classifier can barely be improved when it is trained on such adapted images. A large negative transfer can be observed in the scenario $N \rightarrow V_{20}$ (-11.7 % in $\bar{F}_1$ on average). In this scenario, the standard deviation is rather high, which indicates that the results are not very stable. A closer look at the five repetitions of this experiment reveals that in three cases, this scenario resulted in a minor negative transfer of about -1 % in the $\bar{F}_1$, in one case a small positive transfer was achieved (+0.5 %) and in one case there was a very large negative transfer in which the $\bar{F}_1$ score after UDA was only 25.2 %. The highest improvements in the $\bar{F}_1$ scores were achieved in the scenarios in which $D^T = H$, which is probably due to the fact that in these scenarios the performance of naïve transfer was rather poor.

D^S \ D^T	B	H	N	P_{20}	V_{20}	Avg.
B	-	65.1±2.9 (3.6)	81.6±0.6 (1.2)	74.4±0.8 (3.7)	82.5±0.2 (0.6)	75.9±0.8 (2.3)
H	72.3±0.9 (-4.5)	-	76.0±1.0 (-1.1)	76.6±0.8 (-1.8)	76.2±1.5 (-1.6)	75.3±0.5 (-2.3)
N	83.8±0.2 (0.9)	70.8±1.3 (7.7)	-	79.5±1.2 (1.5)	70.9±22.9 (-11.7)	76.2±5.7 (-0.4)
P_{20}	76.9±0.9 (3.2)	73.7±0.5 (12.9)	79.9±0.5 (2.3)	-	80.3±0.5 (3.1)	77.7±0.2 (5.4)
V_{20}	81.5±0.4 (1.4)	68.2±1.0 (12.4)	80.7±0.3 (1.7)	76.4±0.6 (0.2)	-	76.7±0.2 (3.9)
Avg.	78.6±0.4 (0.3)	69.5±0.6 (9.1)	79.6±0.3 (1.0)	76.7±0.5 (0.9)	77.5±6.0 (-2.4)	76.4±1.2 (1.8)

Table 6.5: $\bar{F}_1$ scores in % obtained for the test sets in D^T after source-training in D^S and adapting the classifier to D^T using the variant V_{BSLN}. Each cell gives mean and standard deviation of $\bar{F}_1$ over five runs of the respective scenario. The value in parentheses corresponds to the difference in $\bar{F}_1$ compared to the naïve transfer setting in which the classifier trained in D^S was directly applied to D^T (cf. Table 6.2). A positive value corresponds to a positive transfer with respect to the $\bar{F}_1$ score of the same scenario. The last row provides the performance for each target domain averaged over all source domains and the last column provides the performance for each source domain averaged over all target domains. The values in the lower right cell correspond to the performance averaged over all scenarios.

Evaluation of V_{RD}: Turning to V_{RD} (Table 6.6), performing UDA using this variant results in a positive transfer for all adaptation scenarios with respect to the $\bar{F}_1$ score averaged of over the five repetitions of each adaptation. In all experiments (100 experiments in total), there were only 4 cases in which the adaptation resulted in a negative transfer, with a drop in $\bar{F}_1$ of $-0.8\,\%$ on average. Thus, this variant is considered to be rather stable and a negative transfer could be avoided in most cases. The average performance of $77.7\,\%$ is $1.3\,\%$ higher than the average performance that was achieved using V_{BSLN} and $3.1\,\%$ higher compared to the results of naïve transfer. The improvement over the variant V_{BSLN} is to a large degree related to the scenarios in which $D^S = H$. In these scenarios, a positive transfer could achieved, while the adaptation using V_{BSLN} resulted in a negative transfer. This is probably the case because the modification aims at improving the semantic consistency, which is particularly important in these scenarios, as it was shown e.g. in Figure 6.1.

D^S \ D^T	B	H	N	P_{20}	V_{20}	Avg.
B	-	68.4±0.6 (6.9)	82.3±0.2 (1.9)	73.0±0.4 (2.3)	82.4±0.3 (0.5)	76.5±0.1 (2.9)
H	77.6±0.9 (0.8)	-	79.2±0.6 (2.1)	79.6±0.6 (1.2)	79.6±0.3 (1.9)	79.0±0.4 (1.5)
N	84.2±0.3 (1.4)	71.2±1.2 (8.1)	-	79.7±0.6 (1.7)	83.5±0.2 (0.9)	79.6±0.3 (3.0)
P_{20}	76.2±0.9 (2.5)	69.7±1.1 (8.9)	80.0±0.2 (2.4)	-	78.9±0.4 (1.7)	76.2±0.5 (3.9)
V_{20}	81.5±0.3 (1.4)	68.4±0.6 (12.5)	81.1±0.4 (2.1)	77.6±0.3 (1.4)	-	77.2±0.1 (4.4)
Avg.	79.9±0.4 (1.5)	69.4±0.6 (9.1)	80.7±0.1 (2.1)	77.5±0.1 (1.6)	81.1±0.2 (1.2)	77.7±0.2 (3.1)

Table 6.6: $\bar{F}_1$ scores in % obtained for the test sets in D^T after source-training in D^S and adapting the classifier to D^T using the variant V_{RD}. The structure of the table follows the one of Table 6.5.

A detailed comparison of the results of V_{RD} to the results of V_{BSLN} shows that in 9/20 scenarios the improvement due to UDA is larger than the standard deviation of the differences over the five repetitions of each experiment. On the other hand, in two scenarios, a negative transfer occurred with a magnitude larger than the standard deviation, namely in $P_{20} \rightarrow V$ and $P_{20} \rightarrow V_{20}$. The differences and corresponding standard deviations are shown in Table 6.7.

D^S \ D^T	B	H	N	P_{20}	V_{20}	Avg.
B	-	3.3±2.8	0.7±0.4	-1.4±1.2	-0.1±0.4	0.6±0.8
H	5.3±0.8	-	3.2±1.1	3.0±1.2	3.4±1.5	3.7±0.9
N	0.4±0.2	0.4±1.8	-	0.2±0.9	12.5±22.8	3.4±5.6
P_{20}	-0.7±1.4	-4.0±1.2	0.1±0.6	-	-1.4±0.8	-1.5±0.5
V_{20}	0.0±0.6	0.2±1.2	0.4±0.3	1.2±0.4	-	0.4±0.2
Avg.	1.3±0.6	0.0±0.7	1.1±0.2	0.8±0.4	3.6±6.0	1.3±1.3

Table 6.7: Difference of $\bar{F}_1$ scores obtained using variants V_{RD} and V_{BSLN}. A positive value means that the variant V_{RD} performed better, considering the average over five repetitions of each experiment. The structure of the table follows the one of Table 6.5.

Interestingly, for the scenario $P_{20} \rightarrow V$, the results of the evaluation of the semantic consistency have shown that the variant V_{RD} leads to a improved semantic consistency compared to V_{BSLN} (cf. Table 6.4) while at the same time the variant V_{RD} performed worse than V_{BSLN} with respect to the performance of the classifier after UDA. It is assumed that in scenarios in which the appearance adaptation is semantically consistent when using the variant V_{BSLN}, applying the regularisation of $\mathcal{D}$ has negative effect on the performance of UDA. The regularisation might prevent the discriminator from achieving a high performance in discriminating the domains to some extent, resulting in a weaker supervised signal for training $\mathcal{A}$. Thus, the adapted images might look less like coming from D^T, which would mean that the classifier cannot adapt well to the appearance of objects in D^T. At the same time, when looking at all five scenarios from Table 6.4, a weak positive correlation between the improvement of semantic consistency when moving from V_{BSLN} to V_{RD} and the improvement of the performance due to UDA in those scenarios can be seen, with a correlation coefficient of 0.74. These observations indicate that improving the semantic consistency in appearance adaptation based UDA mostly leads to an improvement of the performance after UDA, even though this is not always the case.

It is concluded that, compared to V_{BSLN}, the variant V_{RD} results in a considerably better classification performance after UDA in about half of the scenarios, while it is considerably worse in only 2 scenarios. On average, the performance of V_{RD} is better, although the average improvement is as large as the standard deviation of the improvements over five repetitions of the experiment and, thus, only of limited significance. At the same time, the number of negative transfers is drastically reduced, which indicates that the method is more stable. Consequently, V_{RD} is considered to be better than V_{BSLN} because a higher average performance was achieved and no negative transfers with respect to the $\bar{F}_1$ scores occurred.

Evaluation of V_{AG}: The results for the alternative variant V_{AG} are reported in Table 6.8. The performance of this variant is again superior to the baseline variant V_{BSLN} with respect to the average performance of all scenarios and on the same level as the performance of variant V_{RD}. The largest reductions of the performance gap were achieved in the scenarios in which $D^T = H$ (about $8\,\% - 13\,\%$ in the $\bar{F}_1$ score). As these scenarios were identified as being rather difficult, it is concluded that method using the auxiliary generator performs well in challenging scenarios. There are also considerable reductions of the performance gap in the scenarios in which $D^S = P_{20}$. For the remaining scenarios the improvement compared to naïve transfer is rather small. In the scenario $H \rightarrow B$ this variant resulted in a considerable negative transfer of $-1.3\,\%$ with respect to the $\bar{F}_1$ score.

Analogously to Table 6.7, Table 6.9 shows the difference of the performance of the classifiers after UDA using variant V_{AG} compared to the ones that were achieved using variant V_{BSLN}. Comparing the $\bar{F}_1$ averaged over the target domains to those of the variant V_{BSLN} (cf. last column in Table 6.9) it can again be seen that the largest improvement, namely of $2.5\,\%$, is achieved for $D^S = H$. Comparing the results of V_{AG} to V_{BSLN}, in 6/20 scenarios the improvement due to UDA is better than the standard deviation of the differences over the five repetitions of each experiment. Only in one scenario a negative transfer occurred with a magnitude larger than the

D^S \ D^T	B	H	N	P_{20}	V_{20}	Avg.
B	-	69.6±1.0 (8.1)	82.0±0.2 (1.6)	72.9±1.2 (2.1)	82.4±0.2 (0.5)	76.7±0.5 (3.1)
H	75.5±1.1 (-1.3)	-	78.9±1.3 (1.7)	78.4±0.9 (0.0)	78.2±0.8 (0.4)	77.7±0.8 (0.2)
N	83.9±0.1 (1.0)	71.0±0.8 (7.9)	-	79.6±0.3 (1.6)	83.0±0.2 (0.4)	79.4±0.2 (2.7)
P_{20}	77.7±0.6 (4.0)	72.3±1.3 (11.4)	80.3±0.2 (2.7)	-	79.6±0.4 (2.5)	77.5±0.2 (5.2)
V_{20}	81.4±0.7 (1.3)	69.1±1.4 (13.2)	80.8±0.5 (1.7)	77.3±0.7 (1.1)	-	77.1±0.5 (4.3)
Avg.	79.6±0.4 (1.2)	70.5±0.6 (10.2)	80.5±0.3 (1.9)	77.0±0.6 (1.2)	80.8±0.3 (0.9)	77.7±0.2 (3.1)

Table 6.8: $\bar{F}_1$ scores in % obtained for the test sets in D^T after source-training in D^S and adapting to D^T using the variant V_{AG}. The structure of the table follows the one of Table 6.5.

standard deviation, i.e. $B \rightarrow P_{20}$. In the scenario $P_{20} \rightarrow H$, the variant V_{AG} also performed worse than V_{BSLN}, but the magnitude of the difference is as large as the corresponding standard deviation, thus, the difference is only of limited significance.

D^S \ D^T	B	H	N	P_{20}	V_{20}	Avg.
B	-	4.4±2.4	0.4±0.6	-1.6±1.5	-0.1±0.2	0.8±0.8
H	3.2±1.5	-	2.9±1.7	1.8±0.9	2.0±1.6	2.5±0.7
N	0.1±0.1	0.3±1.0	-	0.1±1.0	12.0±22.9	3.1±5.8
P_{20}	0.8±1.2	-1.4±1.4	0.5±0.6	-	-0.6±0.7	-0.2±0.3
V_{20}	-0.1±1.1	0.9±2.0	0.0±0.4	0.9±0.4	-	0.4±0.6
Avg.	1.0±0.5	1.0±1.0	0.9±0.5	0.3±0.5	3.3±5.9	1.3±1.2

Table 6.9: Difference of $\bar{F}_1$ scores obtained using variants V_{AG} and V_{BSLN}. A positive value means that the variant V_{AG} performed better, considering the average over five repetitions of the experiments. The structure of the table follows the one of Table 6.5.

Figure 6.4 shows some examples for the images generated by the auxiliary generator $\mathcal{G}$ in the adaptation process. First of all, it is noticed that the generated images give a similar impression to the images from the respective target domain, cf. third and fourth rows in Figure 6.4 for exemplary images from D^T. The overall layout of the created scenes is not very meaningful, but this was not expected, because the discriminator only rates support windows of size $70 \times 70\,px$. Instead of a meaningful scene layout, it is required for the created images to compensate for potential differences in the local distributions of labels in the label maps.

Regarding this requirement, the generated images are actually meaningful. For example, the image generated for the scenario $B \rightarrow H$ shows sealed ground and high buildings, two classes that are less frequent in the source domain. The image generated for the scenario $H \rightarrow N$ shows structures that can be interpreted as natural ground and water, which are again structures that do not appear in the source domain. In a similar way, the auxiliary generator predicted structures that look like natural ground in the scenario $V_{20} \rightarrow B$ to compensate for the fact that large areas

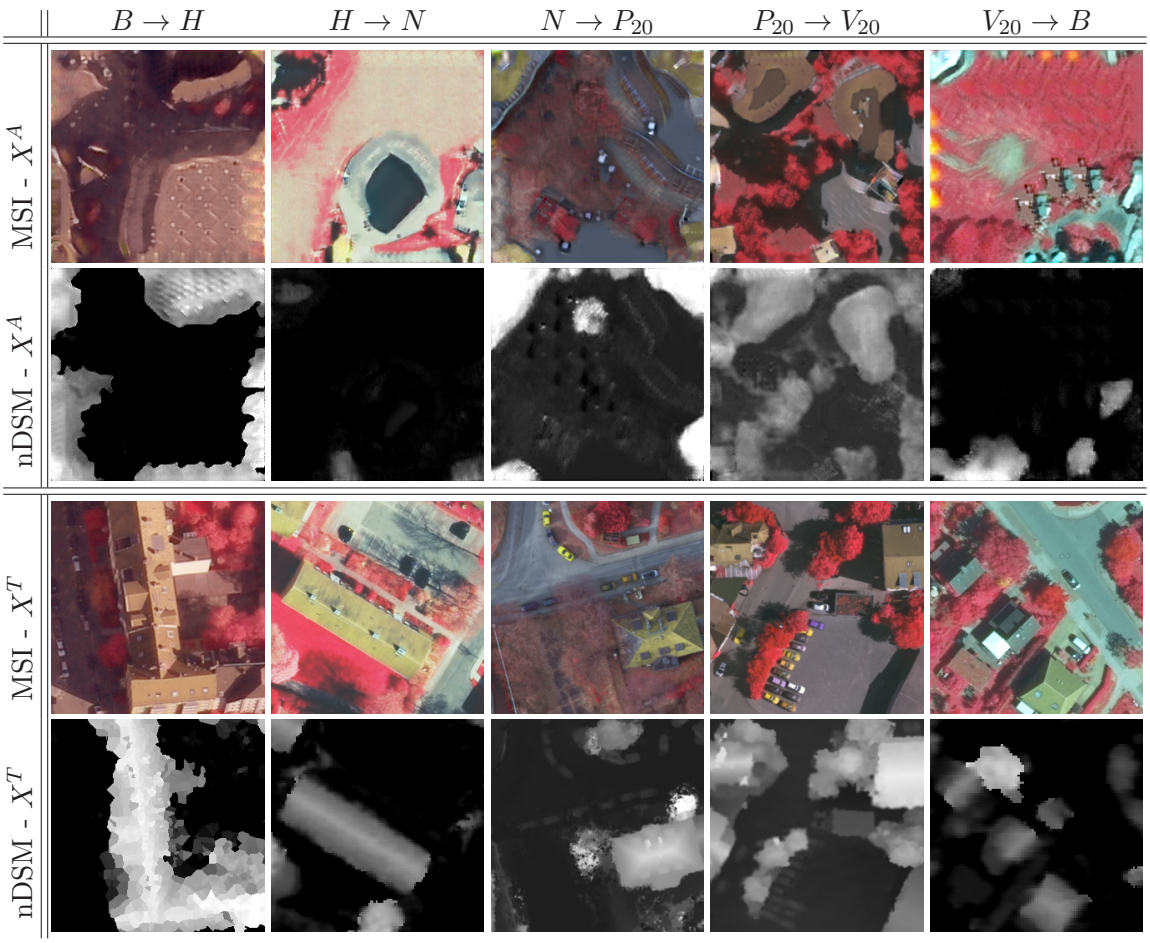

Figure 6.4: Examples for auxiliary images X^A (rows 1, 2) generated by the auxiliary generator G in different adaptation scenarios in variant V_{AG} and exemplary images from X^T (rows 3, 4). The image size is $256 \times 256\,px$. Rows 1 and 3 show the infrared red and green channels as false colour images, and rows 2 and 4 show visualisations of the nDSMs.

showing natural ground are much less frequent in the source domain. For the scenario $N \rightarrow P_{20}$, $\mathcal{G}$ generated images that show high yellow buildings, which are very common in D^T, though not directly related to the label distributions. In the scenario $P_{20} \rightarrow V_{20}$, the global label distributions of both domains are quite similar (cf. Section 5.1.1). In this scenario the baseline variant V_{BSLN} already worked quite well, which is the reason why it is assumed that the local distributions of the labels are also quite similar between the two domains. Thus, there is no need to compensate for differences in the global label distributions and it is reasonable that the generated image seems rather representative for the target domain with respect to the appearance, but also with respect to the underlying label distribution.

To conclude, extending the architecture for deep UDA by the auxiliary generator did work in the expected way. It could increase the semantic consistency and improve the performance of UDA in most scenarios.

Summary of Variant Comparison without ABN: Both variants aiming to improve the semantic consistency did improve the performance of the classifiers after UDA compared to the

baseline variant V_{BSLN}. A direct comparison of the respective improvements for all adaptation scenarios reveals that both extensions achieved similar improvements. This is visualised in Figure 6.5. The figure shows the differences of the averaged $\bar{F}_1$ scores achieved with variants V_{AG} and V_{BSLN} (cf. Table 6.9) as a function of the differences of the averaged $\bar{F}_1$ scores achieved with variants V_{RD} and V_{BSLN} (cf. Table 6.7) in a logarithmic scale. A strong correlation can be observed, i.e. the correlation coefficient is 0.95.

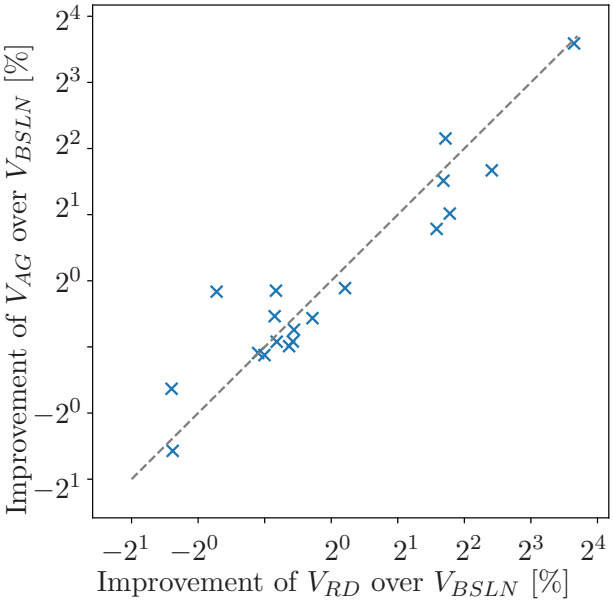

Figure 6.5: Comparison of the improvement of the classification performance of V_{AG} and V_{RD} to the baseline V_{BSLN} with respect to $\bar{F}_1$.

It is noted that in preliminary experiments, a combination of the two extended methods did not result in a higher classification performance. The reason for this might be that both methods are similar with respect to the actual effect on the training of the classifier.

In summary, the methods V_{RD} and V_{AG} perform similarly, i.e. there is no considerable difference in the $\bar{F}_1$ score, averaged over all experiments. However, there are some advantages of the variant V_{RD}. First, the variant V_{RD} resulted in a better performance on average compared to V_{AG} in 15/20 adaptation scenarios. Second, the variant V_{RD} did not result in a negative transfer with respect to the averaged $\bar{F}_1$ scores, but using the variant V_{AG} resulted in a negative transfer in one scenario. Lastly, the variant V_{RD} has a smaller memory footprint, because no additional network is required. This also reduces the training time and makes the whole architecture easier to tune as the number of hyper-parameters decreases. As a consequence, V_{RD} is considered to be preferable to V_{BSLN} and V_{AG}.

6.2.3 Combination of Appearance Adaptation with Adaptive Batch Normalisation

In the experiments reported in this section, the appearance adaptation based strategy for deep UDA proposed in this thesis is combined with adaptive batch normalisation (ABN). In the variant $V_{RD,ABN}$, ABN is used as an additional adaptation step after appearance adaptation using the variant V_{RD} and in variant $V_{AG,ABN}$, ABN is used as an additional adaptation step after adaptation

using the variant V_{AG}. The corresponding results of the variants $V_{RD,ABN}$ and $V_{AG,ABN}$ are presented in Tables 6.10 and 6.11, respectively.

D^S \ D^T	B	H	N	P_{20}	V_{20}	Avg.
B	-	70.4±1.5 (8.9)	83.3±0.2 (2.9)	77.7±0.3 (7.0)	82.6±0.2 (0.7)	78.5±0.4 (4.9)
H	75.4±1.1 (-1.4)	-	73.2±0.9 (-3.9)	79.2±0.8 (0.7)	78.1±0.3 (0.3)	76.5±0.4 (-1.1)
N	84.7±0.1 (1.9)	71.0±0.6 (8.0)	-	79.6±2.7 (1.5)	83.2±0.1 (0.6)	79.6±0.6 (3.0)
P_{20}	77.1±0.4 (3.4)	70.4±0.8 (9.5)	80.1±0.3 (2.5)	-	78.8±0.2 (1.6)	76.6±0.2 (4.2)
V_{20}	80.8±0.2 (0.6)	70.4±0.4 (14.5)	80.8±0.3 (1.8)	79.8±0.2 (3.6)	-	77.9±0.1 (5.1)
Avg.	79.5±0.2 (1.1)	70.5±0.6 (10.2)	79.3±0.2 (0.8)	79.1±0.6 (3.2)	80.7±0.2 (0.8)	77.8±0.1 (3.2)

Table 6.10: $\bar{F}_1$ scores in % obtained for the test sets in D^T after source-training in D^S and adapting to D^T using the variant $V_{RD,ABN}$. The structure of the table follows the one of Table 6.5.

D^S \ D^T	B	H	N	P_{20}	V_{20}	Avg.
B	-	69.9±0.3 (8.4)	82.9±0.3 (2.5)	78.5±0.5 (7.8)	82.8±0.3 (0.9)	78.5±0.2 (4.9)
H	74.0±1.1 (-2.8)	-	72.2±1.4 (-5.0)	77.0±0.8 (-1.5)	75.7±0.7 (-2.0)	74.7±0.7 (-2.8)
N	84.1±0.2 (1.2)	68.0±2.1 (4.9)	-	80.5±0.3 (2.5)	82.8±0.1 (0.2)	78.8±0.5 (2.2)
P_{20}	77.8±0.3 (4.1)	70.1±1.8 (9.3)	80.9±0.3 (3.3)	-	79.9±0.5 (2.7)	77.2±0.4 (4.8)
V_{20}	81.0±0.6 (0.9)	69.0±1.2 (13.1)	80.2±0.3 (1.1)	78.8±0.7 (2.6)	-	77.3±0.4 (4.4)
Avg.	79.2±0.3 (0.9)	69.2±0.9 (8.9)	79.1±0.4 (0.5)	78.7±0.5 (2.9)	80.3±0.3 (0.4)	77.3±0.2 (2.7)

Table 6.11: $\bar{F}_1$ scores in % obtained for the test sets in D^T after source-training in D^S and adapting to D^T using the variant $V_{AG,ABN}$. The structure of the table follows the one of Table 6.5.

Both extended methods, $V_{RD,ABN}$ and $V_{AG,ABN}$ achieve a positive transfer on average with an improvement of the $\bar{F}_1$ score by 3.2 % and 2.7 %, respectively. While in most cases, a positive transfer was achieved, both methods seem to struggle with the scenarios in which $D^S = H$. Here, the variant $V_{AG,ABN}$ leads to a negative transfer in all four scenarios with an average drop of -2.8 %. The variant $V_{RD,ABN}$ results in a negative transfer in the scenarios $H \rightarrow B$ and $H \rightarrow N$, resulting in an average drop of -1.1 % in the scenarios in which $D^S = H$. It is assumed that applying ABN fails in these scenarios, because the global label distribution in H is very different from the other domains (cf. Table 5.1), which results in a wrong alignment of the activation distributions. In principle, this also affects the scenarios in which $D^T = H$. However, in these scenarios the performance of naïve transfer is rather poor (cf. Table 6.2), which makes it easier to achieve a positive transfer.

6.2.4 Final Comparison of Variants

Finally, the results of UDA using the variants V_{BSLN}, V_{RD}, V_{AG}, $V_{RD,ABN}$, $V_{AG,ABN}$ are compared side by side. In Table 6.12, the average $\bar{F}_1$ scores for each D^T, averaged over all source domains is presented for each variant. Table 6.13 shows the values for the overall accuracy, again for each D^T, averaged over all source domains. The variant V_{BSLN} performs worst in the scenarios where $D^T = B$ and $D^T = V_{20}$ and also results in the worst average performance for both metrics. The variant $V_{AG,ABN}$ is the second worst variant with respect to the average performance and does not achieve the highest performance in any case, again considering both metrics. For the remaining target domains, except for the domain H, the variants R_{RD} or $R_{RD,ABN}$ achieve the highest performance metrics. However, for the scenarios in which $D^T = H$, the variant V_{AG} performs best. In particular with respect to the overall accuracy, the variant V_{AG} performs better than the other methods.

Str. \ D^T	B	H	N	P_{20}	V_{20}	Avg.
N.T.	78.4±0.2	60.3±1.2	78.6±0.5	75.9±0.3	79.9±0.2	74.6±0.4
V_{BSLN}	78.6±0.4	69.5±0.6	79.6±0.3	76.7±0.5	77.5±6.0	76.4±1.2
V_{RD}	**79.9**±0.4	69.4±0.6	**80.7**±0.1	77.5±0.1	**81.1**±0.2	77.7±0.2
V_{AG}	79.6±0.4	**70.5**±0.6	80.5±0.3	77.0±0.6	80.8±0.3	77.7±0.2
$V_{RD,ABN}$	79.5±0.2	**70.5**±0.6	79.3±0.2	**79.1**±0.6	80.7±0.2	**77.8**±0.1
$V_{AG,ABN}$	79.2±0.3	69.2±0.9	79.1±0.4	78.7±0.5	80.3±0.3	77.3±0.2
T.T.	87.1±0.2	80.3±0.2	85.5±0.2	89.1±0.1	84.3±0.2	85.3±0.2

Table 6.12: Comparison of different variants of the proposed method for UDA. The values correspond to the average $\bar{F}_1$ score in % by target domain after adaptation from each source domain (averaged over the source domains) and the corresponding standard deviations. The first row shows the results of naïve transfer (N.T.) and the last row (T.T.) shows the results of training in the target domain. Str.: Strategy. Avg.: Average $\bar{F}_1$ score over all adaptation scenarios. The best results after UDA are printed in bold font.

Although all variants achieve a positive transfer on average, there remains a considerable performance gap after UDA. The best performing variants could compensate for about 25 % of the initial performance gap with respect to the overall accuracy and for about 30 % with respect to the $\bar{F}_1$ score.

Based on the results presented in Tables 6.12 and 6.13, the variants R_{RD} and $R_{RD,ABN}$ are considered to be the best performing variants. Variant R_{RD} is the most stable variant with zero negative transfers considering the average $\bar{F}_1$ scores and with the highest average performance considering the overall accuracy. The variant $R_{RD,ABN}$ achieves the highest performance on average, considering the $\bar{F}_1$ score. However, in Section 6.2.3 it was shown that this comes at the cost of more negative transfers. This is the reason why in the following experiments, only the variants R_{RD} and $R_{RD,ABN}$ will be evaluated.

Str. \ D^T	B	H	N	P_{20}	V_{20}	Avg.
N.T.	80.0±0.4	69.5±1.0	82.2±0.4	77.5±0.5	82.7±0.2	78.4±0.4
V_{BSLN}	79.8±0.4	77.1±0.7	82.5±0.4	78.7±0.3	80.4±5.2	79.7±1.1
V_{RD}	**81.2**±0.4	77.3±0.4	**83.9**±0.1	79.0±0.1	**83.8**±0.1	**81.0**±0.1
V_{AG}	80.9±0.4	**77.9**±0.5	83.6±0.4	78.6±0.6	83.4±0.3	80.9±0.2
$V_{RD,ABN}$	80.7±0.3	77.0±0.6	81.4±0.3	**80.8**±0.4	83.1±0.1	80.6±0.1
$V_{AG,ABN}$	80.5±0.4	75.2±1.0	80.9±0.5	80.2±0.6	82.6±0.3	79.9±0.2
T.T.	88.6±0.1	86.4±0.1	88.7±0.1	89.8±0.1	86.9±0.1	88.1±0.1

Table 6.13: Comparison of different variants of the proposed method for UDA. The values correspond to the average overall accuracy in % by target domain after adaptation from each source domain (averaged over the source domains) and the corresponding standard deviations. The first row shows the results of naïve transfer (N.T.) and the last row (T.T.) shows the results of training in the target domain. Str.: Strategy. Avg.: Average $\bar{F}_1$ score over all adaptation scenarios. The best results after UDA are printed in bold font.

6.2.5 Detailed Evaluation of Selected UDA Scenarios

To better assess the capabilities and limitations of the proposed variants, the results for UDA using the variants V_{RD} and $V_{RD,ABN}$ in five adaptation scenarios are evaluated in detail, i.e. by performing a qualitative evaluation and investigating the class-wise performance, measured by the class-specific F_1 scores.

Figure 6.6 shows exemplary predictions made by the classifiers after adaptation using the variants V_{RD} and $V_{RD,ABN}$. The figure also shows predictions made by the classifiers that were trained in D^S (corresponding to the naïve transfer setting), and classifiers that were trained in D^T (corresponding to the intra-domain evaluation setting). The latter predictions should serve as a qualitative reference for label maps that could be predicted when training in the respective target domain. It can be seen that the predictions made by the classifiers trained in D^T are very similar to the reference, but some errors remain, for example the hedge in the upper-right area in the adaptation scenario $N \rightarrow P_{20}$ is not detected by the classifier trained in P_{20}. Figure 6.7 presents the class-wise F_1 scores on the test set of D^T for the selected scenarios. The corresponding numbers can be found in the Appendix B.

For the scenarios $N \rightarrow P_{20}$, $P_{20} \rightarrow V_{20}$ and $V_{20} \rightarrow B$, the difference in the performance in the target domain when comparing the two variants of the proposed method to naïve transfer was comparably small with less than 1.1 % in $\bar{F}_1$ for the variant V_{RD} (cf. Table 6.6) and less than 1.6 % in $\bar{F}_1$ for the variant $V_{RD,ABN}$ (cf. Table 6.10). This is in line with the observations made in the visual comparison of the respective results in Figure 6.6. The predicted label maps corresponding to the variants V_{RD} and $V_{RD,ABN}$ look rather similar to the predictions made by the classifier trained in the respective source domain of those scenarios.

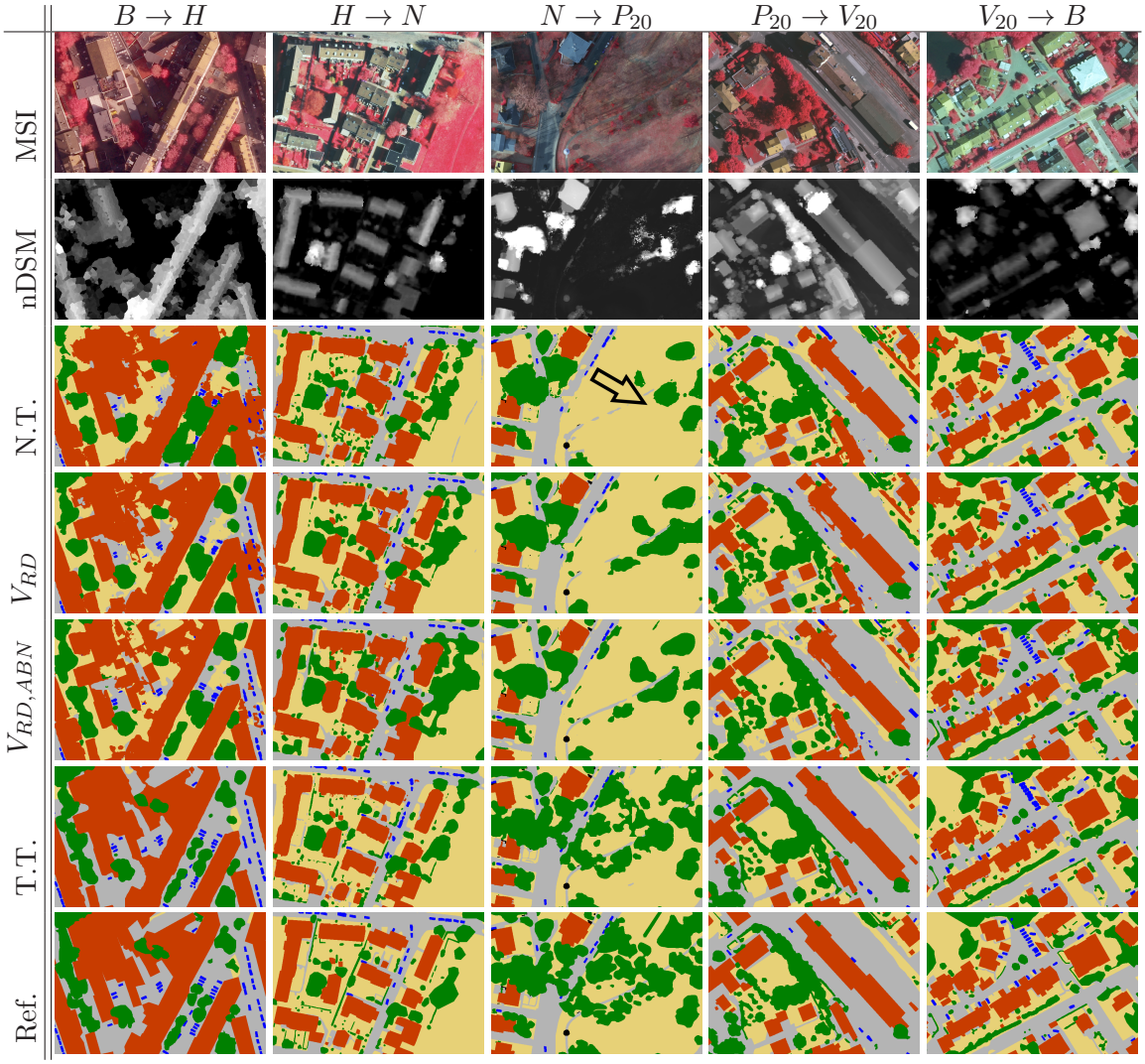

Figure 6.6: Exemplary image patches and predictions for five adaptation scenarios using classifiers that were trained using different variants. The first row shows the MSI data for the image patch and the second row shows the nDSM. N.T.: Naïve transfer (classifier was trained only in D^S). V_{RD}: First variant of the proposed method (classifier trained in D^S and adapted to D^T using the variant V_{RD}). $V_{RD,ABN}$: Second variant of the proposed method (classifier trained in D^S and adapted to D^T using the variant $V_{RD,ABN}$). T.T.: Target training (classifier was trained only in D^T). Ref.: Reference label map.

The most prominent errors in these scenarios are related to the areas showing the class *high vegetation* in the reference. In particular, in the scenario $N \rightarrow P_{20}$ the classifier trained in N cannot classify most of those areas correctly as *high vegetation* but instead predicts the class *low vegetation* (cf. area marked by the black arrow in Figure 6.6). This can be explained by seasonal differences, i.e. by the fact that in N trees without leaves are not represented. Using either of the variants to adapt the classifier to P_{20}, more areas that correspond to the class *high vegetation* are detected, but there remain many false negative predictions for that class. This is most likely based on the fact that trees in P_{20} do not have leaves and, thus, are barely visible in the nDSM. This makes them look very similar to areas that belong to the class *low vegetation* in the target domain. Looking at Figure 6.7, this observation is in line with the behaviour of the class-wise metrics. When

training a classifier in N and directly applying it to P_{20}, a large performance drop (cf. difference between the blue bars and the red bars in Table 6.7) can be observed for the classes *high vegetation* and *low vegetation*. When using either variant of the proposed method for appearance adaptation, the class *high vegetation* can slightly be improved, but a large performance gap remains. It is concluded that severe differences in the appearance of vegetation caused by seasonal effects in the two domains cannot be compensated by the proposed method for UDA. On a more general level it is deduced that objects in D^S which have an appearance that is closer to the appearance of another class in D^T than to the appearance of objects of the the same class in D^T remain problematic after UDA.

In the scenario $B \rightarrow H$, both variants for UDA achieve the largest improvements over naïve transfer among the five adaptation scenarios investigated here. In the predicted label maps in Figure 6.6 it can be seen that the classifier trained solely in D^S struggles with areas in the shadows of the high buildings, probably as such areas are not well represented in D^S. Here, both variants of the method clearly improve the predictions, but there remains a frequent confusion of the classes *low vegetation* and *sealed ground*. In Figure 6.7 it can be seen that the performance metrics for all classes could be improved by both methods for UDA, except for the class *building* which remains at a similar level. The largest improvements were obtained for the class *sealed ground* with +12.8 % and +13.7 % in the F_1 score using the variants V_{RD} and $V_{RD,ABN}$, respectively. Also the improvements for the class *car* are very high (more than 10 % in the F_1 score).

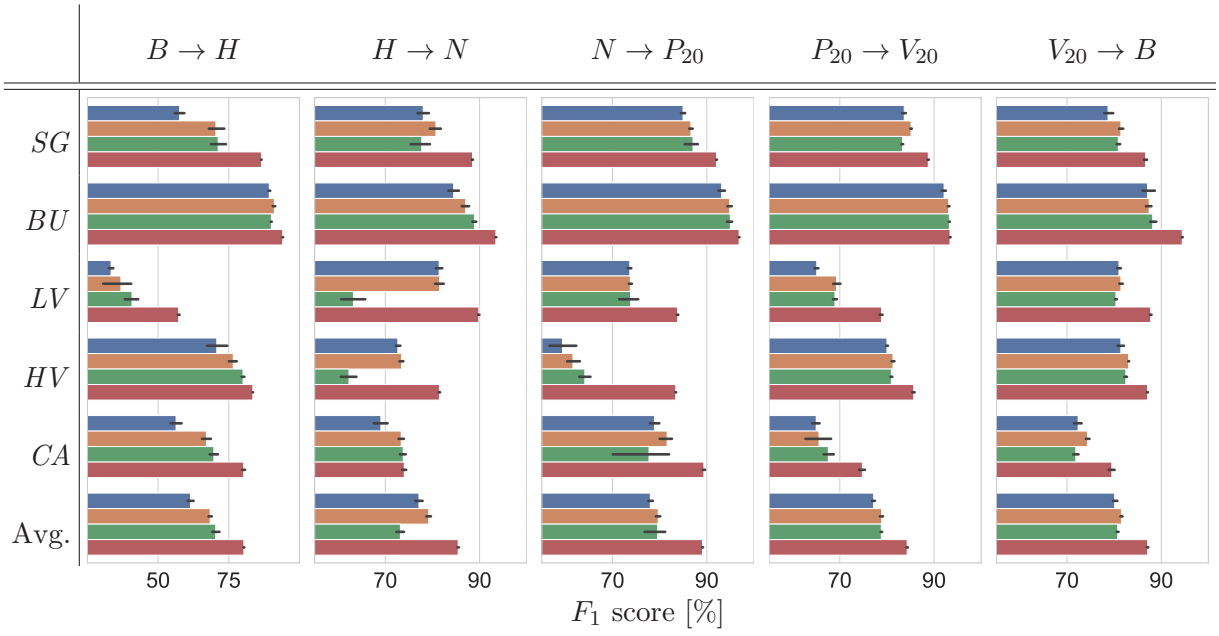

Figure 6.7: Class-wise F_1 scores and $\bar{F}_1$ score for different adaptation scenarios. The blue bars correspond to the classifiers trained in $D^S$and the red bars to the classifiers trained in D^T. The orange bars correspond to the classifiers that were trained in $D^S$and adapted to D^Tusing the proposed method for UDA in variant V_{RD}. The green bars correspond to the classifiers that were adapted using variant $V_{RD,ABN}$. Note that the scaling of the horizontal axis is different for the first adaptation setting. The black bars indicate the standard deviation based on five repetitions of the experiment. In particular, the length of the bars corresponds to twice the standard deviation. Class abbreviations as in Table 5.1.

The scenario $H \rightarrow N$ corresponds to a scenario in which appearance adaptation using V_{RD} resulted in a positive transfer, but the combination of appearance adaptation and ABN, i.e. using variant $V_{RD,ABN}$, lead to a negative transfer. In the qualitative example in Figure 6.6, it can be seen that there are many false positive predictions for the class *tree* in the prediction by the classifier that was adapted using variant $V_{RD,ABN}$, resulting in too large areas showing that class. Looking at the results for the adaptation scenario $H \rightarrow N$ in Figure 6.7, it can be seen that the performance of the classes *low vegetation* and *high vegetation* decreases, which is in line with the observation that after UDA large areas showing *low vegetation* were misclassified as *high vegetation*.

To understand why the performance of the variant $V_{RD,ABN}$ is particularly worse in the scenario $H \rightarrow N$, a closer look at the global label distributions is helpful. Figure 6.8 shows the global label distribution in H and N, but it also shows the distribution of the predicted labels for the validation set in N when using the variants V_{RD} and $V_{RD,ABN}$.

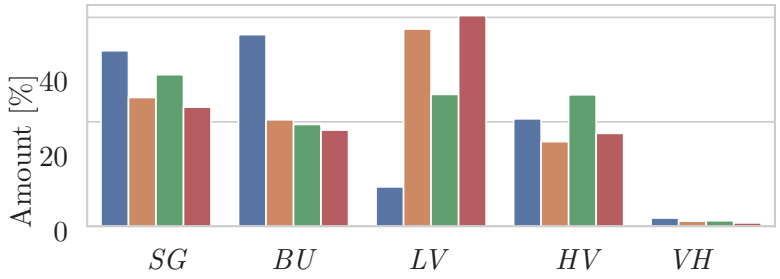

Figure 6.8: Global label distributions for the scenario $H \rightarrow N$. The blue bars show the label distribution in H and the red bars correspond to the one in N. The orange and green bars correspond to the label distribution in the predictions using the variants V_{RD} and $V_{RD,ABN}$, respectively, for the test set from N.

Looking at this figure, a considerable difference in the global label distributions of H and N can be seen, cf. blue and red bars in Figure 6.8. The difference is particularly large for the *classes building* and *low vegetation*. Comparing the label distributions of the predictions made by the two variants, it can be seen that when using variant $V_{RD,ABN}$ there is a large gap in the relative amount of pixels for the class *low vegetation* compared to the actual relative amount of labels for that class in D^T. (cf. green and red bars for the class *low vegetation* in Figure 6.8). This gap is much smaller for the predictions generated by variant V_{RD}. Thus, this problem is assumed to be related to applying ABN. Recall that when applying ABN, the marginal distributions of the features obtained for images from D^T are aligned to the ones that are used during training on the images from D^S. As the features are related to the underlying global label distributions of both domains, this alignment might not be meaningful if there are considerable differences in the global label distributions of both domains. In the toy example provided in Section 2.6.1 it was shown that ABN can result in a relative shift of the decision boundary towards a class which is more frequent in D^T compared to D^S. It is assumed that a similar effect is observed in the adaptation scenario $H \rightarrow N$, considering the class LV to be considerably more frequent in N than in H. Consequently, in the adaptation scenario $H \rightarrow N$ this effect results in many false negative predictions for the class *low vegetation*, because the decision boundaries are shifted towards the features that correspond to the class LV.

A look at the confusion matrix obtained for this adaptation scenario using the variant $V_{RD,ABN}$ shows that in most cases of false negative predictions for the class *low vegetation*, the class *high vegetation* is predicted instead, which can also be seen in Figure 6.6. The reason why the class *high vegetation* is predicted instead is probably because this class has the most similar appearance in the images and, thus, those two classes are difficult to distinguish. Interestingly, the class *building* is not negatively affected by the fact that there is a much higher fraction of pixels showing the class *building* in the source domain. Actually, when applying ABN as a subsequent adaptation step, the performance of that class increases (cf. the difference between the green and orange bars for the scenario $H \rightarrow N$ in Figure 6.7). It is assumed that the features for the class *building* are rather well separated from those of the other classes and, thus, aligning the activation distributions will not lead to wrong decision boundaries. To conclude this observation, it was found, that applying ABN may reduce the performance in the scenario in which the global label distributions are very different. However, classes which are easy to differentiate from the other classes may not be affected by this phenomenon.

Coming back to the thirst research question posed in Section 1.2, whether or not ABN can improve the performance of UDA actually depends on the adaptation scenario. It seems to work well in scenarios in which the global label distributions are rather similar but fails if they are rather different. This is problematic in a real application, because the label distribution in the target domain is unknown in an UDA setting. However, the strategy might be used if one has additional knowledge about the data in the domains. For example, the similarity of the global label distributions of two domains could be assessed by a visual inspection, which is probably much less time consuming than the labelling process in many applications.

6.3 Results of Experiment Set E3: Evaluation of Parameter Selection.

To evaluate the proposed entropy based parameter selection strategy it is compared against the strategy of using the parameter values obtained after a fixed number of iterations, which is used in most methods in the literature. To this end, the variant V_{RD} is considered, i.e. ABN is not considered here, because it has no impact on the parameter selection during appearance adaptation. In the adaptation process of each adaptation scenario, the performance of the classifier after each training epoch in terms of the $\bar{F}_1$ score on the test set of the target domain is tracked and compared against the performance that was achieved when using the parameter set that resulted in the minimal average entropy in D^T. The results are illustrated in Figure 6.9.

First of all, in most scenarios a positive trend of the $\bar{F}_1$ score is visible, which means that the adaptation process continuously improves the performance of the classifier in the target domain. The duration of the adaptation process of 50 epochs seems to be enough in most scenarios, but some still show an ascending trend towards the end of UDA, for example the scenario $P_{20} \rightarrow V_{20}$ (cf. red curve in the scenarios $D^T = V_{20}$ in Figure 6.9).

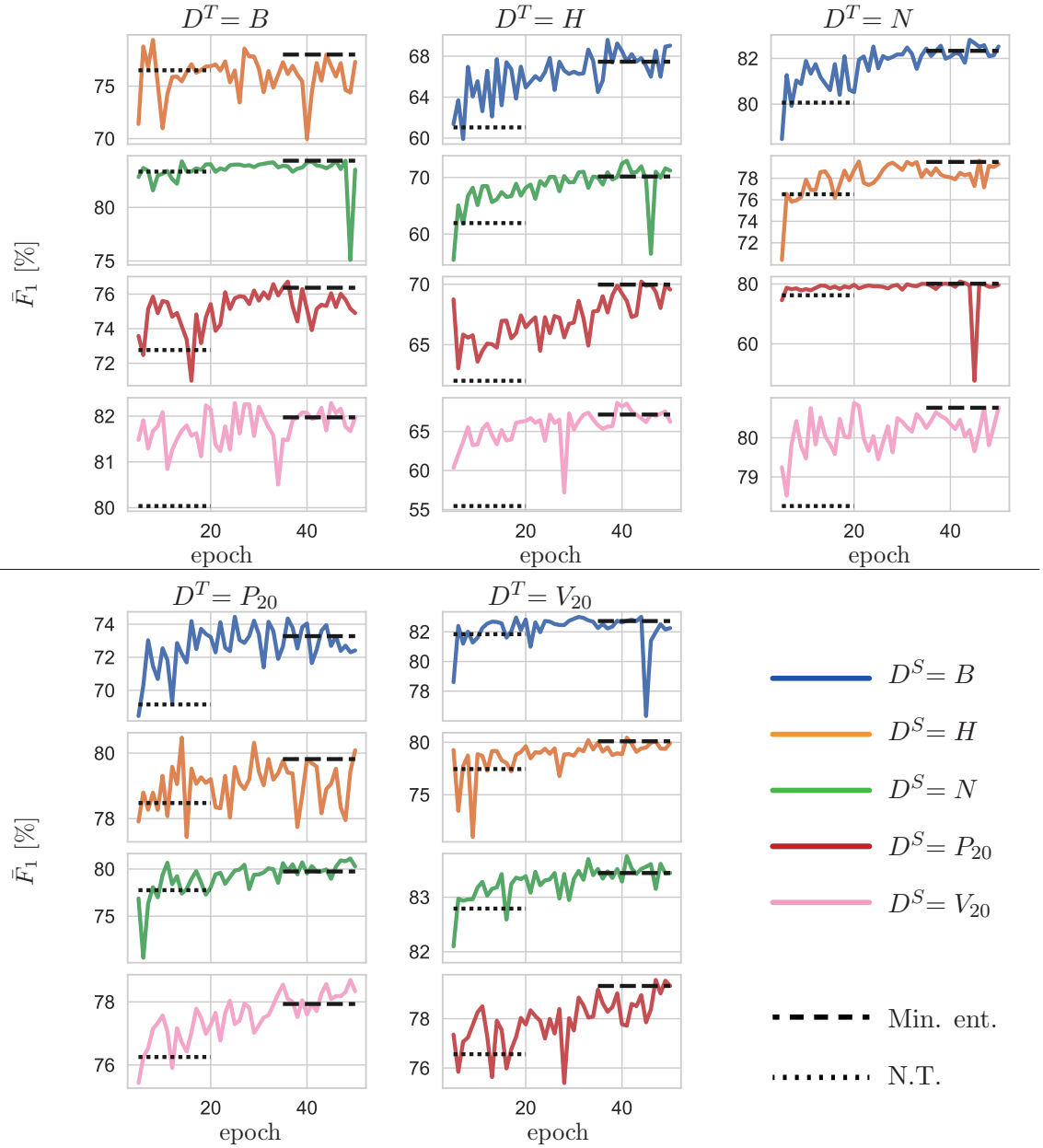

Figure 6.9: Evaluation of the stopping criterion. Coloured lines correspond to the performance of the classifier in D^T in the UDA process as a function of the training epoch. Broken black lines correspond to the performance if the entropy based stopping criterion is applied (Min. ent.), and the dotted black lines indicate the performance of naïve transfer (N.T.).

On the other hand, the scenario $B \rightarrow P_{20}$ (cf. blue curve in the lower left group in Figure 6.9) already shows a decreasing trend, which indicates that a longer training would lead to a further reduction of the performance if the last parameter state would be used for inference. This shows that setting a fixed number of training epochs is likely to result in a decreased performance for some scenarios. It can also be seen that the $\bar{F}_1$ score can drastically change from one epoch to the next. This is well visible in the scenario $N \rightarrow H$ (cf. green curve in the scenarios $D^T = H$ in Figure 6.9). Here, the F_1 scores show a positive trend during most of the adaptation process, but it drops to a very low value towards the end, just to recover immediately afterwards. If the parameter state in this epoch had been selected, the scenario would have resulted in a large negative transfer. The

same applies for three more scenarios, namely $N \rightarrow B$, $P_{20} \rightarrow N$ and $B \rightarrow V_{20}$. It is noted that in these four scenarios, the parameter state which was selected by the proposed selection criterion corresponds to an epoch earlier than the epoch in which the drop occurred.

When assessing the performance of the classifiers selected based on the entropy in D^T, all of them correspond to a positive transfer, i.e. the corresponding $\bar{F}_1$ score is higher than the one after naïve transfer. In most scenarios, a parameter state was selected that is close to the maximum performance tracked during the UDA process, although only in a a single case the very best epoch was selected. On average, i.e. considering the average over all scenarios, the classifiers selected by the proposed criterion achieve a performance of $\bar{F}_1 = 77.7\,\%$, which is only slightly better than the performance of the classifiers that would be obtained when using the last state that achieve a $\bar{F}_1$ score of $77.6\,\%$. In 11/20 scenarios using the states with the minimum entropy in D^T is better than the last states, it is worse in 7/20 scenarios. In the remaining two scenarios the last state corresponds to the one selected by the proposed method. Although the improvement is only marginal on average, the advantage of the method becomes clearer, when considering the worst case scenario. To that end, the lowest performance per scenario in the epochs 40-50 are averaged, resulting in a $\bar{F}_1$ score of $73.3\,\%$ on average, which is considerable lower than the result when using the proposed strategy for parameter selection ($\bar{F}_1 = 77.7\,\%$). Looking at the average performance of the classifiers when using the best parameter states, a $\bar{F}_1$ score of $78.5\,\%$ could have been achieved, which underlines that the method failed to pick the very best parameter state.

Addressing the fourth research question from Section 1.2, it can be concluded that the parameter selection criterion is better than training for a fixed number of epochs; whether or not the difference in the performance is considerable, depends on the scenario. In the scenario presented above, training for a fixed number of 50 epochs would result in a comparable performance. However, it was shown that considerable performance drops occur and if this would happen at the very last epoch, the difference would become larger. Thus, even though the proposed selection criterion, even though it fails to pick the very best parameter state in most cases, is well suited to pick a good epoch and particularly to avoid selecting an epoch that has a very poor performance.

6.4 Results of Experiment Set E4: Comparison to other Strategies and Methods

In this section, the results of the experiment sets E4.1 and E4.2 are reported. In the first set, the main strategy for UDA used in the proposed method will be compared to other strategies for UDA. In the second set, the proposed method is compared to other methods for UDA in RS from literature. In both sets, the variants V_{RD} and $V_{RD,ABN}$, which were selected in experiment set E2, are considered.

6.4.1 Experiment set E4.1: Comparison to other Strategies.

In this section, the main strategy for UDA proposed in this thesis, i.e. appearance adaptation with an optional extension of representation transfer, is compared to other strategies for UDA. Table 6.14 summarises the variants that are compared in this experiment set (cf. also Section 5.3.4).

Str.	Explanation	
N.T.	Naïve transfer	
ABN	Adaptive batch normalisation	
V_{RD}	Proposed method (Appearance adaptation)	variant V_{RD}
$V_{RD,ABN}$	Proposed method (Appearance adaptation + ABN)	variant $V_{RD,ABN}$
RT (e)	Representation matching following (Tsai et al., 2018)	(early layer)
RT (l)		(late layer)
RT (e+l)		(early and late layer)
IT (a)	Instance transfer following (Vu et al., 2019)	(adversarial alignment)
IT (d)		(direct minimisation)
IT (a+d)		(adversarial alignment and direct minimisation)
ID-ST	Performance in intra-domain setting	

Table 6.14: Overview over the different strategies and variants for UDA compared in the experiment set E4.1. For details, cf. Section 5.3.4. Str.: Strategy.

For a better overview, the individual results of all combinations of source and target domains are omitted. Instead, the average performance is compared. In Table 6.15 the $\bar{F}_1$ score for each source domain averaged over all target domains is presented for each variant. Table 6.16 shows the average $\bar{F}_1$ score for each target domain averaged over all source domains for each variant.

The comparison in Table 6.15 shows that one of the variants of the proposed method achieves the best average performance for all source domains. For $D^S = B$, using ABN performs very well. Here, the variant $V_{RD,ABN}$ achieves the highest $\bar{F}_1$ score, followed by only applying ABN and the variant V_{RD}. In the scenarios in which $D^S = H$, the variant V_{RD} performs best, being the only variant that achieves an positive transfer on average. In these scenarios the representation transfer based approach in which the representations of an early layer are aligned performs second best and the variants that use adversarial instance transfer achieve a comparable performance, yet, all result in a negative transfer on average. The variant that uses direct entropy minimization results in a rather large negative transfer of $-8.9\,\%$ with respect to the $\bar{F}_1$ score.

It is also noted that using only ABN to perform UDA works rather well. For the source domains B, N and V, ABN achieves the third best results. For the source domain P_{20}, ABN performs relatively badly, but still results in an average improvement of $1.8\,\%$ in $\bar{F}_1$. Only for the source domain H, ABN results in a negative transfer ($-1.4\,\%$) on average. The combination of appearance adaptation and ABN in variant $V_{RD,ABN}$ outperforms ABN for all source domains.

Str. \ D^S	B	H	N	P_{20}	V_{20}	Avg.
N.T.	73.6±0.8	77.5±0.6	76.6±0.4	72.3±0.6	72.8±0.5	74.6±0.4
ABN	77.3 (3.7)	76.1 (−1.4)	78.3 (1.7)	74.1 (1.8)	76.0 (3.2)	76.4 (1.8)
V_{RD}	76.5 (2.9)	**79.0** (1.5)	**79.6** (3.0)	76.2 (3.9)	77.2 (4.3)	77.7 (3.1)
$V_{RD,ABN}$	**78.5** (4.9)	76.5 (−1.1)	**79.6** (3.0)	**76.6** (4.2)	**77.9** (5.1)	**77.8** (3.2)
RT (e)	74.8 (1.2)	77.3 (−0.3)	77.2 (0.5)	73.6 (1.3)	73.4 (0.5)	75.2 (0.7)
RT (l)	76.1 (2.5)	74.9 (−2.7)	76.0 (−0.6)	74.6 (2.3)	73.9 (1.1)	75.1 (0.5)
RT (e+l)	76.2 (2.6)	75.4 (−2.1)	76.5 (−0.1)	74.9 (2.6)	75.3 (2.5)	75.7 (1.1)
IT (a)	75.2 (1.6)	77.2 (−0.3)	75.8 (−0.9)	74.2 (1.8)	74.4 (1.6)	75.4 (0.8)
IT (d)	74.8 (1.2)	68.7 (−8.9)	76.9 (0.3)	73.6 (1.2)	73.1 (0.3)	73.4 (−1.2)
IT (a+d)	75.5 (1.8)	77.0 (−0.6)	76.8 (0.2)	74.8 (2.4)	74.0 (1.2)	75.6 (1.0)

Table 6.15: Comparison of different strategies for UDA. Average $\bar{F}_1$ score in % by source domain after adaptation to each target domain (averaged over the target domains). Values in parentheses correspond to the improvements compared to naïve transfer (N.T.). Str.: Strategy. Avg.: Average $\bar{F}_1$ score by source domain. For the results of naïve transfer, the standard deviation over five training runs are reported.

Str. \ D^T	B	H	N	P_{20}	V_{20}	Avg.
N.T.	78.4±0.2	60.3±1.2	78.6±0.5	75.9±0.3	79.9±0.2	74.6±0.4
ABN	78.5 (0.1)	65.8 (5.5)	78.5 (−0.1)	**79.1** (3.2)	79.9 (0.1)	76.4 (1.8)
V_{RD}	**79.9** (1.5)	69.4 (9.1)	**80.7** (2.1)	77.5 (1.6)	**81.1** (1.2)	77.7 (3.1)
$V_{RD,ABN}$	79.5 (1.1)	**70.5** (10.2)	79.3 (0.8)	**79.1** (3.2)	80.7 (0.8)	**77.8** (3.2)
RT (e)	79.1 (0.8)	61.7 (1.4)	79.5 (1.0)	75.7 (−0.1)	80.1 (0.3)	75.2 (0.7)
RT (l)	79.6 (1.2)	60.9 (0.6)	77.9 (−0.6)	76.8 (1.0)	80.4 (0.6)	75.1 (0.5)
RT (e+l)	79.7 (1.4)	62.4 (2.1)	78.5 (−0.0)	77.1 (1.2)	80.6 (0.7)	75.7 (1.1)
IT (a)	79.0 (0.7)	61.1 (0.8)	79.4 (0.8)	76.7 (0.9)	80.6 (0.7)	75.4 (0.8)
IT (d)	70.0 (−8.4)	61.1 (0.8)	79.3 (0.7)	76.6 (0.8)	80.0 (0.2)	73.4 (−1.2)
IT (a+d)	79.5 (1.1)	62.1 (1.8)	79.3 (0.7)	77.0 (1.1)	80.1 (0.3)	75.6 (1.0)
ID-ST	87.1±0.2	80.3±0.2	85.5±0.2	89.1±0.1	84.3±0.2	85.3±0.2

Table 6.16: Comparison of different strategies for UDA. Average $\bar{F}_1$ score in % by target domain after adaptation from each source domain (averaged over the source domains). Values in parentheses correspond to the improvements compared to naïve transfer (N.T.). Str.: Strategy. Avg.: Average $\bar{F}_1$ score by target domain. ID-ST: Performance of a classifier trained in the respective target domain. For the results of naïve transfer, the standard deviation over five training runs are reported.

An analysis of Table 6.16 shows that for all target domains, both variants of the proposed method achieve the highest scores. In the two sets $D^T = H$ and $D^T = P_{20}$, the variant $V_{RD,ABN}$ outperforms the variant without ABN. For $D^T = P_{20}$, using only ABN performs also very well, but for $D^T = H$ the strategy of only using ABN is considerably worse compared to the variant $V_{RD,ABN}$. In this set of scenarios, i.e. for $D^T = H$, the proposed method outperforms the other strategies by the largest margin of up to 10 % in the $\bar{F}_1$ score. The difference of the performance using variant $V_{RD,ABN}$ for the target domain P_{20} to most other approaches is also considerable with a difference of $2 - 3$ % in the average $\bar{F}_1$ score. For the other target domains, the proposed method is only slightly better than the other strategies.

The results indicate that the proposed strategy is superior to simple variants of representation transfer and instance transfer in most scenarios and both variants outperform such strategies on average. However, there is no clear winner when comparing the two variants of the proposed method, which was already discussed in Section 6.2.4.

6.4.2 Experiment Set E4.2: Comparison to other Methods

In this section, the proposed method for UDA in the variants V_{RD} and $V_{RD,ABN}$ is compared to three methods from the literature by applying them to the same adaptation scenario on which these methods were evaluated in the original publications. The results are presented in Table 6.17.

		[%]		class-wise F_1 score [%]					
	UDA	OA	$\bar{F}_1$	SG	BU	LV	HV	VH	CL
$P'_5(RGB) \rightarrow V'_9(IRG)$									
I)	N	n.r.	32	n.r.					
	Y		49						
Prop.	N	60.4	52.8	63.5	74.9	31.4	75.5	63.5	8.1
	V_{RD}	65.1	56.9	59.5	75.4	43.8	**77.6**	62.4	12.7
	$V_{RD,ABN}$	**68.0**	**60.5**	**69.4**	**84.5**	**45.9**	72.1	**67.1**	**23.8**
$P'_5(IRG) \rightarrow V'_9(IRG)$									
II)	N	52.6	38.8	51.0	61.2	28.6	69.5	14.0	8.2
	Y	68.2	58.5	72.9	73.3	58.9	73.7	54.0	18.2
III)	N	n.r.	44.3	60.6	72.4	39.6	66.4	22.5	4.4
	Y		65.8	81.5	87.4	61.2	77.2	73.0	14.2
Prop.	N	76.3	66.5	79.8	87.5	60.7	79.3	68.7	22.9
	V_{RD}	77.7	67.8	81.1	90.3	58.6	**79.4**	68.9	28.9
	$V_{RD,ABN}$	**78.8**	**70.3**	**82.6**	**90.1**	**64.4**	79.2	**73.4**	**31.8**

Table 6.17: Comparison to I) (Benjdira et al., 2019), II) (Ji et al., 2020) and III) (Zhao et al., 2023) based on UDA between Potsdam and Vaihingen. Column UDA indicates whether the results were achieved with UDA (Y) or without UDA (N), the latter case indicating that the source classifier was applied to $\mathcal{D}^T$ without any adaptation (naïve transfer). Prop.: Variant of the UDA method proposed in this thesis. n.r.: not reported in the original publication.

It can be seen that using the proposed method for source training already leads to better performing classifiers compared to the classifiers that were adapted using the methods proposed in (Benjdira et al., 2019), (Ji et al., 2020) and (Zhao et al., 2023). In the first setting it can be argued that the strategy for dealing with different GSDs in D^S and D^T has a major impact on the performance. In particular, in the proposed method, the source training was performed on the resampled dataset P'_9 instead of P'_5 as done in (Benjdira et al., 2019). This is assumed to drastically decrease the initial difference between the domains and, thus, to lead to the better performance in D^T without UDA. Zhao et al. (2023) also perform an explicit resampling considering the GSD in both domains during UDA, but not in their strategy for naïve transfer, which explains the poor initial performance of $\bar{F}_1 = 44.3\,\%$ reported in that work. However, when comparing to (Ji et al., 2020),

the initial performance in D^T after source training is also much higher when using the proposed method although Ji et al. (2020) resample the data from D^S to a GSD of 10 cm, which is close to the GSD in D^T. It is assumed that the higher performance of the proposed method in D^T after source training is caused by the radiometric and geometric data augmentation used in this thesis. In particular, Ji et al. (2020) perform only random cropping without rotation and no radiometric augmentation, whereas in the proposed method the images are randomly rotated and augmented radiometrically. This has a major impact on the initial performance, as shown in (Wittich, 2020). Besides, the choice of the architecture for the classifier and the loss function may affect the initial performance of the classifier in D^T.

As reported in the respective publications, the classifiers from (Benjdira et al., 2019), (Ji et al., 2020) and (Zhao et al., 2023) could further be improved using UDA. The same applies for the classifiers adapted using the proposed method. Here, the absolute improvement with respect to the global metrics is smaller compared to the improvements reported in (Benjdira et al., 2019), (Ji et al., 2020) and (Zhao et al., 2023). However, this is to be expected because the initial classifiers already perform much better even before applying UDA. In both adaptation scenarios, the F_1 score for the class *high vegetation* is slightly worse after adaptation when using the variant $V_{RD,ABN}$, while it could be improved when using the variant V_{RD}. For all other classes, the variant $V_{RD,ABN}$ achieves the best F_1 scores, clearly outperforming the methods from the literature by quite a large margin. The variant $V_{RD,ABN}$ also outperforms the variant V_{RD}. Considering the findings from the previous experiment this is to be expected, because the marginal label distributions in the two domains are not too different, in which case ABN seems to work very well.

Lastly, it has to be highlighted that both variants of the proposed method for UDA resulted in a positive transfer in the inhomogeneous UDA setting, i.e. the setting in which the RGB data are used in D^S, whereas in D^T, the data are the IRG images. This means that the method could cope with a scenario in which different features are available in the two domains, i.e. in an inhomogeneous UDA setting. An example for the appearance adaptation is given in Figure 6.10.

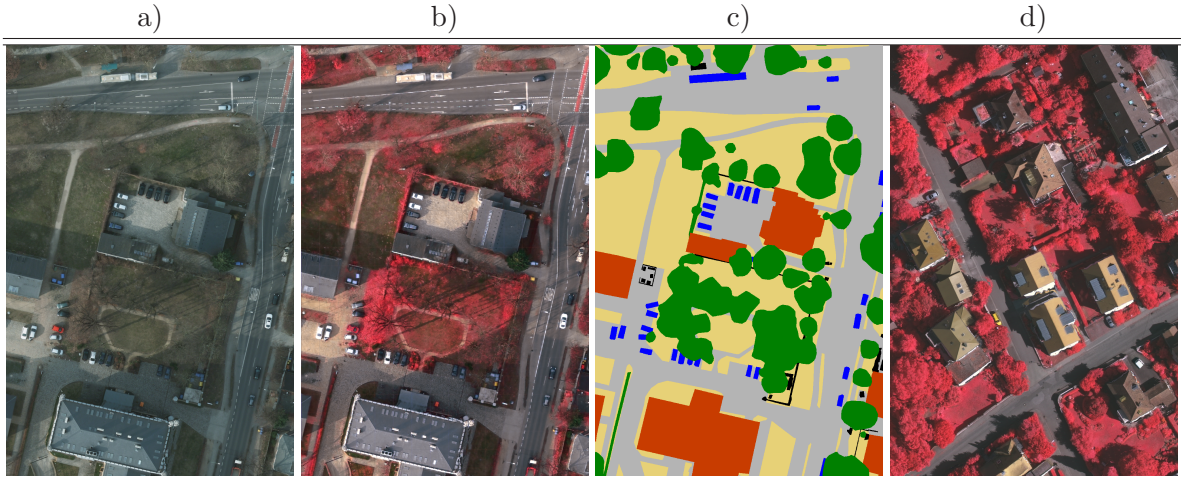

Figure 6.10: Example for image adaptation from RGB to IRG. a) Image from $D^S(P_{20})$ b) Image adapted to appearance from $D^T(V_{20})$ c) Reference label map. Class structure as in the caption of Figure 5.2 d) Exemplary image from $D^T$to visualize the appearance of various objects in that domain.

A qualitative assessment of the appearance adaptation reveals that the images from D^S are adapted in a meaningful way, because they give an impression similar to images in D^T and seem to be semantically consistent. In particular, the areas that correspond to *high vegetation* and *low vegetation* are changed in the appearance adaptation. The trees appear with a saturated red colour, which is representative for healthy vegetation in the false-colour images as the red-channel corresponds to the infra-red band. From the results of this experiment it is deduced that to some extent the proposed method can also deal with inhomogeneous UDA scenarios.

6.5 Results of Experiment Set E5: Evaluation of UDA for Deforestation Detection

In this experiment set, the proposed method for UDA, i.e. the variants V_{RD} and $V_{RD,ABN}$, are evaluated for the application of deforestation detection. Following (Soto et al., 2021), the F_1 score for the class $deforestation$, denoted by $F_{1,D}$, is used to assess the performance of the classifiers after source training and after UDA.

Source Training and Naïve Transfer: First, the results of source training are presented and discussed. Following (Soto et al., 2021), only one variant of source training is used to evaluate both, the intra domain performance but also the initial cross-domain performance corresponding to the naïve transfer approach. That means that no source training variant is used that is optimized for the cross-domain evaluation. The results are summarised in Table 6.18.

D^S \ D^T	MA	PA	RO
MA	92.0±0.4	45.2±1.7	34.3±2.4
PA	73.2±4.5	74.1±1.6	50.6±2.8
RO	45.4±10.3	30.8±4.6	81.0±2.0

Table 6.18: $F_{1,D}$ in % obtained for the test set in D^T after supervised training on the training set of D^S and using the validation set of D^S for parameter selection. The reported values are average and standard deviation of the $F_{1,D}$ scores over five training runs.

For the three domains, there is a rather large difference in the intra-domain performance (cf. elements on the main-diagonal in Table 6.18). The performance in MA is the highest, achieving $F_{1,D} = 92.0\,\%$. The scores in PA and RO are about 18 % and 11 % lower, respectively.

Comparing the cross-domain performance (cf. off-diagonal elements in Table 6.18) to the intra-domain performance it can be seen that there is a major drop in the performance, and the respective standard deviations are quite high. This indicates that the classifiers trained in each domain tend to overfit to the respective domain and show a poor generalisation capability to the other domains. This can be explained with the fact that the datasets are relatively small (cf. Table 5.4).

Unsupervised domain adaptation: The results of UDA using V_{RD} are presented in Table 6.19, and Table 6.20 shows the result of UDA using the variant $V_{RD,ABN}$. Using either variant for UDA, a positive transfer is achieved in all adaptation scenarios, except for the scenario $MA \rightarrow RO$, in which the variant V_{RD} results in a negative transfer of $-0.4\,\%$ in $F_{1,D}$. In this adaptation scenario, two of the five repetitions resulted in a considerable negative transfer of up to $-15\,\%$ in $F_{1,D}$ while in the other three training runs a positiver transfer was achieved. This leads to the high standard deviation in the $F_{1,D}$ score which indicates that the method is not very stable in this adaptation scenario. The same applies for the variants $PA \rightarrow MA$ and $PA \rightarrow RO$, in which the standard deviations are also relatively high. However, in those two scenarios no negative transfers occurred in the individual training runs.

D^S \ D^T	MA	PA	RO
MA	-	65.7±0.8 (20.5)	42.1±13.1 (7.7)
PA	85.7±4.4 (12.5)	-	61.0±9.4 (12.7)
RO	85.6±6.4 (40.2)	52.4±3.2 (21.6)	-
ID-ST	92.0±0.4	74.1±1.6	81.0±2.0

Table 6.19: $F_{1,D}$ scores in % obtained for the test set in D^T after source-training in D^S and adapting to D^T using the variant V_{RD}. The last row presents the results of a classifier trained in D^T (cf.elements on the main diagonal in Table 6.18). The reported values are the averages and standard deviations over five training runs. The values in parentheses correspond to the improvement compared to applying naïve transfer (cf. off-diagonal elements in Table 6.18).

D^S \ D^T	MA	PA	RO
MA	-	61.4±0.6 (16.2)	61.3±8.2 (27.0)
PA	89.7±0.6 (16.5)	-	64.2±5.9 (15.9)
RO	90.6±0.9 (45.1)	59.3±4.1 (28.5)	-
ID-ST	92.0±0.4	74.1±1.6	81.0±2.0

Table 6.20: $F_{1,D}$ scores in % obtained for the test set in D^T after source-training in D^S and adapting to D^T using the variant $V_{RD,ABN}$. The structure of the table corresponds to the one of Table 6.19.

The variant $V_{RD,ABN}$ also shows quite high standard deviations in some adaptation scenarios, but in most cases the average performance is higher than the average performance that was achieved when using V_{RD}. The only exception is the scenario $MA \rightarrow PA$, in which the variant V_{RD} performed better ($4.3\,\%$ in $F_{1,D}$ on average).

Averaged over all adaptation scenarios, the variant $V_{RD,ABN}$ resulted in considerable improvements compared to applying naïve transfer with an increase between $16.2\,\%$ and $45.1\,\%$ in the $F_{1,D}$ score and an average improvement of $24.9\,\%$. In comparison, the average improvement when using

V_{RD} is 19.2 % which is considerably lower but still quite high compared to the improvements that were achieved for the application of land cover classification. Comparing the two variants, the variant $V_{RD,ABN}$ is clearly preferable in this application, because it achieves a better performance on average and does not result in any negative transfer. The performance in the scenarios $PA \rightarrow MA$ and $RO \rightarrow MA$ when adapting using the variant $V_{RD,ABN}$ is considered to be even comparable to the performance that is obtained when training a classifier in D^T with a remaining performance gap of less than 2.3 % in the $F_{1,D}$ score. When comparing this value to the size of the initial performance gap when applying naïve transfer, i.e. of 46.6 % in $F_{1,D}$ in the scenario $RO \rightarrow MA$, it can be concluded that the method almost fully bridges the performance gap. On the other hand, the remaining performance gap for the target domain PA is still about 15 % and for RO it is about 20 % in $F_{1,D}$. Thus, in those scenarios, the method can bridge the performance gap only partially, but still to a large extent.

In Figure 6.11, exemplary predictions made by the adapted classifiers are shown.

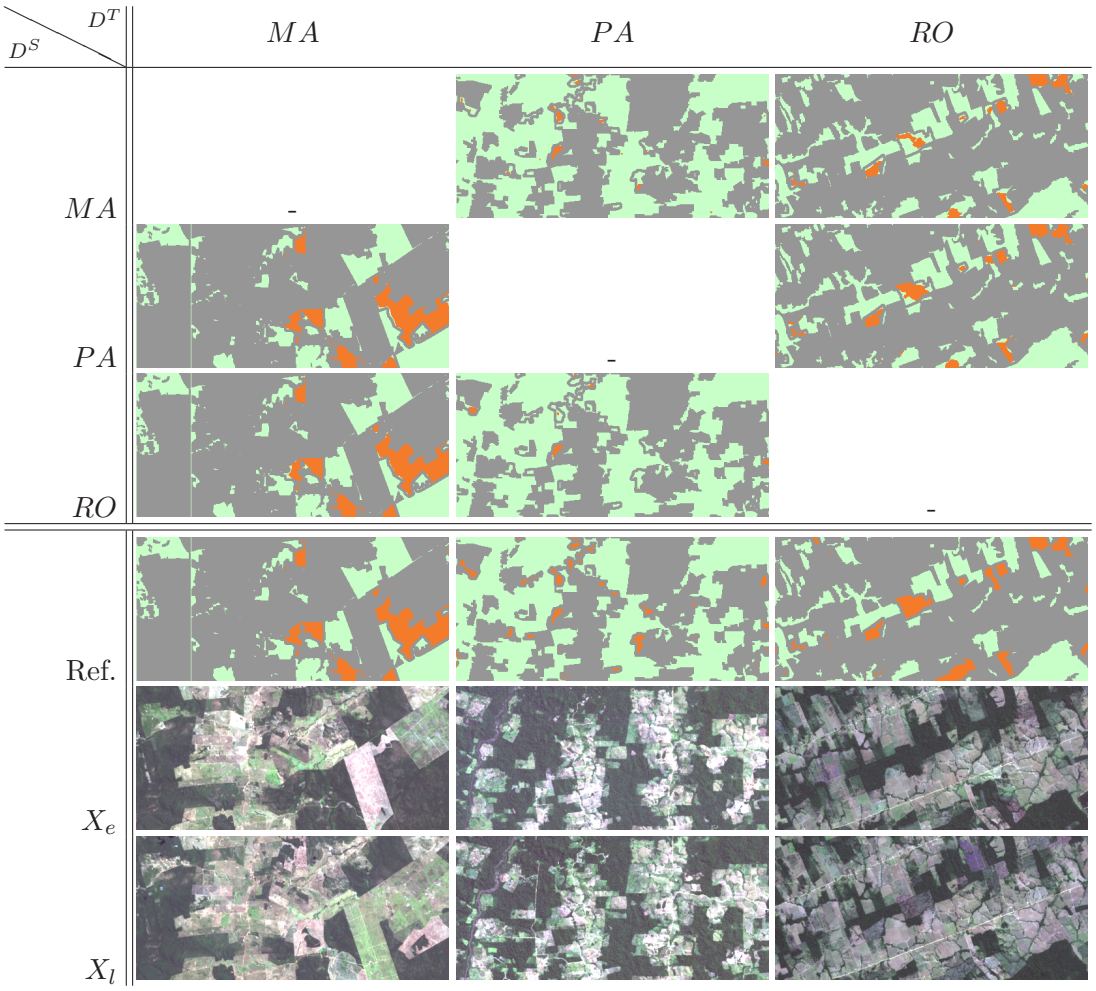

Figure 6.11: Exemplary predictions made by the adapted classifiers. Rows 1-3 show the predictions by the classifiers that were trained in the respective source domain and adapted to the target domain using variant $V_{RD,ABN}$. The fourth row shows the corresponding reference. The last two rows show the RGB channels of the earlier image (X_e) and later image (X_l) for the patches. Colour-code as in Figure 5.3. Note that the padded areas are also visualised in grey colour. Ref: Reference.

On a first glance, the predicted label maps look very similar to the reference maps. The label maps predicted for PA miss some *deforestation* areas completely, in particular when adapting from RO. In RO, there is one region that is falsely predicted as *no deforestation* to a small extent by the classifier adapted from PA and to a larger extent by the classifier adapted from MA. In total, there are no obvious false positive predictions for the class $deforestation$.

Based to these results it is concluded that the proposed method for UDA does not only perform well for the application of land cover classification based on aerial imagery, but also for the application of deforestation detection based on satellite imagery. Regarding the first scientific question, posed in Section 1.2, it can be said that for the application of bi-temporal deforestation detection the proposed method, i.e. the variant $V_{RD,ABN}$, does achieve a stable positive transfer and can reduce the performance gap by a large amount, yet not completely in all scenarios.

Comparison to Soto et al. (2021): In this comparison the domain RO is replaced by RO_0 because this domain was used in (Soto et al., 2021). Note that in this comparison, only the variant $V_{RD,ABN}$ is evaluated, because it performed better for the application of bi-temporal deforestation detection. The results are summarised in Table 6.21.

			Proposed			(Soto et al., 2021)	
	D^S \ D^T	MA	PA	RO_0	MA	PA	RO_0
S.T./N.T.	MA	**92.0**	45.2	24.9	85.5	**70.7**	**47.9**
	PA	**73.2**	74.1	**30.2**	41.7	**83.2**	20.5
	RO_0	59.4	32.0	64.7	**60.3**	**35.5**	**67.0**
UDA	MA	-	61.4	**47.2**	-	**72.0**	45.4
	PA	**89.7**	-	44.7	48.0	-	**48.0**
	RO_0	**87.2**	48.4	-	67.0	**51.0**	-

Table 6.21: Comparison of the method proposed in this work to (Soto et al., 2021). $F_{1,D}$ score in % obtained for the test set in D^T 1) after source training (S.T.) in D^S (upper three rows) and 2) after source-training in D^S and adapting to D^Tusing UDA (lower three rows). The off-diagonal elements in the upper matrices correspond to the results of naïve transfer (N.T.). The highest performance for each scenario is printed in bold font.

After source training, the classifiers achieve quite heterogeneous results in both the intra-domain (D^S=D^T), and the cross-domain evaluation (D^S≠D^T). The classifier trained on MA (first row in Table 6.21) using the proposed method achieves a higher score ($F_{1,D} = 92.0\,\%$) when evaluated on the same domain compared to (Soto et al., 2021) ($F_{1,D} = 85.5\,\%$). However, if the classifiers are evaluated in the other domains those trained in (Soto et al., 2021) perform much better (about 25 % in $F_{1,D}$ in both cases). The initial performance of the classifiers trained on RO_0 using both methods achieve similar $F_{1,D}$ scores for all three target domains, but the classifier trained in (Soto et al., 2021) is slightly better in all three cases. When training on PA the proposed method achieves a lower intra-domain score compared to (Soto et al., 2021), but performs better when applied to

the other two domains. When using the proposed method, the intra-domain performance is lower compared to (Soto et al., 2021), but the cross-domain performance is higher in both target domains. On average, the performance after source training is not too different between both methods. The proposed method achieves an average cross-domain performance of $F_{1,D} = 44.2\,\%$ while Soto et al. (2021) report $F_{1,D} = 46.1\,\%$. The average intra-domain performance using the proposed method is $F_{1,D} = 76.9\,\%$ and Soto et al. (2021) report $F_{1,D} = 78.6\,\%$. Thus, the initial performance using the proposed method is about 2 % worse compared to Soto et al. (2021). In this context, it has to be noted that no hyper-parameter tuning for source training in terms of optimising the performance on a validation set of either domain was performed for the application of deforestation detection. The only changes in the hyper-parameters are based on preliminary experiments on related datasets or adapted from (Soto et al., 2021).

When assessing the performance of the classifiers after UDA, the results are again very heterogeneous. The performance reported in (Soto et al., 2021) is higher for the scenarios $MA \rightarrow PA$, $RO_0 \rightarrow PA$ and $PA \rightarrow RO_0$, but in the remaining three scenarios the proposed method obtains better results in terms of $F_{1,D}$. In the scenarios $MA \rightarrow RO_0$, $RO_0 \rightarrow PA$ and $PA \rightarrow RO_0$, the difference is rather small, with a maximum difference of 3.3 % with respect to $F_{1,D}$. In the setting $MA \rightarrow PA$, the method by Soto et al. (2021) outperforms the proposed method by about 10 % in $F_{1,D}$. The largest differences can be observed in the scenarios $RO_0 \rightarrow MA$ and $RO_0 \rightarrow MA$, where the proposed method performs better by about 20 % and 40 % in the $F_{1,D}$ score, respectively. On average over all adaptation scenarios, this leads to a higher performance, with a $F_{1,D}$ of 63.1 % using the proposed method against 55.2 % achieved by the method from (Soto et al., 2021). It is concluded that the proposed method strongly outperforms the method from (Soto et al., 2021) in two settings and the average performance is also higher by about 8 % in the $F_{1,D}$ score. Furthermore, applying the method by Soto et al. (2021) results in one negative transfer in the scenario $MA \rightarrow RO_0$. However, as the method by Soto et al. (2021) resulted in a better performance in half of the scenarios, there is no clear winner in the comparison.

Lastly, it is again noted that besides the change in the architecture of the classifier, the batch-size, the patch size and the learning rate related to $\mathcal{C}$, all hyper-parameters of UDA are based on the tuning for land cover classification. Using additional settings for tuning the method to this application is assumed to result in a better performance of UDA. However, tuning the method would require an additional pair of source and target domains, which was not available at this point.

7 Conclusions and Outlook

7.1 Conclusion

In this thesis, the problem of deep unsupervised domain adaptation for the pixel-wise classification of images was addressed with a focus on applications from remote sensing. A method for deep UDA was proposed that is based on the strategy of appearance adaptation with an optional extension using adaptive batch normalisation, a method for UDA from the literature. For appearance adaptation, a new training scheme was proposed that is based on jointly training an appearance adaptation network and the actual classification network. This training scheme aims to perform semantically consistent appearance adaptation, which is required for successfully adapting a classifier to the target domain. Two methods were proposed that build upon the joint training scheme and aim to further improve the semantic consistency in scenarios in which differences related to the label distributions in both domains pose additional challenges. The first method is based on regularising the output of the domain discriminator network and the second method corresponds to an extension of the architecture using an auxiliary image generation network. To further improve the performance of UDA, a parameter selection criterion was proposed, which aims to find a good parameter state during the adaptation process.

In the experiments, the method was evaluated on multiple adaptation scenarios and two different applications from remote sensing. It was shown that the joint training scheme can achieve a positive transfer on average (+1.8 % in the $\bar{F}_1$ score), evaluated on 20 different adaptation scenarios related to the task of land cover classification based on aerial imagery. However, some adaptation scenarios remained problematic when using this training scheme without further modifications, resulting in a negative transfer. An evaluation of the appearance adaptation revealed that only using the joint training scheme is not sufficient to achieve semantically consistent appearance adaptation in difficult scenarios, which points out the limitations of this training scheme. However, applying the two methods aiming at an improved semantic consistency was shown to alleviate those problems to a large extent, i.e. they could increase the semantic consistency, leading to an improved average performance of the classifiers after UDA and a reduction of the number of negative transfers. The method which is based on regularising the output of the domain discriminator network was found to be preferable over the variant that uses the auxiliary generator, because it resulted in a comparable improvement of the average performance, while at the same time avoiding a negative transfer in 96/100 cases. Due to the very low rate of negative transfers, this variant is considered to be the most stable one.

A combination of the proposed method for UDA based on appearance adaptation with adaptive batch normalisation was evaluated and it could be shown that this results in a slightly larger aver-

age performance of the adapted classifiers, evaluated on the 20 adaptation scenarios for land cover classification. Considering the average performance of the combined method, it could compensate for about one third of the initial performance gap, reducing it from 10.7 % to 7.5 % in $\bar{F}_1$ on average for the application of land cover classification. In several adaptation scenarios, the improvement of the classifier due to UDA was much higher, with up to 14.4 % with respect to the $\bar{F}_1$ score, and in a few scenarios, the performance of the classifiers after adaptation were actually comparable to the performance of the classifier trained in the target domain with a remaining performance gap of about 1 % in $\bar{F}_1$. However, in those scenarios the initial performance gap was rather small and the improvement due to UDA was only minor. On the other hand, when using the combination of appearance adaptation and ABN, considerable negative transfer occurred in two of the 20 adaptation scenarios. A detailed analysis of five adaptation scenarios showed that the extension by ABN seems to work well whenever the differences in the global label distributions of source and target domain are not too different. If this prerequisite is not fulfilled, the classifiers adapted using ABN will have problems to correctly predict classes that are overrepresented in one of the domains and that at the same time have an appearance that is similar to another class. Consequently, whether the performance of UDA using appearance adaptation can be improved when combining it with ABN depends on the adaptation scenario. The combined variant has a higher potential of reducing the performance gap but at the same time it is less stable than using only the proposed method for appearance adaptation without ABN.

The proposed method, i.e. the variants with and without ABN, were further evaluated for the application of bi-temporal deforestation detection based on satellite imagery. It was shown that both variants work well in this application, evaluated on three different datasets showing regions in Brazil. For this application, the combination of appearance adaptation and adaptive batch normalisation performed better than only using appearance adaptation in 4/6 adaptation scenarios and with respect to the average performance. Using the combined method, the classification performance was improved by 25.5 %, i.e. from 46.6 % to 71.1 %, on average in the F_1 score of the foreground class, thus, compensating for more than two thirds of the initial performance gap (−35.8 %). In one scenario, an even larger improvement of 45.1 % in the F_1 score of the foreground class was achieved, resulting in a performance comparable the one of a classifier that was directly trained in the target domain. In the other settings, the proposed method could also improve the classification performance by a large margin, which, however, to a large amount is related to the fact that the initial performance of naïve transfer was rather poor.

The evaluation in different adaptation scenarios also revealed some limitations of the method. For example, in some of the evaluated adaptation scenarios, trees without leaves only appear in the target domain, but not in the source domain. In those scenarios, the classes *high vegetation* were frequently confused with *low vegetation*, because the two classes look very similar in aerial images. Such classification errors could barely be prevented by the proposed method for UDA, resulting in a rather poor performance of the classes *high vegetation* and *low vegetation* after UDA. In general, if objects of one class in the target domain look more similar to objects of another class in the source domain than objects of the same class in the source domain, appearance adaptation will probably fail to compensate the domain gap.

Another contribution proposed in this thesis is a method for selecting the parameters of a classifier during UDA. The proposed method is rather simple and can be integrated in various methods for deep UDA. In a set of experiments it was shown that this method is superior to the common strategy of simply running the UDA for a fixed number of iterations. Although in the experiments, this strategy achieved only a minor improvement on average, it was shown that it can help to avoid selecting a very bad parameter state. Thus it is considered to be superior to the common strategy of simply running the UDA for a fixed number of iterations.

To further assess the proposed method, it was compared against several different strategies and methods for deep UDA from the literature. It was shown that the proposed method outperforms other adaptation strategies by at least 1.4 % in the average $\bar{F}_1$ score, evaluated on the task of land cover classification. Compared to three methods from the literature the combined method outperformed those from the literature by a rather large margin of about $5-12$ % w.r.t. the $\bar{F}_1$ score. In the corresponding experiments, it was also shown that the method can deal rather well with an inhomogeneous adaptation scenario, even though it was not explicitly developed for such a setting. The method was also compared to one that addresses deep UDA for bi-temporal deforestation detection. When comparing to the results reported for that method in the literature, the method proposed in this thesis achieved a comparable performance in most settings, but outperformed the method from the literature by a large margin in two adaptation scenarios and by about 8 % in the F_1 score of the foreground class on average over all adaptation scenarios.

To conclude, the contributions proposed in this thesis do not completely solve the main problem of missing labelled training data in any scenario, because in many cases the classification performance after adaptation is not at a satisfactory level. However, the contributions are considered as a step in the direction of reducing the performance gap as much as it is possible given the available training data. It was shown that in a few scenarios, the proposed method leads to a major improvement of the performance that is actually comparable to directly training in the target domain. In such a setting, the method does indeed remove the requirement for having access to labelled data in the target domain. However, it is not yet possible to automatically and reliably predict the performance of a classifier after UDA. Thus, until this is possible, manual intervention is required to some extent, for example to assess the similarity of two domains, to assess whether a domain adaptation was successful or to assess if the classification results after UDA are satisfactory for the addressed application.

Research Questions: The conducted experiments are considered to allow to answer all research questions posed in Section 1. The first, second and third questions asked how the different variants behave with respect to the stability, adaptation performance and where the limitations of the method are. These aspects were directly addressed in the variant comparisons. The variant without ABN but using the extension of the discriminator achieved the highest stability with almost no negative transfers. Combining the method with ABN resulted in a higher performance for both applications on average, but was shown to be limited to adaptation scenarios in which the global label distributions of the source and target domains are not too different. The fourth research question asked whether the proposed stopping criterion is better than using a fixed number of

epochs during UDA. Based on the results of the experiments, this question can be answered with yes. However, it is again noted that the stopping criterion failed to pick the very best parameter states. Nevertheless, it helped to avoid to use a very bad parameter state, which is the main reason why it is considered to be better than using a fixed number of epochs. In the experiments it was also shown that the method clearly outperforms several existing methods and strategies for the application of land cover classification and outperforms a method for UDA addressing deforestation detection with respect to the average performance, but not in all scenarios. This allows to answer the last research question, that asked how the method compares to approaches from the literature.

7.2 Outlook

The results of the experiments allowed to answer the initial research questions, but they also showed that not all problems could be solved by the proposed method. In the following paragraphs an outlook on possible extensions and future work is provided

Predicting and Assessing the Success of UDA: It the experiments, it was shown that in some scenarios the proposed method for UDA can almost fully bridge the domain gap while at the same time a very large performance gap remains in others. Having an automated method to reliably estimate the success of UDA, either before it is applied or after having adapted the classifier to the target domain, would help in this regard. Predicting the success of UDA in advance is very challenging because it depends on many factors, such as the similarity of the appearance of objects in the two domains and the similarity of the label distributions. In particular the latter one cannot be assessed if no labels are available in the target domain. An alternative approach is to assess the success of UDA after having adapted the classifier, i.e. by assessing the label maps predicted by the adapted classifier for the target domain. For example, some hand-crafted criteria could be applied, such as checking the height of buildings relative to the height of the ground, knowing that they are in general higher by a couple of meters. Of course, this has the disadvantage of requiring some knowledge about the target domain. Yet an alternative approach is to use a cyclic adaptation approach, i.e. to adapt a classifier from the source domain to the target domain before adapting it back to the source domain, an approach used e.g. in (Bruzzone and Marconcini, 2009). A consistency after one cycle is a strong indicator for the adaptation to have been successful, but does not guarantee it.

Further Improving the Semantic Consistency: Further potential for follow-up work is related to the proposed methods aiming to improve the semantic consistency. Both of these methods have been shown to achieve their goal, however, it was also shown that in some scenarios the semantic consistency was not perfect. One possible reason is that the hyper-parameters that are related to the extensions were set to the same values for all domains. This may not result in an optimal performance of those methods, as adaptation scenarios with a larger domain gap may require a different set of hyper-parameters than those with a smaller domain gap. Consequently, it is suggested to investigate the option of making the hyper-parameters adaptive to the performance of the

appearance adaptation, which could, for example, be measured by the classification performance of the domain discriminator. In that regard, much potential is seen in the extended adversarial training scheme using the auxiliary generator. This concept has been proven to work in principle as expected, even though it did not outperform the variant based on regulating the discriminator output. However, it is considered to be much more flexible in principle, because it can better adapt to the actual differences in the label distributions. It is noted that regarding the method using the auxiliary generator, there is plenty of room for evaluating different hyper-parameters and architectural design choices. For example, in the literature, several alternative loss functions for adversarial training were proposed, e.g. (Mao et al., 2017), that have been shown to be superior to the regular adversarial loss formulation for the task of image generation, but that were not evaluated in this thesis.

Improving Adaptive Batch Normalisation: In the experiments, it was shown that ABN has a high potential of further increasing the performance after UDA. At the same time, it was shown that ABN struggles in an adaptation scenario in which the global label distributions between source and target domain are very different. For future work, it is suggested to explicitly address this problem of ABN, for example by performing a sample selection process of images that are used to perform ABN, such that the label distribution of the selected samples is comparable to the one in the source domain. In an UDA setting, this requires to use semi-labels, which might be problematic if the initial performance of the classifier is rather poor. However, it was shown that the proposed method for appearance adaptation results in a rather stable improvement of the classifiers even in situations in which ABN results in a negative transfer. Thus, it is suggested to use the classifier after adaptation using the proposed method for appearance adaption to predict semi-labels, which then are used to improve the performance of ABN.

Parameter Selection: The proposed entropy based parameter selection criterion has been shown to be well suitable for avoiding to select a very bad set of parameters. Nevertheless, it has largely failed in selecting the very best set of parameters. This implies that the entropy is not a perfect measure for the performance in the target domain. One reason could be that the average entropy is lower in a constellation in which fewer objects, and, thus, fewer object boundaries are predicted, because the entropy is usually high at the boundaries of objects. This means that the entropy based parameter selection would prefer a parameter state in which fewer objects are predicted, even though the classification performance is actually worse. Future work should address this problem, for example by not considering the boundary areas in the predicted label maps when calculating the average entropy, which has been shown to be advantageous in the context of direct entropy minimisation in (Wittich, 2020).

Transition to other Training Scenarios: In future work, the proposed training scheme or individual aspects of the proposed method could also be transferred to different training scenarios. For example, the method could easily be transferred to a semi-supervised training scenario in which few labelled images in the target domain are available. The joint training scheme could be used in

the same way as in the UDA setting, but the classifier would be trained simultaneously using the existing labelled data in the target domain. Having few labelled data in the target domain could also be advantageous to better assess the differences in the label distributions, for example to decide whether to apply ABN as a subsequent adaptation step. Another aspect of the proposed method that could be transferred to another training scenario is the proposed use of an auxiliary generator. The extended adversarial training scheme could also be used for appearance adaptation without using the adapted images for domain adaptation. For example, the approach of CycleGAN could be extended. This is supposed to improve the consistency in situations in which the two datasets are very different with respect to the underlying class structure, which is of course given only implicitly in this training scenario. As a last example, the proposed entropy based parameter selection criterion could be transferred to many other settings, including supervised training. In particular in scenarios where the amount of labelled training samples is limited but an additional set of unlabelled samples is available, the proposed approach for parameter selection could be very useful, because it does only require a set of unlabelled samples. Consequently, the unlabelled samples could be used for parameter selection while all of the labelled data could be used for training.

Transition to other Applications: Although the proposed method was only evaluated on applications from remote sensing, it is not limited to this domain. In particular, the method can directly be applied to any adaptation scenario in which the task is the pixel-wise classification. With minor modifications, the approach could also be transferred to other image related task, such as object detection, panoptic segmentation or multi-task problems. In principle, any task can be used as long as the final loss can be propagated back to the input images. Besides, the approach could also be transferred to different types of input data, such as image sequences or even point clouds. Of course, this would require major changes of the architectures used, but it would not require to change the training scheme.

Embedding the Method in Active Learning: When it comes to the practical use of deep learning based approaches, it is often not yet possible to fully compensate for manually generated labels. The research field of *active learning* addresses this fact by developing strategies that minimise the required human effort in the labelling process. Usually, this is done in an iterative procedure. In the first step, the operator will provide some annotations which are used to train a classifier. In the second step, the trained model is used to make predictions and aims to identify difficult samples. Then the operator will label those samples, and the classifier is re-trained. This process is repeated until a satisfactory performance is achieved. The proposed strategy could be integrated in such a procedure as intermediate adaptation step after the model has been re-trained. It is assumed that this would reduce the number of iterations required to achieve a satisfactory performance.

Acknowledgements

I would like to thank the *Landesamt für Vermessung und Geoinformation Schleswig Holstein* and the *Landesamt für Geoinformation und Landesvermessung Niedersachsen (LGLN)* for providing the datasets Schleswig, Hameln, Nienburg, Buxtehude and Hannover. I also thank the International Society for Photogrammetry and Remote Sensing (ISPRS) for providing the data of the ISPRS labelling challenge. The Vaihingen dataset was provided by the German Society for Photogrammetry, Remote Sensing and Geoinformation (DGPF) (Cramer, 2010): `http://www.ifp.uni-stuttgart.de/dgpf/DKEP-Allg.html`.

I would also like to thank Franz Rottensteiner for the excellent supervision, the numerous scientific discussions and the constant support. Further thanks go to Christian Heipke for the excellent management of the institute and to all the colleagues who made me enjoy going to work every day.

Bibliography

Arazo, E., Ortego, D., Albert, P., O'Connor, N. E. and McGuinness, K., 2020. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In: *IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8.

Assis, L. F., Ferreira, K. R., Vinhas, L., Maurano, L., Almeida, C., Carvalho, A., Rodrigues, J., Maciel, A. and Camargo, C., 2019. TerraBrasilis: a spatial data analytics infrastructure for large-scale thematic mapping. *ISPRS International Journal of Geo-Information* pp. 513–540. Vol. 8(11).

Benaim, S., Galanti, T. and Wolf, L., 2018. Estimating the success of unsupervised image to image translation. In: *European Conference on Computer Vision (ECCV)*, pp. 218–233.

Benjdira, B., Bazi, Y., Koubaa, A. and Ouni, K., 2019. Unsupervised domain adaptation using generative adversarial networks for semantic segmentation of aerial images. *Remote Sensing* 11(11), pp. 1369–1392.

Breiman, L., Friedman, J. H., Olshen, R. A. and Stone, C. J., 1984. *Classification and regression trees*. Chapman and Hall, London, UK.

Bruzzone, L. and Marconcini, M., 2009. Domain adaptation problems: A DASVM classification technique and a circular validation strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(5), pp. 770–787.

Bruzzone, L., Chi, M. and Marconcini, M., 2006. A novel transductive SVM for semisupervised classification of remote-sensing images. *IEEE Transactions on Geoscience and Remote Sensing* 44(11-2), pp. 3363–3373.

Chang, W.-L., Wang, H.-P., Peng, W.-H. and Chiu, W.-C., 2019. All about structure: Adapting structural information across domains for boosting semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1900–1909.

Chen, M., Xue, H. and Cai, D., 2019a. Domain adaptation for semantic segmentation with maximum squares loss. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2090–2099.

Chen, Y.-C., Lin, Y.-Y., Yang, M.-H. and Huang, J.-B., 2019b. CRDOCO: Pixel-level domain transfer with cross-domain consistency. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1791–1800.

Chen, Y., Li, W., Chen, X. and Gool, L. V., 2019c. Learning semantic segmentation from synthetic data: a geometrically guided input-output adaptation approach. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1841–1850.

Cheng, Z., Chen, C., Chen, Z., Fang, K. and Jin, X., 2021. Robust and high-order correlation alignment for unsupervised domain adaptation. *Neural Computing and Applications* 33(12), pp. 6891–6903.

Choi, J., Kim, T. and Kim, C., 2019. Self-ensembling with GAN-based data augmentation for domain adaptation in semantic segmentation. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 6829–6839.

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1251–1258.

Cohen, J. P., Luck, M. and Honari, S., 2018. Distribution matching losses can hallucinate features in medical image translation. In: *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, Vol. Vol. 21, pp. 529–536.

Cortes, C. and Vapnik, V., 1995. Support-vector networks. *Machine learning* 20(3), pp. 273–297.

Cox, D. R., 1958. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B* 20(2), pp. 215–232.

Cramer, M., 2010. The DGPF test on digital aerial camera evaluation. *Photogrammetrie Fernerkundung Geoinformation* 2(2010), pp. 73–82.

Csurka, G., 2017. A comprehensive survey on domain adaptation for visual applications. In: *Domain Adaptation in Computer Vision Applications*, Springer, pp. 1–35.

Csurka, G., 2020. Deep visual domain adaptation. In: *22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pp. 1–8.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255.

Du, L., Tan, J., Yang, H., Feng, J., Xue, X., Zheng, Q., Ye, X. and Zhang, X., 2019. SSF-DAN: Separated semantic feature based domain adaptation network for semantic segmentation. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 982–991.

Engel, J. H., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C. and Roberts, A., 2019. Gansynth: Adversarial neural audio synthesis. In: *International Conference on Learning Representations (ICLR)*.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M. and Lempitsky, V., 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research* 17(1), pp. 2096–2030.

Gatys, L. A., Ecker, A. S. and Bethge, M., 2016. Image style transfer using convolutional neural networks. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2414–2423.

Gong, R., Li, W., Chen, Y. and Gool, L. V., 2019. DLOW: Domain flow for adaptation and generalization. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2477–2486.

Goodfellow, I. J. and Vinyals, O., 2015. Qualitatively characterizing neural network optimization problems. In: *International Conference on Learning Representations (ICLR)*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. In: *Advances in Neural Information Processing Systems*, Vol. 27, pp. 2672–2680.

Gritzner, D. and Ostermann, J., 2020. Using semantically paired images to improve domain adaptation for the semantic segmentation of aerial images. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Science*, Vol. V-2, pp. 483–492.

Guo, R., Liu, J., Li, N., Liu, S., Chen, F., Cheng, B., Duan, J., Li, X. and Ma, C., 2018. Pixel-wise classification method for high resolution remote sensing imagery using deep neural networks. *ISPRS International Journal of Geo Information* pp. 110–133. Vol. 7(3).

Guo, X., Chen, W. and Yin, J., 2016. A simple approach for unsupervised domain adaptation. In: *IEEE International Conference on Pattern Recognition (ICPR)*, pp. 1566–1570.

Haeusser, P., Frerix, T., Mordvintsev, A. and Cremers, D., 2017. Associative domain adaptation. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2765–2773.

Hara, K., Saito, D. and Shouno, H., 2015. Analysis of function of rectified linear unit used in deep learning. In: *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8.

He, K., Zhang, X., Ren, S. and Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034.

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Hein, M., Andriushchenko, M. and Bitterwolf, J., 2019. Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 41–50.

Ho, T. K., 1995. Random decision forests. In: *IEEE Proceedings of the 3rd International Conference on Document Analysis and Recognition*, pp. 278–282.

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A. and Darrell, T., 2018. Cycada: Cycle-consistent adversarial domain adaptation. In: *International Conference Machine Learning (ICML)*, pp. 1989–1998.

Hoffman, J., Wang, D., Yu, F. and Darrell, T., 2016. FCNs in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv Computing Research Repository (CoRR)*.

Hong, W., Wang, Z., Yang, M. and Yuan, J., 2018. Conditional generative adversarial network for structured domain adaptation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1335–1344.

Hoyer, L., Dai, D. and Gool, L. V., 2022. DAFormer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9924–9935.

Hu, L., Kan, M., Shan, S. and Chen, X., 2018. Duplex generative adversarial network for unsupervised domain adaptation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1498–1507.

Huang, H., Huang, Q. and Krahenbuhl, P., 2018a. Domain transfer through deep activation matching. In: *European Conference on Computer Vision (ECCV)*, pp. 590–605.

Huang, J., Lu, S., Guan, D. and Zhang, X., 2020. Contextual-relation consistent domain adaptation for semantic segmentation. In: *European Conference on Computer Vision (ECCV)*, Springer, pp. 705–722.

Huang, X., Liu, M., Belongie, S. J. and Kautz, J., 2018b. Multimodal unsupervised image-to-image translation. In: *European Conference on Computer Vision (ECCV)*, pp. 179–196.

Iakubovskii, P., 2019. Segmentation models pytorch. `https://github.com/qubvel/segmentation_models.pytorch`.

Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International Conference Machine Learning (ICML)*, Vol. 37, pp. 448–456.

Iqbal, J. and Ali, M., 2020. MLSL: Multi-level self-supervised learning for domain adaptation with spatially independent and semantically consistent labeling. In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1853–1862.

Isola, P., Zhu, J.-Y., Zhou, T. and Efros, A. A., 2017. Image-to-image translation with conditional adversarial networks. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1125–1134.

Ji, S., Wang, D. and Luo, M., 2020. Generative adversarial network-based full- space domain adaptation for land cover classification from multiple-source remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing* 59(5), pp. 1–13.

Kang, G., Wei, Y., Yang, Y., Zhuang, Y. and Hauptmann, A. G., 2020. Pixel-level cycle association: A new perspective for domain adaptive semantic segmentation. In: *Conference on Neural Information Processing Systems (NeurIPS)*, Vol. 33, pp. 3569–3580.

Karras, T., Aila, T., Laine, S. and Lehtinen, J., 2018. Progressive growing of GANs for improved quality, stability, and variation. In: *International Conference on Learning Representations (ICLR)*.

Kingma, D. and Ba, J., 2014. Adam: a method for stochastic optimization. In: *International Conference on Learning Representations (ICLR)*.

Krizhevsky, A., Sutskever, I. and Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In: *Conference on Neural Information Processing Systems (NeurIPS)*, Vol. 25, pp. 1106–1114.

Krogh, A. and Hertz, J., 1991. A simple weight decay can improve generalization. *Advances in Neural Information Processing Systems*. Vol. 4.

Kwak, G.-H. and Park, N.-W., 2022. Unsupervised domain adaptation with adversarial self-training for crop classification using remote sensing images. *Remote Sensing* 14(18), pp. 4639–4663.

LeCun, Y., Haffner, P., Bottou, L. and Bengio, Y., 1999. Object recognition with gradient-based learning. In: *Shape, Contour and Grouping in Computer Vision*, pp. 319–347.

Lee, H.-Y., Tseng, H.-Y., Huang, J.-B., Singh, M. and Yang, M.-H., 2018. Diverse image-to-image translation via disentangled representations. In: *European Conference on Computer Vision (ECCV)*, pp. 35–51.

Lee, K., Lee, H. and Hwang, J. Y., 2021. Self-mutating network for domain adaptive segmentation in aerial images. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 7068–7077.

Li, G., Kang, G., Liu, W., Wei, Y. and Yang, Y., 2020a. Content-consistent matching for domain adaptive semantic segmentation. In: *European Conference on Computer Vision (ECCV)*, Springer, pp. 440–456.

Li, X., Luo, M., Ji, S., Zhang, L. and Lu, M., 2020b. Evaluating generative adversarial networks based image-level domain transfer for multi-source remote sensing image segmentation and object detection. *International Journal of Remote Sensing* 41(19), pp. 7327–7351.

Li, Y., Wang, N., Shi, J., Hou, X. and Liu, J., 2018. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition* 80, pp. 109–117.

Li, Y., Yuan, L. and Vasconcelos, N., 2019. Bidirectional learning for domain adaptation of semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6936–6945.

Lian, Q., Duan, L., Lv, F. and Gong, B., 2019. Constructing self-motivated pyramid curriculums for cross-domain semantic segmentation: A non-adversarial approach. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 6757–6766.

Lin, T.-Y., Goyal, P., Girshick, R., He, K. and Dollár, P., 2017. Focal loss for dense object detection. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988.

Liu, M.-Y., Breuel, T. and Kautz, J., 2017. Unsupervised image-to-image translation networks. In: *Advances in Neural Information Processing Systems*, Vol. 30, pp. 700–708.

Liu, W. and Qin, R., 2020. A multikernel domain adaptation method for unsupervised transfer learning on cross-source and cross-region remote sensing data classification. *IEEE Transactions on Geoscience and Remote Sensing* 58(6), pp. 4279–4289.

Liu, W., Su, F. and Huang, X., 2020. Unsupervised adversarial domain adaptation network for semantic segmentation. *IEEE Geoscience and Remote Sensing Letters* 17(11), pp. 1978–1982.

Long, J., Shelhamer, E. and Darrell, T., 2015a. Fully convolutional networks for semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440.

Long, M., Cao, Y., Wang, J. and Jordan, M., 2015b. Learning transferable features with deep adaptation networks. In: *International Conference Machine Learning (ICML)*, pp. 97–105.

Luo, Y., Liu, P., Guan, T., Yu, J. and Yang, Y., 2019a. Significance-aware information bottleneck for domain adaptive semantic segmentation. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 6777–6786.

Luo, Y., Zheng, L., Guan, T., Yu, J. and Yang, Y., 2019b. Taking a closer look at domain shift: category-level adversaries for semantics consistent domain adaptation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2507–2516.

Maas, A. L., Hannun, A. Y. and Ng, A. Y., 2013. Rectifier nonlinearities improve neural network acoustic models. In: *International Conference Machine Learning (ICML)*, Workshop on Deep Learning for Audio, Speech and Language Processing.

Maggiori, E., Tarabalka, Y., Charpiat, G. and Alliez, P., 2017. Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In: *International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 3226–3229.

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z. and Smolley, S. P., 2017. Least squares generative adversarial networks. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2813–2821.

Marmanis, D., Wegner, J. D., Galliani, S., Schindler, K., Datcu, M. and Stilla, U., 2016. Semantic segmentation of aerial images with an ensemble of CNSS. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Science*, Vol. III-3, pp. 473–480.

Matasci, G., Volpi, M., Kanevski, M. F., Bruzzone, L. and Tuia, D., 2015. Semisupervised transfer component analysis for domain adaptation in remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing* 53(7), pp. 3550–3564.

McCulloch, W. S. and Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics* 5(4), pp. 115–133.

Mei, K., Zhu, C., Jiang, L., Liu, J. and Qiao, Y., 2020. Cross-stained segmentation from renal biopsy images using multi-level adversarial learning. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1424–1428.

Michieli, U., Biasetton, M., Agresti, G. and Zanuttigh, P., 2020. Adversarial learning and self-teaching techniques for domain adaptation in semantic segmentation. *IEEE Transactions on Intelligent Vehicles* 5(3), pp. 508–518.

Miyato, T., Kataoka, T., Koyama, M. and Yoshida, Y., 2018. Spectral normalization for generative adversarial networks. In: *International Conference on Learning Representations (ICLR)*.

Mnih, V., 2013. Machine learning for aerial image labeling. PhD thesis, University of Toronto, Canada.

Morerio, P., Cavazza, J. and Murino, V., 2018. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. In: *International Conference on Learning Representations (ICLR)*.

Murez, Z., Kolouri, S., Kriegman, D., Ramamoorthi, R. and Kim, K., 2018a. Image to image translation for domain adaptation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4500–4509.

Murez, Z., Kolouri, S., Kriegman, D., Ramamoorthi, R. and Kim, K., 2018b. Image to image translation for domain adaptation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4500–4509.

Musto, L. and Zinelli, A., 2020. Semantically adaptive image-to-image translation for domain adaptation of semantic segmentation. *arXiv e-prints*.

Nair, V. and Hinton, G. E., 2010. Rectified linear units improve restricted boltzmann machines. In: *International Conference Machine Learning (ICML)*, pp. 807–814.

Noa, J., Soto, P. J., Costa, G. A. O. P., Wittich, D., Feitosa, R. Q. and Rottensteiner, F., 2021. Adversarial discriminative domain adaptation for deforestation detection. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Science* V-3-2021, pp. 151–158.

Palladino, J. A., Slezak, D. F. and Ferrante, E., 2020. Unsupervised domain adaptation via CycleGAN for white matter hyperintensity segmentation in multicenter MR images. In: *International Symposium on Medical Information Processing and Analysis*, pp. 1158302–1158313. Vol. 11583.

Pan, F., Shin, I., Rameau, F., Lee, S. and Kweon, I. S., 2020. Unsupervised intra-domain adaptation for semantic segmentation through self-supervision. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3763–3772.

Pan, S. J. and Yang, Q., 2009. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10), pp. 1345–1359.

Pandey, P., Tyagi, A. K., Ambekar, S. and Prathosh, A. P., 2020. Unsupervised domain adaptation for semantic segmentation of NIR images through generative latent search. In: *European Conference on Computer Vision (ECCV)*, Springer, pp. 413–429.

Peng, D., Guan, H., Zang, Y. and Bruzzone, L., 2022. Full-level domain adaptation for building extraction in very-high-resolution optical remote-sensing images. *IEEE Transactions on Geoscience and Remote Sensing* pp. 1–17. Vol. 60.

Perona, P., 1995. Deformable kernels for early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(5), pp. 488–499.

Pizzati, F., de Charette, R., Zaccaria, M. and Cerri, P., 2020. Domain bridge for unpaired image-to-image translation and unsupervised domain adaptation. In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 2979–2987.

Prechelt, L., 1998. Early stopping-but when? In: *Neural Networks: Tricks of the trade*, Springer Berlin Heidelberg, pp. 55–69.

Riz, E., Demir, B. and Bruzzone, L., 2016. Domain adaptation based on deep denoising auto-encoders for classification of remote sensing images. In: *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 197–204. Vol. 10004.

Ronneberger, O., Fischer, P. and Brox, T., 2015. U-Net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, Vol. 9351, pp. 234–241.

Rosenblatt, F., 1962. *Perceptions and the theory of brain mechanisms.* Spartan Books, Washington, D. C., USA.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J., 1986. Learning representations by back-propagating errors. *Nature* pp. 533–536. Vol. 323.

Sakaridis, C., Dai, D. and Gool, L. V., 2019. Guided curriculum model adaptation and uncertainty-aware evaluation for semantic nighttime image segmentation. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 7373–7382.

Santurkar, S., Tsipras, D., Ilyas, A. and Madry, A., 2018. How does batch normalization help optimization? In: *Conference on Neural Information Processing Systems (NeurIPS)*, Vol. 31, pp. 2488–2498.

Shan, Y., Chew, C. M. and Lu, W. F., 2020. Semantic-aware short path adversarial training for cross-domain semantic segmentation. *Neurocomputing* pp. 125–132. Vol. 380.

Shorten, C. and Khoshgoftaar, T. M., 2019. A survey on image data augmentation for deep learning. *Journal of Big Data* 6(54), pp. 60–108.

Soto, P., Costa, G., Feitosa, R., Happ, P., Ortega, M., Noa, J., Almeida, C. and Heipke, C., 2020. Domain adaptation with cyclegan for change detection in the amazon forest. In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Science*, Vol. XLIII-B3, pp. 1635–1643.

Soto, P. J., da Costa, G. A. O. P., Feitosa, R. Q., Ortega Adarme, M. X., de Almeida, C. A., Heipke, C. and Rottensteiner, F., 2021. An unsupervised domain adaptation approach for change detection and its application to deforestation mapping in tropical biomes. *ISPRS Journal of Photogrammetry and Remote Sensing* pp. 113–128. Vol. 181.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1), pp. 1929–1958.

Subhani, M. N. and Ali, M., 2020. Learning from scale-invariant examples for domain adaptation in semantic segmentation. In: *European Conference on Computer Vision (ECCV)*, pp. 290–306.

Sun, B. and Saenko, K., 2016. Deep CORAL: Correlation alignment for deep domain adaptation. In: *European Conference on Computer Vision (ECCV)*, pp. 443–450.

Sun, B., Feng, J. and Saenko, K., 2016. Return of frustratingly easy domain adaptation. In: *AAAI Conference on Artificial Intelligence*, Vol. 30(1), pp. 2058–2065.

Tanwani, A. K., 2020. DIRL: Domain-invariant representation learning for sim-to-real transfer. In: *Conference on Robot Learning (CoRL)*, Vol. 155, pp. 1558–1571.

Tasar, O., Happy, S., Tarabalka, Y. and Alliez, P., 2020a. Colormapgan: Unsupervised domain adaptation for semantic segmentation using color mapping generative adversarial networks. *IEEE Transactions on Geoscience and Remote Sensing* 58(10), pp. 7178–7193.

Tasar, O., Happy, S., Tarabalka, Y. and Alliez, P., 2020b. Semi2i: Semantically consistent image-to-image translation for domain adaptation of remote sensing data. In: *International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 1837–1840.

Tong, X.-Y., Xia, G.-S., Lu, Q., Shen, H., Li, S., You, S. and Zhang, L., 2020. Land-cover classification with high-resolution remote sensing images using transferable deep models. *Remote Sensing of Environment*. Vol. 237, Paper nr. 111322.

Tsai, Y., Hung, W., Schulter, S., Sohn, K., Yang, M. and Chandraker, M., 2018. Learning to adapt structured output space for semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7472–7481.

Tsai, Y., Sohn, K., Schulter, S. and Chandraker, M., 2019. Domain adaptation for structured output via discriminative patch representations. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1456–1465.

Tuia, D., Persello, C. and Bruzzone, L., 2016. Domain adaptation for the classification of remote sensing data: An overview of recent advances. *IEEE Geoscience and Remote Sensing Magazine* 4(2), pp. 41–57.

Tzeng, E., Hoffman, J., Saenko, K. and Darrell, T., 2017. Adversarial discriminative domain adaptation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2962–2971.

Ulyanov, D., Lebedev, V., Vedaldi, A. and Lempitsky, V. S., 2016. Texture networks: Feed-forward synthesis of textures and stylized images. In: *International Conference Machine Learning (ICML)*, JMLR Workshop and Conference Proceedings, Vol. 48, pp. 1349–1357.

Varga, D. and Szirányi, T., 2016. Fully automatic image colorization based on convolutional neural network. In: *IEEE International Conference on Pattern Recognition (ICPR)*, pp. 3691–3696.

Vu, T.-H., Jain, H., Bucher, M., Cord, M. and Perez, P., 2019. ADVENT: Adversarial entropy minimization for domain adaptation in semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2512–2521.

Wang, J., Ma, A., Zhong, Y., Zheng, Z. and Zhang, L., 2022. Cross-sensor domain adaptation for high spatial resolution urbanland-cover mapping: From airborne to spaceborne imagery. *Remote Sensing of Environment*. Vol. 277, Paper nr. 113058.

Wang, M. and Deng, W., 2018. Deep visual domain adaptation: A survey. *Neurocomputing* pp. 135–153.

Wang, Z., Jing, B., Ni, Y., Dong, N., Xie, P. and Xing, E. P., 2020a. Adversarial domain adaptation being aware of class relationships. In: *European Converence on Artificial Intelligence*, Vol. 325, pp. 1579–1586.

Wang, Z., Yu, M., Wei, Y., Feris, R., Xiong, J., Hwu, W., Huang, T. S. and Shi, H., 2020b. Differential treatment for stuff and things: A simple unsupervised domain adaptation method for semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12632–12641.

Wegner, J.-D., Rottensteiner, F., Sohn, G. and Gerke, M., 2017. The ISPRS 2d semantic labelling contest. www2.isprs.org/commissions/comm2/wg4/benchmark/semantic-labeling. (accessed 22/12/2020).

Wei, G., Lan, C., Zeng, W. and Chen, Z., 2021. MetaAlign: Coordinating domain alignment and classification for unsupervised domain adaptation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16643–16653.

Wiejak, J., Buxton, H. and Buxton, B. F., 1985. Convolution with separable masks for early image processing. *Computer Vision, Graphics, and Image Processing* 32(3), pp. 279–290.

Wittich, D., 2020. Deep domain adaptation by weighted entropy minimization for the classification of aerial images. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Science*, Vol. V-2, pp. 591–598.

Wittich, D. and Rottensteiner, F., 2019. Adversarial domain adaptation for the classification of aerial images and height data using convolutional neural natworks. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Science*, Vol. IV-2/W7, pp. 197–204.

Wittich, D. and Rottensteiner, F., 2021. Appearance based deep domain adaptation for the classification of aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing* pp. 82–102. Vol. 180.

Wittich, D., Rottensteiner, F., Voelsen, M., Heipke, C. and Müller, S., 2022. Deep learning for the detection of early signs for forest damage based on satellite imagery. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Science*, Vol. V-2-2022, pp. 307–315.

Xu, M., Wu, M., Chen, K., Zhang, C. and Guo, J., 2022. The eyes of the gods: A survey of unsupervised domain adaptation methods based on remote sensing data. *Remote Sensing* 14(17), pp. 4380–4413.

Yang, C., Rottensteiner, F. and Heipke, C., 2019. Towards better classification of land cover and land use based on convolutional neural networks. In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Science*, Vol. XLII-2/W13, pp. 139–146.

Yang, J., An, W., Wang, S., Zhu, X., Yan, C. and Huang, J., 2020a. Label-driven reconstruction for domain adaptation in semantic segmentation. In: *European Conference on Computer Vision (ECCV)*, pp. 480–498.

Yang, J., Xu, R., Li, R., Qi, X., Shen, X., Li, G. and Lin, L., 2020b. An adversarial perturbation oriented domain adaptation approach for semantic segmentation. In: *AAAI Conference on Artificial Intelligence*, pp. 12613–12620.

Yang, Y. and Soatto, S., 2020. FDA: Fourier domain adaptation for semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4085–4095.

Yang, Y., Lao, D., Sundaramoorthi, G. and Soatto, S., 2020c. Phase consistent ecological domain adaptation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9011–9020.

Yosinski, J., Clune, J., Bengio, Y. and Lipson, H., 2014. How transferable are features in deep neural networks? In: *Conference on Neural Information Processing Systems (NeurIPS)*, Vol. 27, pp. 3320–3328.

Zhang, G., Lei, T., Cui, Y. and Jiang, P., 2019a. A dual-path and lightweight convolutional neural network for high-resolution aerial image segmentation. *ISPRS International Journal of Geo-Information* 8(12), pp. 582–603.

Zhang, P., Zhang, B., Zhang, T., Chen, D., Wang, Y. and Wen, F., 2021a. Prototypical pseudo label denoising and target structure learning for domain adaptive semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12414–12424.

Zhang, Q., Zhang, J., Liu, W. and Tao, D., 2019b. Category anchor-guided unsupervised domain adaptation for semantic segmentation. In: *Conference on Neural Information Processing Systems (NeurIPS)*, Vol. 32, pp. 433–443.

Zhang, Y., David, P. and Gong, B., 2017. Curriculum domain adaptation for semantic segmentation of urban scenes. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2020–2030.

Zhang, Y., Liu, T., Long, M. and Jordan, M. I., 2019c. Bridging theory and algorithm for domain adaptation. In: *International Conference Machine Learning (ICML)*, Vol. 97, pp. 7404–7413.

Zhang, Y., Qiu, Z., Yao, T., Liu, D. and Mei, T., 2018a. Fully convolutional adaptation networks for semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6810–6818.

Zhang, Y., Wang, N., Cai, S. and Song, L., 2018b. Unsupervised domain adaptation by mapped correlation alignment. *IEEE Access* 6, pp. 44698–44706.

Zhang, Z., Doi, K., Iwasaki, A. and Xu, G., 2021b. Unsupervised domain adaptation of high-resolution aerial images via correlation alignment and self training. *IEEE Geoscience and Remote Sensing Letters* 18(4), pp. 746–750.

Zhang, Z., Liu, Q. and Wang, Y., 2018c. Road extraction by deep residual U-Net. *IEEE Geoscience and Remote Sensing Letters* 15(5), pp. 749–753.

Zhao, H., des Combes, R. T., Zhang, K. and Gordon, G. J., 2019. On learning invariant representations for domain adaptation. In: *International Conference Machine Learning (ICML)*, Vol. 97, pp. 7523–7532.

Zhao, Y., Guo, P., Gao, H. and Chen, X., 2023. Depth-assisted ResiDualGan for cross-domain aerial images semantic segmentation. *IEEE Geoscience and Remote Sensing Letters*. Vol. 20, Paper nr. 2500305.

Zhu, J.-Y., Park, T., Isola, P. and Efros, A. A., 2017a. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2223–2232.

Zhu, X. X., Tuia, D., Mou, L., Xia, G.-S., Zhang, L., Xu, F. and Fraundorfer, F., 2017b. Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine* 5(4), pp. 8–36.

Zou, Y., Yu, Z., Kumar, B. and Wang, J., 2018. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In: *European Conference on Computer Vision (ECCV)*, pp. 289–305.

Appendix

A: Additional Examples of Image Adaptation

Evaluation of adaptation for the scenarios $N \rightarrow P$ (cf. Figure 7.1) and $V \rightarrow B$ (cf. Figure 7.2).

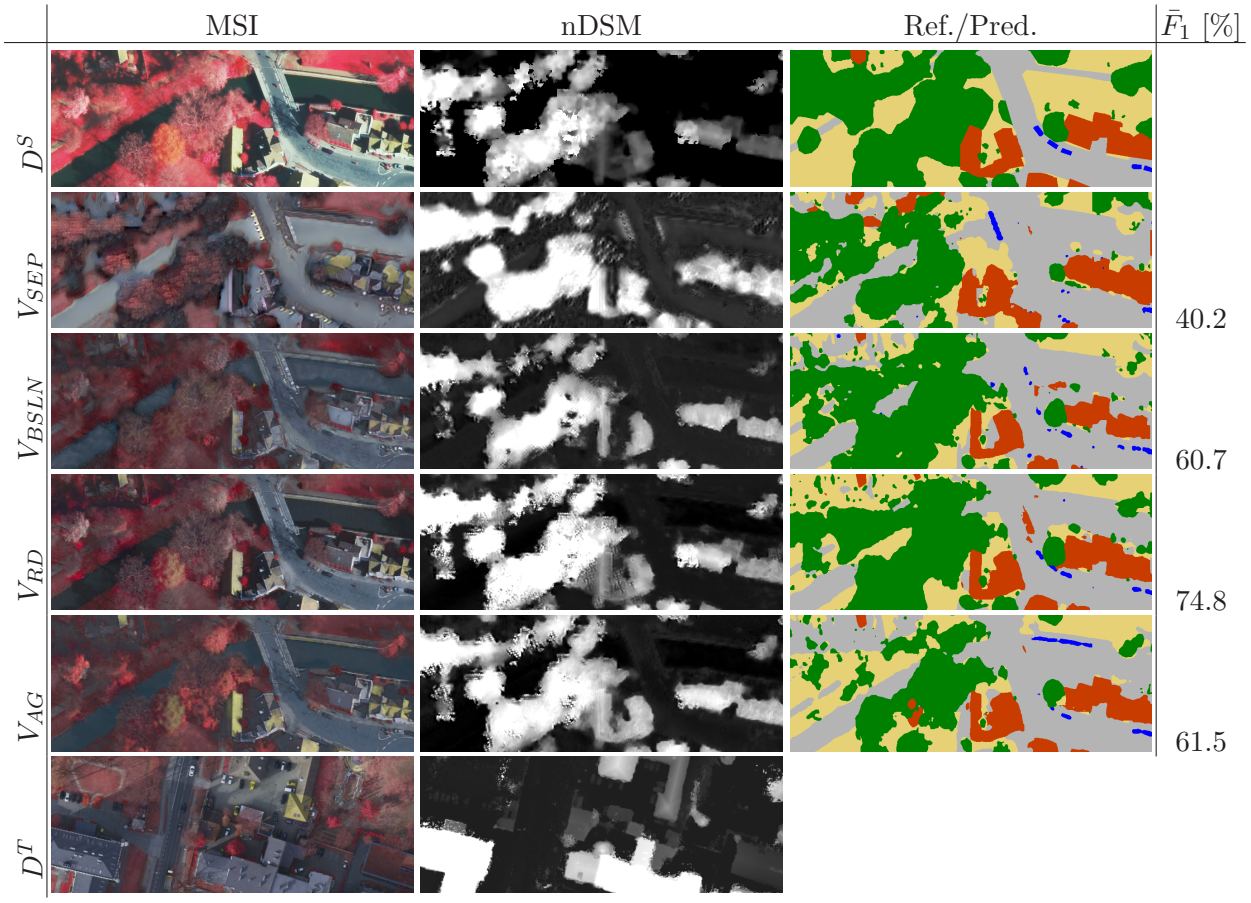

Figure 7.1: Examples for appearance adaptation in scenario $N \rightarrow P_{20}$. The structure of the figure follows the one of Figure 6.1.

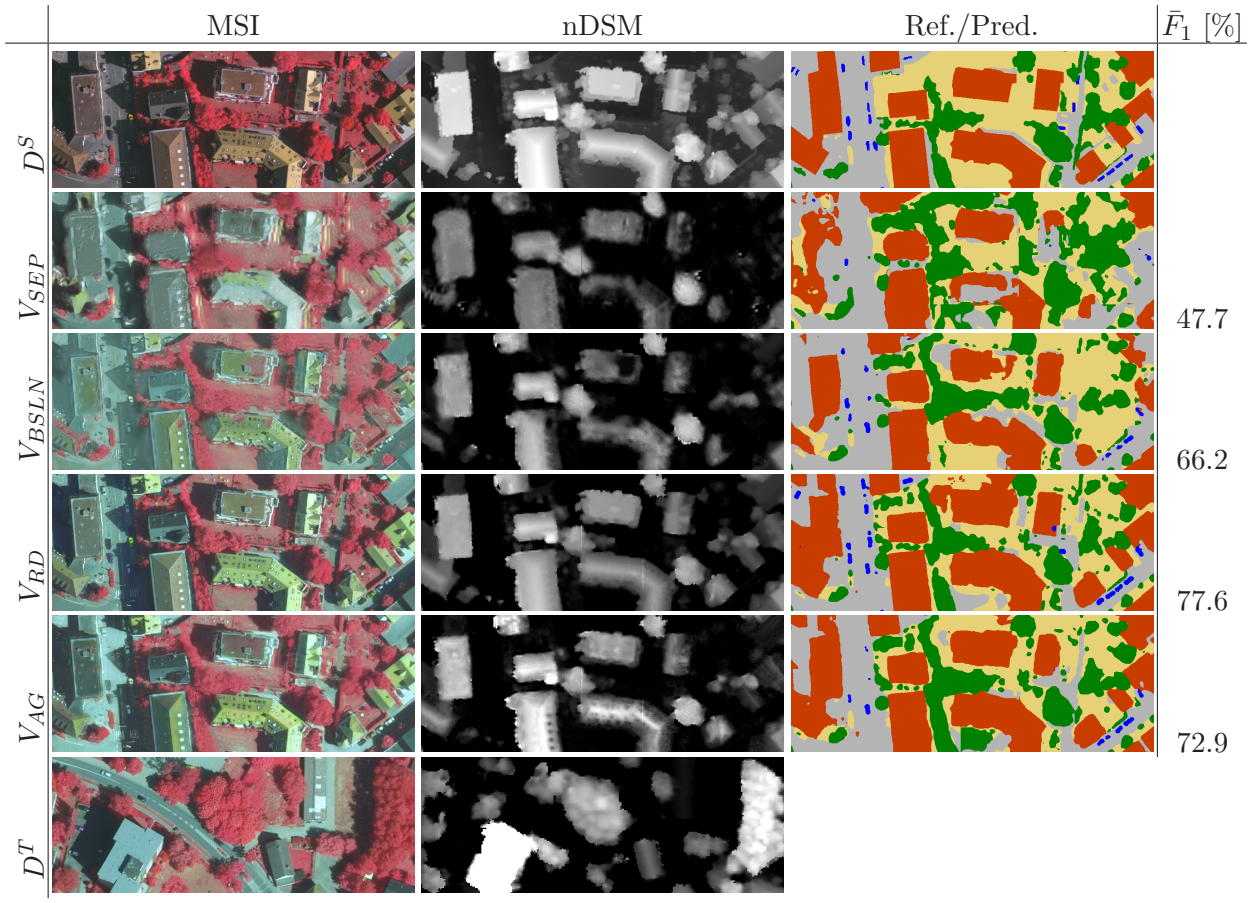

Figure 7.2: Examples for appearance adaptation in scenario $V_{20} \rightarrow B$. The structure of the figure follows the one of Figure 6.1.

B: Numerical Results for Detailed Evaluation

Table 7.1 provides the class-wise quality metrics corresponding to Figure 6.7.

Class	Strategy	$B \to H$	$H \to N$	$N \to P_{20}$	$P_{20} \to V_{20}$	$V_{20} \to B$
SG	N.T.	57.6±1.9	78.1±1.4	85.0±0.5	83.7±0.4	78.7±1.2
	V_{RD}	70.4±3.0	80.7±1.4	86.6±0.4	85.1±0.2	81.4±0.5
	$V_{RD,ABN}$	71.3±3.0	77.7±2.4	87.1±1.7	83.3±0.2	80.9±0.5
	S.T.	86.5±0.1	88.5±0.1	92.0±0.1	88.8±0.1	86.7±0.4
BU	N.T.	89.3±0.3	84.5±1.3	93.2±0.8	92.1±0.5	87.1±1.6
	V_{RD}	91.1±0.5	87.0±0.9	94.8±0.6	93.1±0.2	87.4±0.7
	$V_{RD,ABN}$	90.0±0.2	89.0±0.4	95.0±0.7	93.2±0.1	88.2±0.7
	S.T.	94.1±0.1	93.5±0.1	96.9±0.1	93.4±0.1	94.5±0.1
LV	N.T.	33.4±0.9	81.4±0.8	73.7±0.5	65.1±0.5	81.0±0.5
	V_{RD}	36.9±6.1	81.5±1.0	73.8±0.4	69.3±0.8	81.4±0.4
	$V_{RD,ABN}$	40.8±2.8	63.2±3.0	73.8±2.5	69.0±0.5	80.4±0.2
	S.T.	57.3±0.4	89.8±0.2	83.8±0.2	78.8±0.4	87.8±0.2
HV	N.T.	70.8±4.2	72.7±0.6	59.4±3.3	80.0±0.2	81.4±0.8
	V_{RD}	76.6±1.7	73.4±0.5	61.6±1.6	81.3±0.4	83.1±0.1
	$V_{RD,ABN}$	80.0±0.5	62.3±1.9	64.1±1.4	80.9±0.2	82.5±0.3
	S.T.	83.5±0.1	81.5±0.1	83.4±0.1	85.7±0.3	87.1±0.1
CA	N.T.	56.4±2.3	69.1±1.7	78.9±1.2	65.0±0.9	72.4±0.9
	V_{RD}	67.1±1.9	73.4±0.7	81.6±1.6	65.6±3.1	74.3±0.4
	$V_{RD,ABN}$	69.7±1.8	73.8±0.6	77.7±7.8	67.6±1.3	71.9±0.6
	S.T.	80.3±0.6	74.0±0.5	89.4±0.3	74.8±0.7	79.5±0.7
Avg.	N.T.	61.5±1.2	77.1±0.9	78.0±0.5	77.2±0.4	80.1±0.5
	V_{RD}	68.4±0.6	79.2±0.6	79.7±0.6	78.9±0.4	81.5±0.3
	$V_{RD,ABN}$	70.4±1.5	73.2±0.9	79.6±2.7	78.8±0.2	80.8±0.2
	T.T.	80.3±0.2	85.5±0.2	89.1±0.1	84.3±0.2	87.1±0.2

Table 7.1: Class wise and average F_1 scores in % for different adaptation scenarios. Each cell gives the mean and standard deviation computed over five repetitions of the experiments. N.T.: naïve transfer (classifier was trained in D^S). V_{RD}, $V_{RD,ABN}$: Variants of the proposed method (classifier trained in D^S and adapted to D^T). T.T.: Target training (classifier was trained in D^T). Class abbreviations as in Table 5.1.