DGK Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 879

Bashir Kazimi

Self Supervised Learning for Detection of Archaeological Monuments in LiDAR Data

München 2021

Bayerische Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5291-8

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 379, Hannover 2021

DGK Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 879

Self Supervised Learning for Detection of Archaeological Monuments in LiDAR Data

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

von

M.Sc. Bashir Kazimi

Geboren am 04.07.1991 in Ghazni

München 2021

Bayerische Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5291-8

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 379, Hannover 2021

Adresse der DGK:

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften (DGK)

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

Telefon +49 – 89 – 23 031 1113 ● Telefax +49 – 89 – 23 031 - 1283 / - 1100

e-mail post@dgk.badw.de ● http://www.dgk.badw.de

Prüfungskommission:

Vorsitzender: Prof. Dr.-Ing. Ingo Neumann

Referent: Prof. Dr.-Ing. habil. Monika Sester

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

Prof. Dr. Kourosh Khoshelham

Tag der mündlichen Prüfung: 07.07.2021

ISSN 0065-5325 ISBN 978-3-7696-5291-8

Abstract

Detecting and localizing archaeological monuments and historical man-made terrain structures is essential for learning and preserving our cultural heritage. With the advancement of laser scanning technology, it is possible to acquire Airborne Laser Scanning (ALS) point clouds and create Digital Terrain Models (DTMs), which can be analyzed by archaeologists for interesting monuments and structures. However, manually inspecting high volumes of DTM data is a time-consuming task. The goal of this research is to utilize deep learning for automated detection of archaeological monuments and historical man-made terrain structures in DTMs. Southern Lower Saxony, i.e. specifically the Harz mining region, was chosen as the study region because a significant number of monuments can be found here. Due to the limited amounts of annotated data and the large amounts of unlabeled data, the focus is on Self Supervised Learning (SSL).

SSL involves two steps: pretext and downstream. In the pretext, a model is trained on unlabeled data to learn intrinsic characteristics and interesting patterns in the input. Downstream is the second step, which involves learning patterns from annotated datasets. In the downstream step, the trained model from the pretext step is either used a fixed feature extractor or directly finetuned for supervised tasks on annotated datasets.

In this research, convolutional encoder-decoder networks and Generative Adversarial Networks (GANs) are trained on unlabeled DTM data in the SSL pretext. The trained models are then customized for downstream tasks such as classification, instance segmentation, and semantic segmentation. They are then finetuned on small amounts of annotated data for detection of archaeological monuments and man-made terrain structures in the Harz region in Lower Saxony.

Experiments are conducted on three different datasets from the Harz region. The first dataset contains areal structures which includes archaeological monuments such as charcoal kilns, burial mounds and mining holes and other man-made terrain structures such as bomb craters. The second dataset contains linearly elongated structures which includes archaeological monuments such as ditches and hollow ways and other man-made structures such as paths and roads. The third dataset from Harz includes annotated examples of historical stone quarries. Results of the experiments indicate the positive impact of SSL pretraining on the downstream tasks. The best classification algorithm performs similar with and without SSL pretraining. However, for instance and semantic segmentation tasks which are much more complex, SSL pretraining improves the Mean Average Precision (MAP) score by 5.28 % and the Mean Intersection Over Union (MIOU) score by 4.72 %, respectively, on the Harz areal dataset. On the linear structures dataset, the increase in MAP and MIOU scores are 6.18 % and 1.22 %, respectively. Finally, SSL pretraining leads to an increase of 3.02 % in the MIOU score in the stone quarries dataset.

Keywords: Self Supervised Learning, LiDAR, Archaeology, Historical Mining

Kurzfassung

Das Detektieren und Lokalisieren von archäologischen Denkmälern und historischen, von Menschenhand geschaffenen Geländestrukturen ist für den Schutz unseres kulturellen Erbes unerlässlich. Mit dem Fortschritt der Laserscanning-Technologie ist es möglich, Airborne Laser Scanning (ALS) Punktwolken zu erfassen und Digitale Geländemodelle (DGM) zu erstellen, die von Archäologen auf interessante Denkmäler und Strukturen analysiert werden können. Allerdings ist die manuelle Detektion im digitalen Geländemodell (DGM) eine zeitaufwändige Aufgabe. Das Ziel der Arbeit ist es, Deep Learning für die automatisierte Erkennung von archäologischen Denkmälern und historischen, von Menschenhand geschaffenen Geländestrukturen in DGMs einzusetzen. Als Modellregion wurde Südniedersachsen, d.h. speziell die Montanregion Harz gewählt, weil sich hier eine besondere Dichte von Denkmälern befindet. Aufgrund der begrenzten Menge an gelabelten Daten und der großen Menge an ungelabelten Daten liegt der Fokus auf Self Surpervised Learning (SSL).

SSL umfasst zwei Schritte: Pretext und Downstream. Im Pretext wird ein Modell auf ungelabelten Daten trainiert, um intrinsische Eigenschaften und interessante Muster in der Eingabe zu lernen. Downstream ist der zweite Schritt, der das Lernen von Mustern aus gelabelten Datensätzen beinhaltet. Im Downstream wird das trainierte Modell aus dem Pretext-Schritt entweder als fester Feature-Extraktor verwendet oder direkt für überwachte Aufgaben auf gelabelten Datensätzen nachtrainiert.

In dieser Arbeit werden Convolutional-Encoder-Decoder-Netzwerke und Generative Adversarial Networks (GANs) auf ungelabelten DGM-Daten im SSL-Pretext trainiert. Die trainierten Modelle werden dann im SSL-Downstream für Klassifizierung, Instanzsegmentierung und semantische Segmentierung angepasst. Anschließend werden sie mit Hilfe kleiner Mengen gelabelter Daten für die Erkennung archäologischer Denkmäler in der Harzregion in Niedersachsen nachtrainiert.

Die Experimente werden mit drei verschiedenen Datensätzen aus der Harzregion durchgeführt. Der erste Datensatz enthält kompakte, geschlossene Strukturen, zu denen archäologische Denkmäler wie Meilerplätze, Grabhügel und Pingen und andere vom Menschen geschaffene Geländestrukturen wie Bombentrichter gehören. Der zweite Datensatz enthält langgestreckte lineare Strukturen, zu denen archäologische Denkmäler wie Gräben und Hohlwege und andere vom Menschen geschaffene Strukturen wie Wege und Straßen gehören. Der dritte Datensatz aus dem Harz umfasst annotierte Beispiele von historischen Steinbrüchen. Die Ergebnisse der Experimente zeigen den positiven Einfluss des SSL-Pretrainings auf die Downstream Aufgaben. Der beste Klassifikationsalgorithmus schneidet mit und ohne SSL-Vortraining ähnlich ab. Für Instanz- und semantische Segmentierungsaufgaben, die viel komplexer sind, verbessert das SSL-Pretrainings jedoch den Mean Average Precision (MAP)-Score um 5,28 % und den Mean Intersection Over Union (MIOU)-Score um 4,72 %, jeweils auf dem Harzer Areal-Datensatz. Beim Datensatz für lineare Strukturen beträgt die Steigerung der MAP- und MIOU-Scores 6,18 % bzw. 1,22 %. Schließlich führt das SSL-Pretraining zu einer Erhöhung des MIOU-Scores im Datensatz Steinbrüche um 3,02 %.

Schlagworte: Selbstüberwachtes Lernen, LiDAR, Archäologie, Historischer Bergbau

Contents

1. Introduction

This chapter introduces the the motivation and goal of this research. It gives details of the region under study and lists an overview of how this thesis is structured.

1.1. Motivation and Research Goal

Archaeology studies physical remains of objects to discover facts about human history and culture from prehistoric and historical era. It helps us have a glance at the lives of people in the past and how things have changed through time. While facts and information about the historical era are found in written records, prehistoric era for which no written records are available can only be discovered through study and analysis of archaeological remains. Places with physical remains of human activities in the past are called archaeological sites. Settlements, cemeteries, and historical mining regions are examples of archaeological sites.

Archaeological sites, features and artifacts, and historical terrain structures are part of our cultural heritage. They are indicators of past human accomplishments and sometimes the only sources of information about the history. Therefore, it is important to preserve and protect them. The first step in this direction is to identify and document archaeological sites. Archaeologists discover archaeological features and artifacts by surveying and visual observation of the surface of the earth. Anything that can be used to understand human life in the past is marked as interesting for archaeology and analyzed later.

To speed up the process, drones, air balloons, aerial photography and satellite imagery are used to remotely scan potential archaeological sites and find features and artifacts. However, there are features and artifacts underground or historical terrain structures that are not directly visible by the naked eye. Moreover, archaeological sites could be covered by forest essentially rendering them undetectable in aerial images. In such cases, remote sensing technologies such as Light Detection And Ranging (LiDAR) or Airborne Laser Scanning (ALS) can be used to scan such regions and collect point cloud data by measuring the range and reflectance of the earth's surface and objects on it. The collected point cloud is processed to create DTM data which can be used in analysis and visualization of terrain structures. Archaeologists use Geographic Information System (GIS) software such as ArcGIS (ArcGIS) or Quantum GIS (QGIS) to perform analysis and create visualizations of the DTMs for identifying and registering archaeological monuments.

Even though LiDAR technology is efficient in collecting ALS point clouds for big regions, manual analysis of the DTM by-products is still laborious and time-consuming. Archaeologists need to search and mark potential features and artifacts and then perform a field survey on the marked regions for validation. Moreover, the process has to be repeated every time data is collected for a new region. To automate the process and avoid manual inspection of DTM data for the same set of artifacts and features every time a new region is scanned, Machine Learning (ML) techniques are applied. Archaeologists determine unique characteristics of interesting archaeological monuments and preprocess the data to create annotated datasets. Such a dataset consists of hand-engineered input features paired with their corresponding labels. ML models such as Support Vector Machine (SVM), decision trees, logistic regression, Naive Bayes classifiers, K-nearest neighbours, and more, are trained using the annotated datasets to detect archaeological structures in DTM data given the

hand-engineered features as input. Trained ML models are then used for new datasets every time a region is scanned. Thus, archaeologists do not need to manually inspect DTM data every time.

Classical ML techniques need their input features to be hand-engineered by domain experts. Additionally, many of them only work for linearly separable input data and cannot learn non-linearities or complex relationships between input data and target labels. With the immense increase in availability of data in many research areas and the emergence of powerful computing tools such as Graphics Processing Units (GPUs), Deep Learning (DL) techniques are utilized for automating many tasks such as objection detection in natural images, machine translation, speech recognition, medical image analysis and many more in other research fields. High volumes of LiDAR point clouds and already interpreted datasets in archaeolgocial database systems encourage utilization of DL techniques in archaeology as well. DL models handle input data directly without hand-engineered features, solve complex tasks, and can be trained with large datasets efficiently taking advantage of recent advances in computing. To train such models, a dataset of input DTMs and corresponding target labels are required without the need for hand-engineered input features by domain experts.

Supervised DL models require large annotated datasets for training in order to learn, generalize and perform well on unseen data. In many other domains such as recognition and localization of objects in natural images, machine translation and speech recognition where deep learning research has matured, tons of annotated datasets are available. Examples are the Citscapes dataset for urban scene understanding (Cordts et al., 2016), ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset (Russakovsky et al., 2015), Pascal Visual Object Classes (VOC) dataset (Everingham et al., 2015) in Computer Vision (CV). In Natural Language Processing (NLP) and speech recognition, there are datasets such as Europarl (Koehn, 2005), blog authorship corpus (Schler et al., 2006), and speech commands (Warden, 2018). There are annotated datasets available in the remote sensing community as well including SpaceNet (Van Etten et al., 2018), Deepsat (Basu et al., 2015), Brazilian Coffee Scenes (Penatti et al., 2015), and 2D semantic labeling datasets from International Society for Photogrammetry and Remote Sensing (ISPRS). However, there are no annotated LiDAR datasets publicly available for archaeological research. DL researchers in archaeology create their own datasets and train models for detection of archaeological monuments (Verschoof-van der Vaart and Lambers, 2019; Trier et al., 2019; Gallwey et al., 2019; Soroush et al., 2020; Kazimi et al., 2019b). These datasets are small compared to those in other domains and are not made publicly available. Moreover, manual annotation of large DTMs is quite laborious and time-consuming. Currently, researchers in archaeology working with LiDAR data tackle the problem of insufficient training data by the use of transfer learning from other domains. For example, pretrained deep learning models on ImageNet and other datasets are used to finetune models on LiDAR data acquired for archaeological purposes (Trier et al., 2018, 2019; Verschoof-van der Vaart and Lambers, 2019). While this aids in learning to detect archaeological structures, pretraining deep learning models on LiDAR data directly can lead to better detection and faster convergence. This is the main contribution and aim of this research, i.e., to pretrain deep learning models on LiDAR data and finetune them on the same for archaeological applications. Even though large annotated datasets are not available, unlabeled LiDAR datasets and its derived products, e.g., DTMs, can be exploited by an exciting DL field called Self Supervised Learning (SSL).

The goal of this research is to exploit large volumes of unlabeled DTMs created from LiDAR point clouds for self supervised detection of archaeological monuments. This is done using SSL in two steps: pretext and downstream. In the pretext phase, unlabeled DTMs are used to train Deep Neural Networks (DNNs) for learning hidden features and capturing intrinsic characteristics of the input DTM. In the second step, i.e., the downstream phase, small datasets annotated with objects

and structures related to historical mining are used to finetune the trained DL models from the pretext in order to learn detecting such structures.

DNNs in the first step in SSL, i.e., pretext, utilize unlabeled data for learning and extracting useful features and hidden representations. However, for the models to learn this, some implicit labels or supervision signals are required. Implicit labels can be the input data itself or any type of data that can automatically be created from the available data with ease and without manual annotations. Examples include rotating input images by 0, 90, 180 and 270 degrees and training a model to predict the applied rotation (Gidaris et al., 2018), or shuffling image patches and training a model that learns the shuffling permutations (Noroozi and Favaro, 2016). Further examples of implicit labeling or self supervised representation learning are studied by Zhang et al. (2019b), Doersch et al. (2015), Zhang et al. (2016c), Larsson et al. (2016), Vondrick et al. (2018), Lee et al. (2017), and Misra et al. (2016), among others.

As implicit labels in representation learning of DTMs, an array of derived rasters originally intended for better visualization of the terrain can be utilized. In principle, the DTM patches alone can be used for this purpose by applying rotations, or cropping parts of the DTM patches, and training a model to learn the applied rotations or reconstruct the cropped regions. However, relief visualization rasters derived from the DTMs are shown to be effective, compared to the original DTMs, in training deep learning models for supervised tasks (Kazimi et al., 2019a). Therefore, such rasters are used as implicit supervision signals in this research. Rasters such as Simple Local Relief Model (SLRM), Local Dominance (LD), Positive Openness (POS), Negative Openness (NEG), Sky View Factor (SVF) and slope (explained in detail in Chapter 2) are derived from DTMs, each of which is suitable for visualization of certain structures in the terrain that are not directly visible by the naked eye in the DTMs. Many GIS tools have functions or toolboxes that calculate raster derivatives given an input DTM. In this research, the so called Relief Visualization Toolbox (RVT) software (Kokalj and Somrak, 2019) is used to calculate 6 previously mentioned raster derivatives from unlabeled DTM data. Such rasters are used to learn hidden representations in DTMs by training DL models that take DTMs as inputs and learn to generate previously mentioned raster derivatives for them. Thus, the first hypothesis to be investigated in this research is:

Research Hypothesis 1 *Relief visualization rasters are useful implicit supervision signals for training deep learning models for learning hidden representations and properties in Digital Terrain Models (DTMs).*

Deep learning models in the second step in SSL, i.e., downstream, aim to utilize annotated datasets to solve a supervised task, e.g., image classification, bounding box detection, or semantic and instance segmentation. When large datasets with labels are available, such models perform well. However, when the dataset is small and there is a possibility to get huge amounts of unlabeled data for the same domain, pretrained models from the first step, i.e., pretext, prove useful. The trained pretext models are utilized to extract useful representations from the inputs in annotated datasets. They are finetuned for supervised downstream tasks. It is proved that utilizing pretrained models and finetuning them leads to improved performance compared to training them directly from scratch. Using SSL pretraining, the model already learns important properties of the input data which helps gain improved performance in the supervised downstream task. This leads to the second hypothesis investigated in this thesis:

Research Hypothesis 2 *Self supervised pretraining and finetuning the pretrained models for supervised learning tasks with annotated DTM datasets leads to improved performance compared to training supervised models with random weight initialization.*

The downstream task in this research is the automated detection of archaeological monuments and historical man-made terrain structures in DTM data. Three distinct datasets are created for the

experiments in this research. The distinction is based on the shapes and the sizes of structures in each dataset. The categories studied in the first dataset include archaeological monuments such as charcoal kilns, burial mounds and mining holes and other man-made terrain structures like bomb craters. They have closed, compact and areal shapes and are more or less similar in size. The second dataset contains linearly elongated structures which include archaeological monuments such as ditches and hollow ways, and man-made structures such as paths and roads. The third dataset consists of annotated examples of historical stone quarries. Even though stone quarries also have closed and areal shapes, they are relatively big in size and hence they are not included in the first dataset. Models pretrained in the pretext step for representation learning from unlabeled DTMs are finetuned in the downstream step with the small annotated datasets to detect such structures. They are customized to perform various supervised tasks such as classification, instance segmentation and semantic segmentation.

In classification, the model is customized and finetuned to produce a single label for every DTM input image showing the existence (or absence) of previously mentioned structures. While this approach is useful in cases where a coarse indication of existence for objects of interest is adequate, it is not suitable for fine-grained identification of objects with their precise locations. For exact delineation of each object in the DTM input, instance segmentation is more appropriate in which the model is trained to learn bounding box coordinates, segmentation masks and class labels for each object instance in the given input. This gives rise to the third hypothesis in this research:

Research Hypothesis 3 *Classification models are suitable for a coarse identification of regions containing objects of interest, but instance segmentation techniques are more desirable when precise locations of each object is desired.*

Instance segmentation models work well when objects to be detected have closed areal shapes such as mining holes or bomb craters. However, for objects that are thin and elongated along the whole input, but cover only a small number of pixels, the bounding box predicted by the instance segmentation models is essentially the bounding box for the whole input image and the segmentation masks are either not accurate or they are incomplete. In such cases, semantic segmentation models are more useful. In semantic segmentation, the models learn to assign categories to every pixel in the input DTM. Pixel-wise class labels predicted by such models can be post-processed to create separate instances of each object or structure in the input DTM regardless of their shape. This is the final hypothesis studied in this research:

Research Hypothesis 4 *Instance segmentation models prove useful for areal objects, but do not give useful predictions for thin linearly elongated structures. Semantic segmentation models, however, give pixel-wise predictions for a given input, rendering them suitable for objects of any shape.*

The region of study for this research is the Harz mountains located in the center of Germany with a maximum altitude of 1141 meters. The region is rich in ore and mineral deposits explored and processed over thousands of years (Segers-Glocke et al., 2000; Bartels and Klappauf, 2012; Malek, 2017). The Harz region played a vital role in the history of Europe during the Middle Ages as the ore found on this region was the main source for minting coins. The city of Goslar located in this region was the residence for German kings and emperors in the 11^{th} century, as evidenced by the Imperial Palace, the largest of its kind in Germany. The region also gained importance in the late 15^{th} century after smelting silver with the help of lead ores became possible due to water drainage and technical innovations. This region is home to the ore mines of Rammelsberg and the medieval Town of Goslar which were listed as UNESCO World Heritage Sites in 1992. The Rammelsberg mine is the first industrial monument in Germany to be listed as the world heritage site. The Upper Harz Water Management System which was declared as UNESCO World Heritage Site in 2010 is

also located in this region. Currently, all the monuments in the region listed as world heritage encompass a total of 200 square kilometers. The mining activities in the region started at the end of 3rd millennium BC and left traces, as evidenced by new scientific studies on two bronze finds (Malek and Klappauf, 2017).

As evidenced by the archaeological monuments and structures explained previously, the region is home to important cultural heritage items. Therefore, it is important to carry out precise, interdisciplinary research and develop methods to identify, document and protect them. With that in mind, this research aims at incorporating self supervised deep learning techniques for detection of archaeological monuments and man-made terrain structures in the region using DTM data.

1.2. Outline

The rest of the thesis is organized as follows:

Chapter 2 gives a brief introduction to fundamental concepts in archaeology, Geographic Information System (GIS), remote sensing, Light Detection And Ranging (LiDAR) or Airborne Laser Scanning (ALS) technology, Digital Terrain Model (DTM) and its derivatives, and Deep Learning (DL).

Chapter 3 discusses previous research on this topic. It outlines previous techniques for detection of archaeological monuments, applications of DL techniques in remote sensing and LiDAR data.

Chapter 4 gives details of the datasets, archaeological objects and terrain structures of interest in this thesis and the data processing techniques.

Chapter 5 explains the core methodologies for this research including descriptions of pretext and downstream tasks in Self Supervised Learning (SSL), and different DL architectures used in each step.

Chapter 6 includes details of experiments conducted. It describes data processing methods, training tools and setups, evaluations and results.

Chapter 7 concludes the thesis with a summary and discussion of the methods, experiments and evaluation results. It also gives hints on possible future research in this direction.

2. Basics

This chapter contains an introduction to methods and concepts behind this research. Section 2.1 gives an introduction to archaeology. Fundamental concepts in GIS are explained in Section 2.2. Section 2.3 gives the definition of remote sensing and contains detailed discussions of LiDAR, DTMs and rasters derived from the DTMs, which are the main data sources for this research. Basics of DL, the underlying methodology for this thesis, are discussed in Section 2.4.

2.1. Archaeology

Archaeology comes from the Greek word 'arkhaois' meaning ancient. It is the study of recent and ancient human history using material remains. It analyzes objects created, used and modified in the past to comprehend human culture. Portable remains such as tools, clothing and decorations are called artifacts while non-portable remains such as pyramids, mining holes, and charcoal kilns are called features. While information about historic civilizations is obtained through past written records, artifacts and features could sometimes be the only sources of information about prehistoric civilizations.

The two major disciplines in archaeology are historic and prehistoric civilizations. Prehistoric archaeologists study human history from times with no written records while historic archaeologists deal with times after writing was developed. Depending on the type of artifacts and features, the time period of the phenomena and the civilizations investigated, there are multiple sub-disciplines in archaeology.

Bioarchaeology studies human remains (Martin et al., 2013), zooarchaeology is the study of animal remains from archaeological sites (Reitz et al., 1999) and paleoethnobotany researches on archaeological plant remains (Pearsall, 2015). Archaeologists in cultural resource management preserve items of value to culture such as archaeological sites, historical buildings and museums (Schiffer and Gumerman, 1977; Green and Doershuk, 1998; Praetzellis and Praetzellis, 2011).

Another field of study in archaeology, underwater archaeology, investigates the artifacts and features submerged under water and found at the bottom of lakes, rivers and oceans (Bowens, 2011). Industrial archaeology (Hudson, 2014), ethnoarchaeology (David et al., 2001), environmental archaeology (Reitz and Shackley, 2012), forensic archaeology (Cox and Hunter, 2005), paleopathology (Armelagos and Cohen, 1984) are other sub-domains in archaeology.

Finally, mining archaeology studies the process of ancient mining and ore extraction (Stöllner, 2014). It studies how raw mineral materials were extracted, prepared, used and traded. This thesis is focused on applications of deep learning techniques for automated detection of structures related to historical mining.

2.2. Geographic Information System

GIS is a framework for capture, storage, analysis and presentation of spatial data (Worboys and Duckham, 2004). The first task in a GIS workflow includes collecting spatial data through Global Positioning System (GPS), photography or remote sensing systems such as multispectral scanning

or LiDAR and storing them. The collected data are analyzed for extracting information, and then visualized and presented.

2.2.1. Spatial Reference System

Data for the same area could be collected at different times or using different acquisition devices. In order to overlay and connect information from different sources about the same geographic location, spatial reference systems are used. Spatial reference systems describe where features are located in the real world. An example of spatial reference systems is the European Petroleum Survey Group (EPSG) number system (Obe and Hsu, 2011).

2.2.2. Coordinate Reference Systems

To identify points on the reference systems, coordinate systems are used. There are two types of coordinate systems: geographic and projected coordinates.

- **Geographic coordinates** use the latitude and longitude measured in degrees. Latitude starts from the equator and goes northwards (positive) and downwards (negative) while longitude starts from the meridian in Greenwich and goes right (positive) and left (negative) in the range of -180 to 180 degrees. The advantage in such a coordinate system is that only one system is used for the whole world. The disadvantages are the use of degrees and not metric units and calculations in spherical trigonometry.
- **Projected coordinates**: The 3D shape of the earth is projected into a 2D plane using different kinds of projections, e.g., conical, cylindrical or azimuthal projections. The advantage is that now, a cartesian coordinate system could be used with metric units and calculations are done simply in planar trigonometry. However, due to the deformations in the size and shape after projection, there are multiple map projections each for different zones around the globe. Examples of projected systems include Universal Transversal Mercator (UTM) coordinates (worldwide) (Langley, 1998), and Gauss–Krüger coordinates (Germany) (Snyder, 1987).

2.2.3. Raster and Vector Data

Spatial information could be stored as raster and vector data. Vector data represent information as points, lines or polygons and are mostly used for storing discrete information such as political boundaries, rivers and lakes. Objects represented in vector format are clearly distinguished from each other as there is a crisp boundary among them. Raster data, on the other hand, are functions of feature positions, e.g., $f(x,y)$, and represent information in the form of a matrix or grid of cells. Raster data are mostly utilized for storing continuous information such as temperature, elevation or precipitation.

2.2.4. GIS Software

There are many software applications used in GIS such as ESRI ArcGIS, GRASS GIS, and QGIS, among others. ArcGIS developed by ESRI is a well known commercial software used by many organizations. The ArcGIS Desktop trinity includes a suite of ArcCatalog, ArcMap and ArcToolbox which are for geodata management, analysis and visualization, and geoprocessing operations, respectively. ArcGIS software is, however, only compatible with Windows operating system and cannot be installed on Unix-based operating systems. Another well known GIS

software is QGIS. It is open source and freely available. It is compatible with all operating systems. It can carry out most of the operations available in ArcGIS.

2.2.5. GIS Data File Formats

Vector data are stored in a format called *shapefile* that is used and compatible with most GIS software including ArcGIS and QGIS. A shapefile contains features of the common geometry, thematic attributes and spatial extent.

Raster data are stored in matrices or gridded cells where each cell or pixel is a function of point locations and denote a continuous phenomenon. Examples of file formats for raster data are JPG, GIF, PNG, and BMP, among others. The most commonly used file format is the GeoTiff file format which is platform independent and portable among different GIS software products.

2.3. Remote Sensing

Remote sensing refers to sensing, detecting and recording information about the earth's surface or targets of interest without being in contact with it (Fischer et al., 1976; Awange and Kyalo Kiema, 2013). Energy is emitted from a source towards the target surface. The reflected energy from the target is then sensed and recorded by a sensor. The reflected energy or the time and distance of travel by the energy between the source and the target is used to analyze and extract useful information about the target. Remote sensing can be carried out from the ground, the air (i.e., from an aircraft) or from the space (i.e., using satellites) (Aggarwal, 2004).

A remote sensing system is mainly composed of an energy source that emits energy, a sensor that collects and processes the reflected energy from the target. The radiated energy has two important characteristics: wavelength and frequency. Wavelength refers to the length of one wave cycle, measured in units of meters and micro and nano meters. Frequency is defined as the number of cycles of a wave passing a particular point in unit of time. It is measured in Hertz (i.e., cycles per second). Depending on the wavelength of the energy radiated from the source, the amount of reflected energy will vary for different types of matter. This helps in distinguishing different types of illuminated target surfaces (Verhoeven, 2017). The range of frequencies and wavelengths for electromagnetic radiations is called the electromagnetic spectrum. It is illustrated in Figure 2.1.

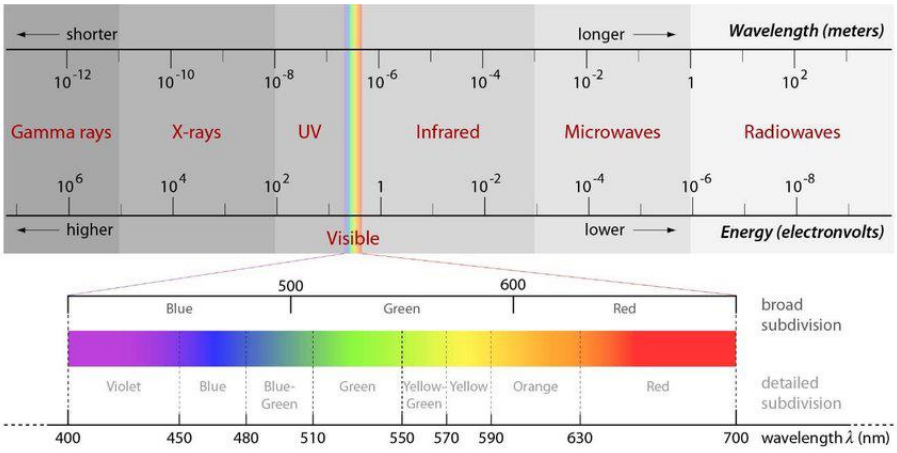


Figure 2.1.: The electromagnetic spectrum. Image from Verhoeven (2017).

A spectral remote sensing system collects light energy within specific ranges of the electromagnetic spectrum called bands. Objects on the earth's surface reflect or emit energy in unique ways. The

response for one band may not be used to distinguish a feature, while that of another band may be useful. For example, the reflections for water and vegetation using visible light might be similar while they are very distinct for infrared light. Hence, multispectral systems are used to cover different types of surfaces. Some terminologies related to remote sensing data are explained below.

Spectral Resolution

Reflected radiation within a certain wavelength is stored as one channel or band in remote sensing images. This range is called the bandwith, which determines the spectral resolution of remotely sensed data. The smaller the range of wavelength for a bandwidth, the better different objects are recognized in the data (Klemas, 2010).

Spatial Resolution

It is also referred to as the ground sampling distance and represents the size of the smallest area or object covered by the sensor at any particular instant (Cracknell, 2007). The visible area at any instant of time is usually referred to as the Instantaneous Field of View (IFOV). For a feature to be detected, its size has to be greater than or equal to the spatial resolution.

Radiometric Resolution

It refers to the number of possible different outputs a single pixel of remote sensing data could represent. This is usually measured in bits. For example, a pixel in remote sensing data with a radiometric resolution of 8 bits could store one of $2^8 = 256$ different values. The higher the radiometric resolution, the better the differences among different objects are detected.

Scale

The ratio of distance on a remote sensing image or the map to the actual distance on the ground is called the scale. For example, in a map with a scale of $1 : 100000$, an actual ground distance of of 1 km will correspond to 1 cm.

2.3.1. Passive and Active Remote Sensing

Based on the type of energy source, remote sensing systems are divided into passive and active remote sensing systems (Cracknell, 2007). Passive remote systems such as multispectral and hyperspectral cameras use natural energy sources such as the sun for illumination or radiation. However, the disadvantage is that the source is not available at all times, e.g., there is no reflected energy from the sun as the source at night. Active remote sensing systems have their own source of energy and measurements can be taken at anytime. Examples of active remote sensing systems are Radio Detection and Range (RADAR) and LiDAR systems. RADAR systems have their own source of energy. They emit microwave energy towards the target surface, and measure the time of travel (Campbell, 2002). The time is used to calculate the distance from the source to target. The distance and the amount of reflected energy for the target surface is stored and analyzed for extracting useful information about the surface.

2.3.2. LiDAR Systems

Terrestrial or airborne LiDAR systems emit laser light pulses towards the target to measure the top of target features. A LiDAR system is equipped with a GPS that records the coordinates of the light energy and an Inertial Measurement Unit (IMU) which registers the orientation of the LiDAR system. Using the coordinates, orientation, and the time of the reflected energy, it is possible to measure the target locations.

Multiple reflections might be recorded for target points creating a distribution or a waveform. LiDAR systems record information in two different ways: either individual information at the peaks of the distribution or waveform or the full distribution. Reflected light at individual points are called discrete returns and a discrete LiDAR system may record $1-4$ returns for each emitted laser pulse. Full wave LiDAR systems store the whole distribution of the reflected energy. An example illustration for airborne LiDAR is given in Figure 2.2.

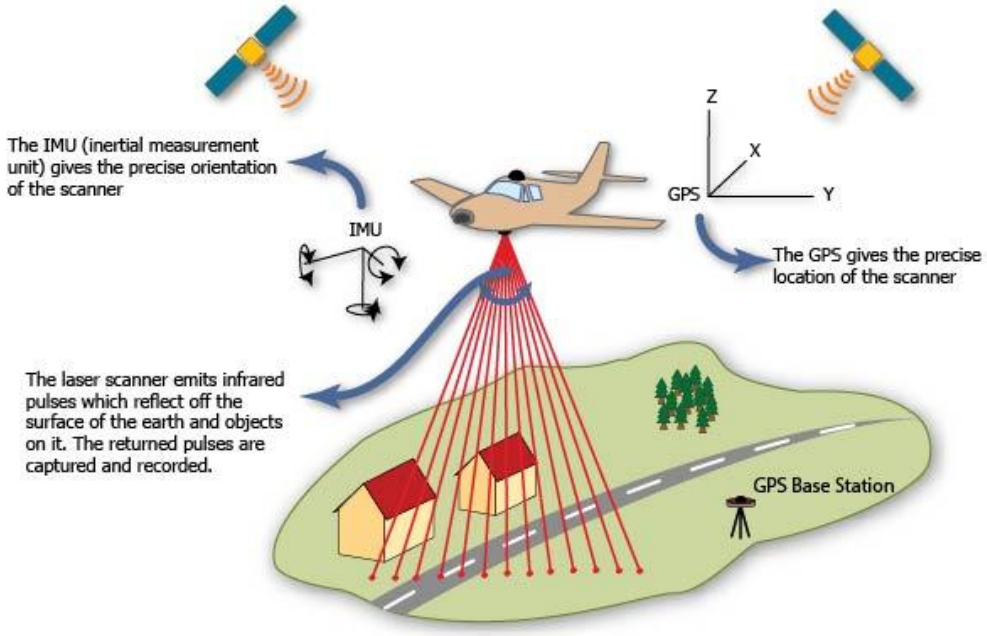


Figure 2.2.: Schematics of Airborne LiDAR Systems. Reprinted from Center for Advanced Spatial Technologies (CAST) Website. ©2021 CAST.

Collections of recorded information for points using LiDAR systems are called point clouds where each point has the x and y coordinates for location and z values representing elevations. LiDAR points can be classified as belonging to ground or non-ground surfaces such as vegetation, buildings, or other infrastructure which can be used by national mapping agencies. Some of the common class codes for LiDAR points include bare earth (class 2), low (3) to high (5) vegetation classes, and building (class 6) (Dong and Chen, 2017). While a simple LiDAR dataset could be used for 3D visualization of the point clouds, classified point clouds are used for creating DTMs. Class labels for LiDAR points can be used to filter out certain structures, e.g., buildings or trees, and retain only the points on the terrain.

2.3.3. Processing LiDAR Data

Most LiDAR point clouds are stored in files with the *.las* file format and represent a collection of discrete return points. Vector-based methods (e.g., Triangulated Irregular Networks (TINs)

(Peucker et al., 2017)) or raster based methods are used to create continuous surface models from these discrete points.

TIN is a vector-based representation of continuous information comprising of non-overlapping triangles. The triangles are created from irregularly distributed points. Delaunay triangulation is a well known method for filtering LiDAR point clouds to filter non-ground points and create triangles representing elevation surfaces for the bare earth (Axelsson, 2000). Delaunay triangulation refers to creating non-overlapping triangles using the data points as the vertices such that no other data point lies within their circumcircle, i.e., circle passing through all three vertices of the triangle. Axelsson (2000) creates a TIN by starting with a small set of points and iteratively adding new data points to the TIN if they meet the criteria, e.g., the elevation value is below a certain threshold. The process is stopped when all the points are classified as ground or non-ground. The completed TIN contains triangles faces of which represent homogeneous elevations for the surface.

Another method for representing LiDAR data as continuous infromation is the raster based method. Different spatial interpolation methods are used to create raster grids such as DTMs from LiDAR point clouds. Spatial interpolation refers to determining the value of a point based on other points within a specific distance. Examples of raster interpolation techniques are Inverse Distance Weighting (IDW), thin-plate splines and kriging, among others (Li and Heap, 2008).

2.3.4. Digital Terrain Models and Derived Rasters

DTMs model a terrain using elevation values of 2D points. It is a continuous representation of the bare earth surface. The elevation values are approximations of vertical distance from each point on the terrain to a reference surface. The reference surface is usually the mean sea level. Rasters including height values for non-ground objects such as building, vegetation and more are called Digital Surface Models (DSMs) (Hirt, 2014). Another term for DTM is Digital Elevation Model (DEM), which is used interchangeably with DTM in literature. DTMs are used for many tasks in hydrology (Quinn et al., 1991), geomorphology (Chorowicz et al., 1989; Moore et al., 1991), geography (Carrara et al., 1997), cartography (Yoeli, 1983), archaeology (Risbøl et al., 2013; Harris and Lock, 1988; Dubbini et al., 2016), planning and modeling of road systems (Liu and Sessions, 1993), forests (Maguya et al., 2014; Yu et al., 2005), and landscapes (Florinsky, 1998), and disaster management (Price and Vojinovic, 2008), among others.

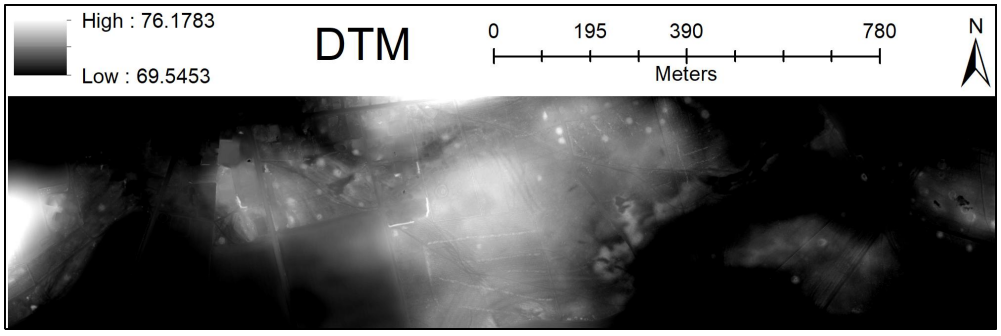


Figure 2.3.: Example DTM raster.

An example DTM raster is illustrated in Figure 2.3. It is created from an ALS point cloud using TIN with linear interpolation. To visualize DTMs , different methods are used to derive other rasters from them. The derived rasters are generally either in grayscale or color image formats. Examples of rasters derived from DTMs are slope, aspect, shaded relief, SLRM, SVF, LD, and openness (POS and NEG), among others. Each derived raster makes certain structures stand out

that are useful for visualization of objects and structures. A brief summary of the derived rasters and how they are calculated is given below.

Slope

Slope is related to the first derivative and is a raster product that indicates the steepness of a surface. It is calculated as the maximum rate of change of elevation for a point with respect to its neighboring points (Gelbman and Papo, 1984; Kokalj and Hesse, 2017). Slope is calculated using Equation 2.1.

$$\textbf{slope} = \arctan(\frac{\text{Rise}}{\text{Run}}) = \arctan(\sqrt{(\frac{\partial z}{\partial x})^2 + (\frac{\partial z}{\partial y})^2}) \tag{2.1}$$

Where Rise and Run, shown in Figure 2.4a, are the difference in elevation for a pixel with respect to a fixed number of neighboring pixels in x and y direction, respectively. Values in slope rasters are in degrees and range from 0 to 90 degrees (pixels with no information are set to -1). Slope raster calculated for the DTM in Figure 2.3 is shown in Figure 2.5.

(a) Slope *(b) Aspect*

Figure 2.4.: Slope and aspect rasters.

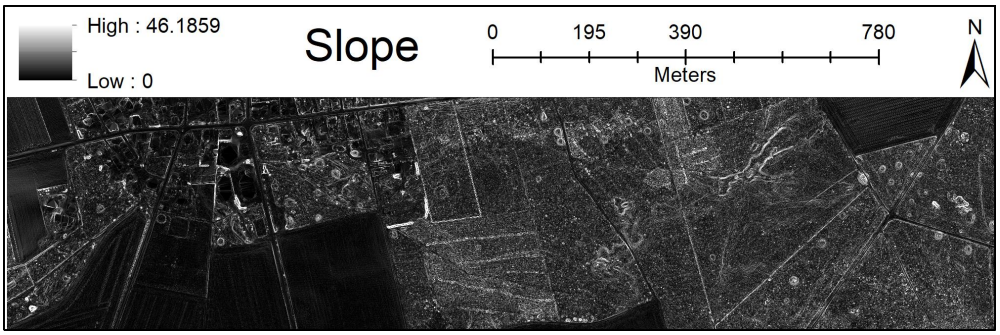


Figure 2.5.: Slope raster for the DTM in Figure 2.3

Aspect

Aspect indicates the direction of maximum rate of change in elevations with respect to the neighboring points, as illustrated by the arrow in Figure 2.4b. Aspect is calculated using Equation 2.2.

$$\textbf{aspect} = \arctan(\frac{\frac{\partial z}{\partial x}}{\frac{\partial z}{\partial y}}) \tag{2.2}$$

Values in aspect rasters are in degrees and range from 0 to 360 degrees (pixels with no information are set to -1). Aspect raster calculated for the DTM in Figure 2.3 is shown in Figure 2.6.

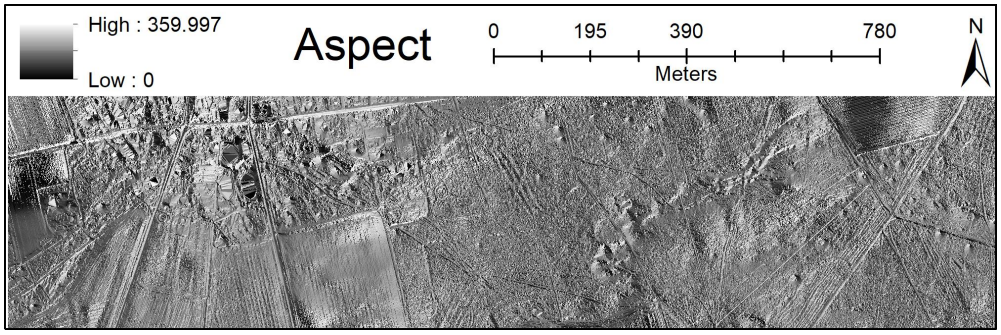


Figure 2.6.: Aspect raster for the DTM in Figure 2.3

Hillshade relief

Hillshade relief is a representation of terrain surfaces based on shadows created by a hypothetical light from a certain direction, generally the northwest. Pixels in the regions perpendicular to the light source are highly illuminated and are thus assigned a high value while those at an angle higher than 90° are dark and thus take small values (Kokalj and Hesse, 2017). To calculate hillshade, it is necessary to first calculate the slope (using Equation 2.1) and aspect (using Equation 2.2) rasters. In addition, the altitude and azimuth angles of the hypothetical light source must be defined. The altitude angle is defined as the angle between the horizontal plane that an observer is standing on and an imaginary line between the observer and the light source. Azimuth is the angle measured clockwise from the true north to the point on the plane directly below the light source. It is illustrated in Figure 2.7.

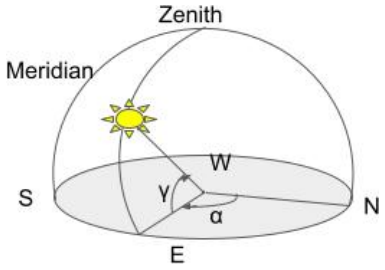


Figure 2.7.: Altitude (γ) and azimuth (α) angles with respect to a reference point and the light source.

After calculating the slope and aspect as explained previously and setting the angles for altitude (γ) and azimuth (α), the hillshade relief raster can be calculated using Equation 2.3.

$$\textbf{Hillshade} = 255.0 * ((\cos(90 - \gamma)) * \cos(S)) + (\sin(90 - \gamma) * \sin(S) * \cos(\alpha - A)) \tag{2.3}$$

Where S and A denote the slope and aspect values calculated previously.

Three hypothetical light sources from different directions can be used to create multiple shaded reliefs and represented as RGB images. An example RGB shaded relief with three different azimuth (α) values for each channel ($R = 315°$, $G = 15°$, and $B = 75°$) and a fixed altitude angle $\gamma = 30°$ is illustrated in Figure 2.8.

Sky View Factor (SVF)

SVF represents the visibility percentage of the sky for every pixel. Values in SVF rasters range between 0 and 1 and they are helpful in visualizing archaeological mining structures (Zakšek et al., 2011; Kokalj and Hesse, 2017). SVF values are calculated taking the zenith angles above the

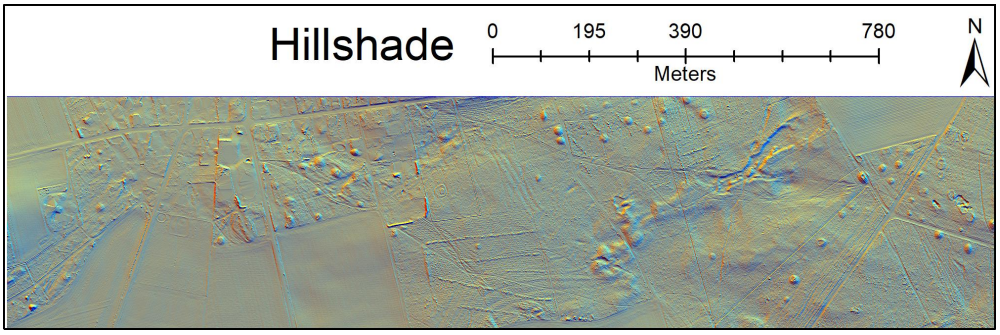


Figure 2.8.: RGB Hillshade raster for the DTM in Figure 2.3

horizontal plane into account and are therefore suitable for concave structures such as mining holes (Kokalj et al., 2013; Doneus, 2013). SVF is calculated using Equation 2.4.

$$\mathbf{SVF} = 1 - \frac{\sum_{i=1}^{n} \sin(\gamma_i)}{n} \tag{2.4}$$

Where each γ_i is the elevation angle as shown in Figure 2.9. It is calculated and normalized for n directions.

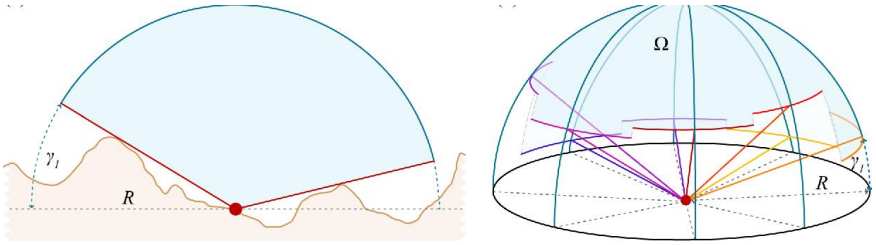


Figure 2.9.: The SVF is defined by the visible part of the sky denoted as Ω on the right image. γ_i is the vertical elevation angle in n directions with a radius of R pixels. ©(Zakšek et al., 2011).

Pixels in SVF rasters range from 0 to 1 showing the visibility of the sky. Example SVF raster calculated from the DTM in Figure 2.3 with a radius of 64 pixels for 16 directions is shown in Figure 2.10

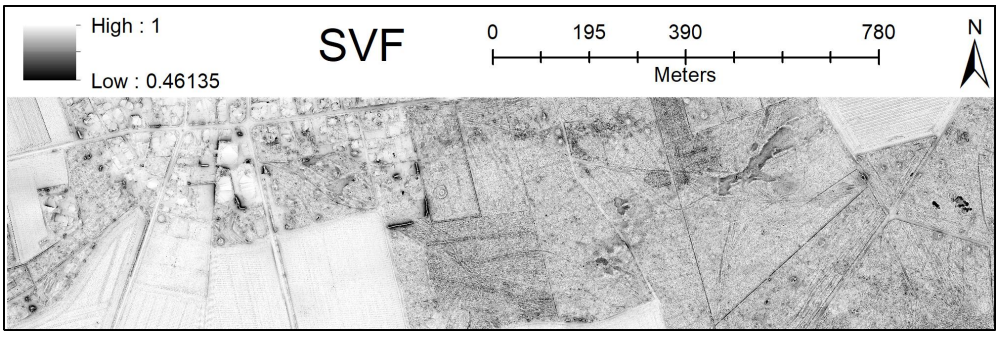


Figure 2.10.: SVF raster for the DTM in Figure 2.3

Openness

Positive openness (POS) quantifies the mean zenith angle of horizon elevations within a search radius while negative openness (NEG) is the mean nadir angle of all determined horizons (Yokoyama

et al., 2002). For each pixel, the zenith (β) and nadir (δ) angles are determined based on the profiles for multiple directions, i.e., azimuth angles α and fixed radius r. Examples for two different points and two azimuth angles are shown in Figure 2.11. For 8 different directions with azimuth angles $\alpha \in 0°,45°,90°,...,315°$ and a radius r, POS is calculated as shown in Equation 2.5.

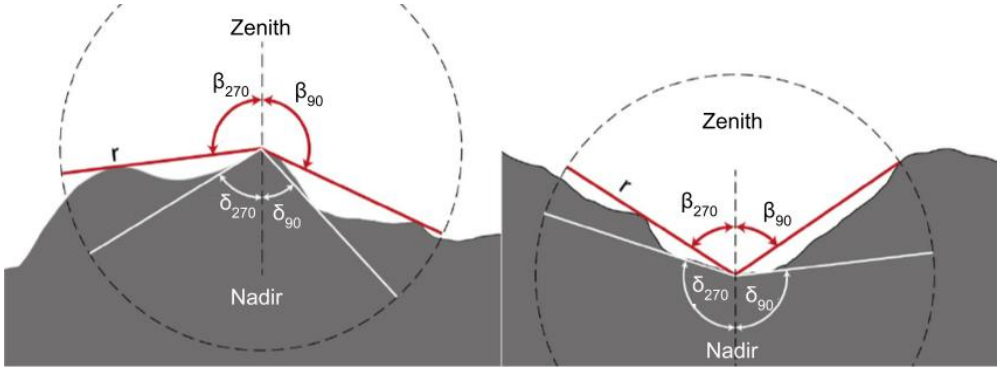


Figure 2.11.: Zenith and nadir angles used in calcuation of POS and NEG, respectively, for two pixels along two directions with azimuth angles $\alpha \in 90°,270°$. Adapted from Figure 1 in Doneus (2013).

$$\text{POS} = \frac{\beta_{r,0} + \beta_{r,45} + ... + \beta_{r,315}}{8} \tag{2.5}$$

Where $\beta_{r,45}$ denotes the zenith angle determined for a point with respect to r neighboring pixels and the direction with an azimuth angle α of 45°.

For the same setting, NEG is calculated using Equation 2.6.

$$\text{NEG} = \frac{\delta_{r,0} + \delta_{r,45} + ... + \delta_{r,315}}{8} \tag{2.6}$$

Where $\delta_{r,45}$ denotes the nadir angle determined for a point with respect to r neighboring pixels and the direction with an azimuth angle α of 45°.

Openness highlights the outlines and the highest and lowest parts of the raster. POS rasters are suitable for visualizing structures such as sunken paths, bomb craters, and ridges between holloways while NEG rasters help visualize the actual holloways, and burial mounds (Doneus, 2013). Pixel values in POS and NEG range from 0° to 180°. POS and NEG rasters calculated for the the DTM in 2.3 are illustrated in Figures 2.12 and 2.13.

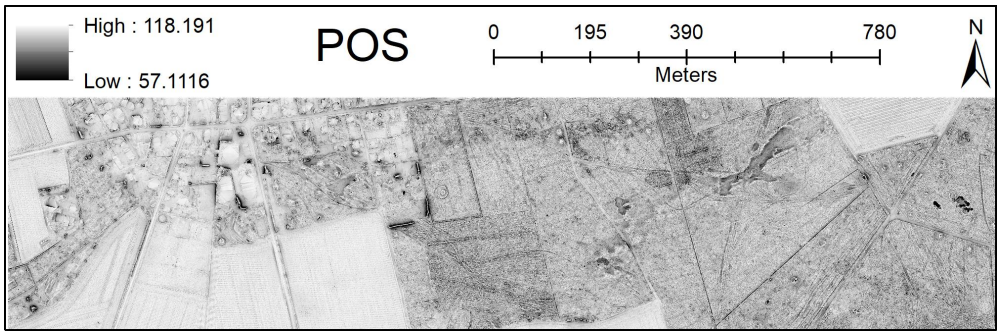


Figure 2.12.: POS raster for the DTM in Figure 2.3

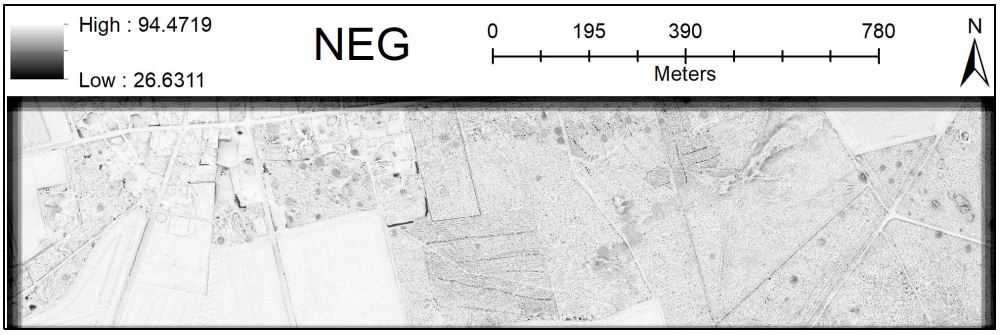


Figure 2.13.: NEG raster for the DTM in Figure 2.3

Local Dominance (LD)

LD indicates how dominant an observer would be from a specific point with respect to its neighboring points (Hesse, 2016). The dominance value for each point is calculated using the average angle at which a virtual observer standing at the said point would look down at the neighboring points within a fixed radius r. As a result, pixels at the local peaks get high dominance values and appear brighter while those at the local sinks have small dominance values and appear darker. LD is suitable for visualization of protruded features such as burial mounds and sunken features such as hollow ways. It is used in detection of archaeological mining structures and landform recognition (Kazimi et al., 2020). An LD raster for the DTM in Figure 2.3 is shown in Figure 2.14.

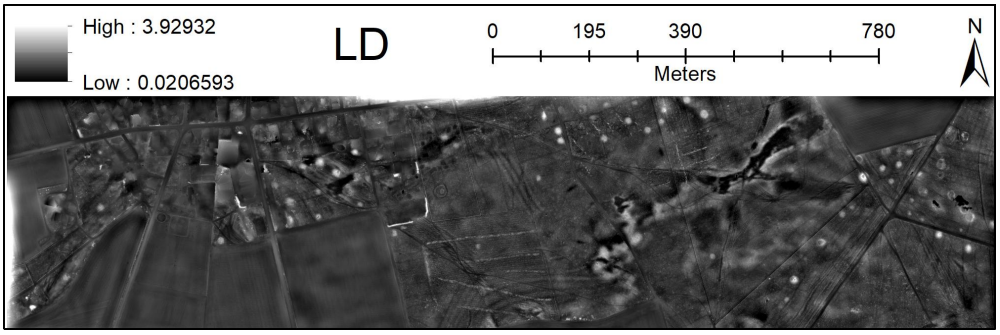


Figure 2.14.: LD raster for the DTM in Figure 2.3

Simple Local Relief Model (SLRM)

SLRM is a raster calculated from DTM and is used for separating small scale scale features from large scale landscapes and making them stand out. First, the DTM is smoothed using a low pass convolutional filter, and the result is subtracted from the original DTM to create a trend removal map. This helps separate small scale features from large landscapes, but their elevations are underestimated. To alleviate this problem, a purged DTM is calculated by creating zero contours in the trend removal map , assigning original elevations to the contour lines, and interpolating the points. The final raster created by subtracting the purged DTM from the original DTM is called SLRM and contains a less biased representation of small scale features. SLRMs are suitable for visualization and analysis of landscapes containing archaeological features such as burial mounds, mining holes, ridge and furrow fields, and charcoal kilns, among others (Hesse, 2010; Gallwey et al., 2019). SLRM is also used in small wetland detection (Leonard et al., 2012), landform analysis in geomorphology (Raper, 1989), and visualization of subtle topographic change (Orengo and Petrie, 2018). An SLRM raster for the DTM in Figure 2.3 is shown in Figure 2.15.

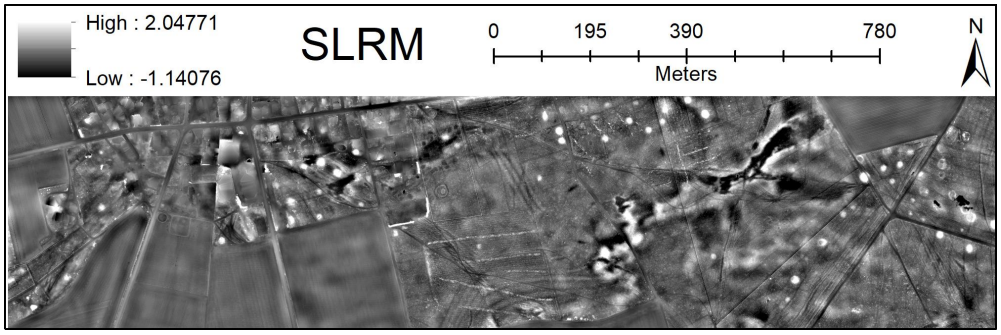


Figure 2.15.: SLRM raster for the DTM in Figure 2.3

2.4. Deep Learning

Deep learning is a technique that utilizes multiple parameterized processing layers to learn complicated structures in data by using backpropagation and gradient-based optimization algorithms (LeCun et al., 2015). A deep learning model or a DNN is composed of an input layer, an output layer and two or more hidden layers. Each layer contains one or more neurons which are basic information processing units. Input data are fed to the input layer, the hidden layers process and learn the intrinsic characteristics of the input data. The output layer produces an output that should be as similar to the the desired outcome as possible. Fundamental components of DNNs, training procedures and types of deep learning techniques are explained in the following subsections.

2.4.1. Neurons

Neurons or *nodes* are basic building blocks of neural networks. They are mathematical models inspired by the biological neuron. A neuron computes the weighted sum of its inputs, and its output depends on a so called *activation function* as shown in Equation 2.7 below.

$$a = f(\sum_{i=1}^{N} w_i x_i) \tag{2.7}$$

Where N denotes the number of input features, f stands for the choice of activation function and w_i and x_i are the i^{th} weight or *parameter* and input feature, respectively. Components of a simple neuron are illustrated in Figure 2.16.

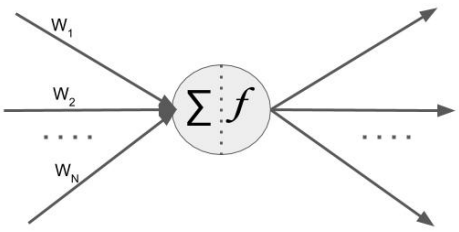


Figure 2.16.: A single neuron.

Some common activation functions are described as follows.

Sigmoid

The sigmoid function defined by Equation 2.8 squashes its input values in the range between 0 and 1.

$$f(z) = \frac{1}{\exp(-z) + 1} \tag{2.8}$$

Where z is the weighted sum of a neuron's inputs.

Hyperbolic Tangent (Tanh)

The *Tanh* function defined by Equation 2.9 normalizes the inputs in the range between -1 and 1.

$$f(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} \tag{2.9}$$

Rectified Linear Unit

The Rectified Linear Unit (ReLU) function defined by Equation 2.10 is regularly used in DNNs and transforms the inputs into the range $[0,\infty]$.

$$f(z) = \begin{cases} z & : z > 0 \\ 0 & : z \leq 0 \end{cases} \tag{2.10}$$

Exponential Linear Unit

Exponential Linear Unit (ELU), as shown in Equation 2.11 is similar to ReLU, i.e., it is an identity function for non-negative numbers. For negative numbers, ReLU becomes horizontal sharply while ELU has a constant parameter α which allows the function to become horizontal slowly until its output equals $-\alpha$.

$$f(z) = \begin{cases} z & : z > 0 \\ \alpha(\exp(z) - 1) & : z \leq 0 \end{cases} \tag{2.11}$$

2.4.2. Layers

Layers are composed of neurons or nodes. Combinations of multiple layers form a (deep) neural network. The first layer is called the *input* layer, the final layer is the *output* layer and those in between are referred to as the *hidden* layers. There are different types of layers, each of which are effective for certain types of input data and certain tasks. Common types of layers are explained in what follows.

Dense or Fully Connected Layer

As the name suggests, a dense or fully connected layer is one in which every neuron in a layer is connected to every neuron in the previous layer. A neural network with two (dense) hidden layers is illustrated in Figure 2.17.

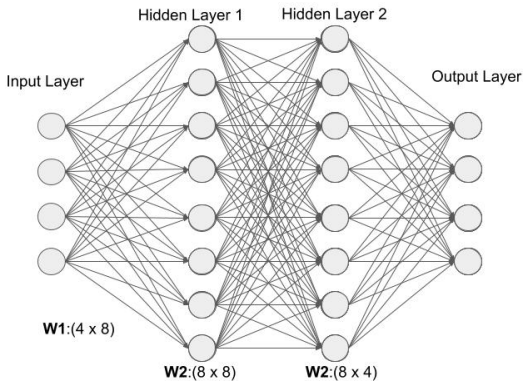


Figure 2.17.: A neural network with 2 hidden layers. Circles represent nodes or neurons, arrows represent the connection or weights between nodes and $\boldsymbol{W}$ *denotes the weight matrix.*

Convolutional Layer

Neurons in a dense layer are fully connected to those of the previous layer and each neuron functions independently without sharing any connections. Therefore, for inputs with high dimensions such as images, it does not scale well. For example, for an input image of size $227 \times 227 \times 3$, a single neuron in the first layer of a neural network needs $227 \times 227 \times 3 = 154\,587$ connections or parameters (ignoring the bias term). Adding more layers and more neurons would result in orders of magnitude larger number of parameters. High numbers of parameters lead to inefficiency in computation as well as *overfitting*. Overfitting is referred to the phenomenon that a model performs well on training data, but poorly on test or unseen data. To handle aforementioned issues, a convolutional layer proves useful. The main advantages that convolutional layers bring to the equation are *local connectivity* and *weight sharing*.

Instead of every pixel in an input volume, e.g., input image, a neuron is connected to only a small region of the input. For this connection, usually a matrix of size $k_1 \times k_2 \times d_{input}$, usually referred to as the *filter*, *kernel* or the weight matrix, is used where k_1 and k_1 are the hyperparameters for the width and height and d_{input} is the depth of the input volume. The output is calculated by taking the sum of element-wise multiplication of the filter F with a small $k_1 \times k_2 \times d_i$ region R of the input. The convolution for two-dimensional filters, F, and inputs, I, are shown in Equation 2.12.

$$Z(i,j) = (I * F)(i,j) = \sum_{m}^{k_1} \sum_{n}^{k_2} I(m,n)F(i-m,j-n) \tag{2.12}$$

This operation is applied to other regions of the input, but with the same *kernel* or weights. In other words, every local region R is connected to the neurons in the following layer by the same weights instead of independent weights for each region. This leads to a huge reduction in the number of parameters. For example, for the same input of size $227 \times 227 \times 3$ and a kernel matrix of size $5 \times 5 \times 3$, there are only $5 \times 5 \times 3 = 45$ parameters (ignoring the bias term) required. The output of convolutional layers maintain the grid structure, like image inputs, but the spatial dimensions and the depth are controlled by hyperparameters such as *stride*, *padding* and the *size* of the feature maps. **Stride** refers to the distance between neighboring positions of the filter matrix F on the input volume while calculating the output. The spatial dimensions of the output volume are inversely proportional to the stride size as shown in Equation 2.13. In some cases, it is desirable to keep the spatial dimensions of the output volume similar to those of the input. Therefore, the input is first padded with zeros and the convolution is applied to the padded version. In general, the output size for a convolutional layer is calculated as follows.

$$W_{\text{output}} = \lfloor \frac{W_{\text{input}} - f + 2p}{s} \rfloor + 1 \tag{2.13}$$

Where W_{output} and W_{input} denote the dimensions (height or width) of the output and input volumes, f, s and p stand for the filter size, stride size and the size of zero padding, respectively. For example, for an image of size $227 \times 227 \times 3$, a kernel matrix of size $5 \times 5 \times 3$, a stride size of 2, and no zero padding, the height and width of the output volume is calculated to be $H = W = \lfloor \frac{227-5+2\times0}{2} \rfloor + 1 = 112$, using Equation 2.13. The output, with a size of 112×112 is a so called feature map produced by a single kernel matrix of size $5 \times 5 \times 3$ that specializes in detecting a specific feature, e.g., edges, in all regions of the input volume. To learn parameters that specialize at detecting multiple different features, more kernel matrices can be introduced into the layer and this will define the depth of the output volume referred to as the feature maps. For example, for the input size of $227 \times 227 \times 3$ and 96 different kernels of size $5 \times 5 \times 3$ the final output volume will be of size $112 \times 112 \times 96$. This brings the number of parameters for the layer to be $5 \times 5 \times 3 \times 96 = 7200$. The overall structure of a single convolutional layer is illustrated in Figure 2.18.

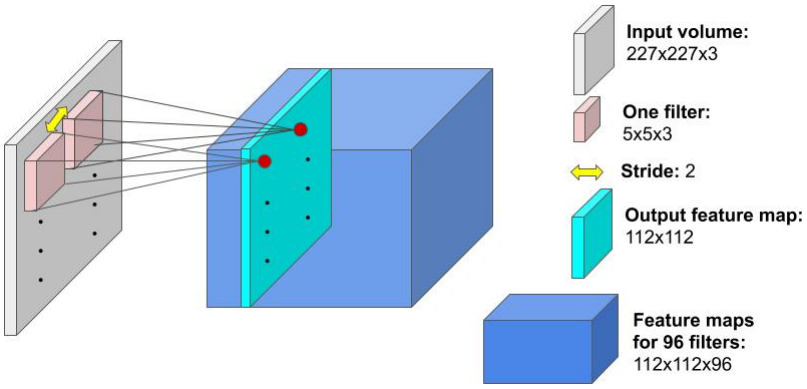


Figure 2.18.: A single convolutional layer. No zero padding is used in this example.

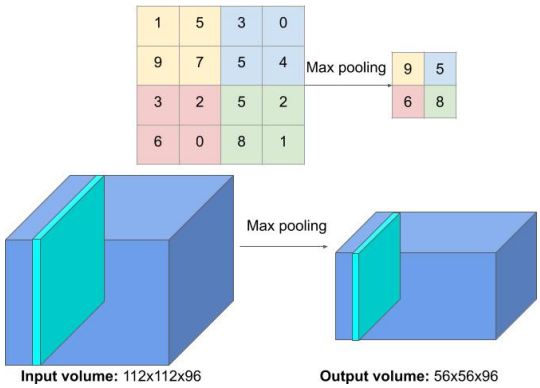


Figure 2.19.: Max pooling operation shown for a volume of size $112 \times 112 \times 96$ *with a window size of* 2×2 *and strides of* 2*. For clarity, it is also shown using a single slice input of* 4×4 *pixels.*

Convolutional layers are usually followed by another operation called *max pooling* introduced below:

– **Max pooling** refers to downsampling the input. It helps avoid overfitting and adds translation invariance to the network. Due to downsampling, the computational cost is also reduced. Max pooling is carried out by scanning the input using a small window and giving the maximum value inside the window as the output. It is illustrated in Figure 2.19.

2.4.3. Objective Functions

Objective functions, also referred to as the *loss*, *cost* or *error* function in the literature, are the core of training a DNN. The goal is to train a model that makes predictions as similar to the ground truth, i.e., expected outcome, as possible. This is facilitated by minimizing the objective function which varies based on the type of task and the nature of the expected output. Some of the commonly used objective functions are explained in this section.

Mean Squared Error

Mean Squared Error (MSE), as shown in Equation 2.14, is a function that calculates the average squared difference between the values estimated by the model, $\hat{y}$, and the actual ground truth, y. It is also referred to as the *quadratic* or *L2* loss in literature.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{2.14}$$

Where N denotes the number of examples. MSE is suitable for regression tasks where the expected output values are continuous real numbers such as house prices, temperature, etc.

Mean Absolute Error

Mean Absolute Error (MAE), as shown in Equation 2.15, is a function that calculates the average absolute difference between the estimated values by the model, $\hat{y}$, and the actual ground truth, y. It is also referred to as the *L1* loss in literature.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|(\hat{y}_i - y_i)| \tag{2.15}$$

Where N denotes the number of examples. MAE, like MSE, is also suitable for regression tasks.

Cross Entropy

Cross Entropy (CE), as shown in Equation 2.16, is a function that calculates the difference between two distributions. It is mainly suitable for classification tasks where the values are discrete numbers. In the case of neural networks, the distributions are the predicted output by the model, $\hat{y}$, and the ground truth labels, y represented as one-hot vectors to imitate a distribution.

$$CE = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{N} y_{ij}\mathbf{log}(\hat{y}_{ij}) \tag{2.16}$$

Where M and N denote the number of examples and number of categories, respectively. It is necessary to point out that the output of a neural network might not necessarily represent a probability distribution. To facilitate this, the final activation function for multi-class classification networks is a special function called *softmax*. As shown in Equation 2.17, the softmax function squashes its input values in the range of 0 to 1 such that their sum over all class labels equals 1 in order to represent a probability distribution.

$$\sigma(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^{N}\exp(z_j)} \tag{2.17}$$

for $i = 1,...,N$, and $\mathbf{z} = (z_1,...,z_N) \in \mathbb{R}^N$ where N is the number of categories or the length of the output vectors.

In the case of two categories, i.e., $N = 2$, the CE function becomes a special case referred to as the Binary Cross Entropy (BCE).

2.4.4. Evaluation Metrics

While objective functions are used to train the models, evaluation metrics are used to evaluate their performance. Common evaluation metrics are explained here. Before going into the details of the evaluation metrics, there are a number of terms that need to be clarified.

- **True Positive (TP)**: In binary classification, there are mainly two categories, i.e., 1 or positive and 0 or negative. True positive (TP) is the number of correct positive predictions. In other words, it is the number of examples with the positive label 1 which were correctly labeled as 1 by the model.
- **True Negative (TN)**: It is the number of correct negative predictions. In other words, it is the number of examples with the negative label 0 which were correctly labeled as 0 by the model.
- **False Positive (FP)**: It is the number of negative examples falsely classified as positive by the model.
- **False Negative (FN)**: It is the number of positive examples falsely classified as negative by the model.

Accuracy

Accuracy is defined as the fraction of correct predictions by the model. It is shown in Equation 2.18 below.

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.18}$$

Precision

Precision is defined as the fraction of positive predictions that actually belongs to the positive class. It is shown in Equation 2.19.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.19}$$

Recall

Recall is defined as the fraction of positive examples correctly predicted by the model. It is shown in Equation 2.20.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.20}$$

F1-score

A model that labels all the examples as positive will have a 100% recall, but very low precision. On the other hand, if it only assigns a positive label to very few examples of the positive class and labels everything else as negative, it will have a high precision but low recall. Therefore, a better evaluation metric that is a trade-off between good precision and recall scores is the F1-score. It is the harmonic mean of precision and recall defined by Equation 2.21.

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.21}$$

Accuracy, precision, recall and F1-score are common evaluation metrics for classification tasks.

Intersection over Union (IOU)

Intersection Over Union (IOU), also called the Jaccard Index, denotes how well the predicted output matches the ground truth. It is suitable for semantic segmentation tasks and shown in Equation 2.22 below.

$$\text{Jaccard}(y,\hat{y}) = \text{IOU}(y,\hat{y}) = \frac{\|y \cap \hat{y}\|}{\|y \cup \hat{y}\|} = \frac{TP}{TP + FP + TN} \tag{2.22}$$

Where y and $\hat{y}$ denote the ground truth and predicted output, respectively.

Mean Average Precision

Mean Average Precision (MAP) is the standard evaluation metric for instance segmentation tasks. In binary classification, the model outputs a probability score between 0 and 1. To assign a positive or a negative label to the example, a threshold is set. For example, the threshold can be set to 0.5 and for any example, if the predicted probability is above 0.5, it is labeled as positive, and negative otherwise. The threshold can be increased or decreased for optimal labeling depending on the task. Using different threshold values, the precision and recall can be calculated, and the area under the curve for precision and recall can be used in calculating the MAP score. An example of precision-recall curve is shown in Figure 2.20.

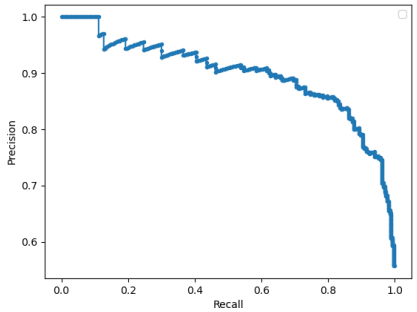


Figure 2.20.: An example of precision-recall curve.

MAP is defined as the area under the precision-recall curve averaged for n categories, as shown in Equation 2.23.

$$\text{MAP} = \frac{1}{n} \sum_{k=1}^{n} \text{AP}_k \tag{2.23}$$

Where AP denotes the area under the precision-recall curve and is defined in Equation 2.24.

$$\mathrm{AP} = \int_0^1 p(r)\, dr \tag{2.24}$$

2.4.5. Backpropagation

Backpropagation is an algorithm that is used in training a neural network. It calculates the gradient of the error function with respect to parameters of the network. The gradients are used to update the parameters in order to minimize the objective or loss function. Backpropagation makes use of the chain rule of calculus to reuse the calculated gradients with respect to parameters in a layer, l, in calculating the gradients with respect to parameters in the previous layer, $l-1$. The chain rule of calculus states that if $y = f(u)$ and $u = g(x)$, then the derivative of y is calculated using Equation 2.25.

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u}\frac{\partial u}{\partial x} \tag{2.25}$$

The procedure for backpropagation, or calculating the gradients of the error function with respect to the network parameters is as summarized as follows.

- Give an input x, pass it through the hidden layers all the way to the output layer, and store all intermediate values.
- Compute the error function E using the output and the desired target.
- Calculate the gradients of the error function E with respect to parameters in the output layer.
- Use the chain rule of calculus to calculate the gradients for the previous layer $l-1$ by reusing the calculated gradients in layer l and the stored values, all the way to the input layer.
- Use the computed gradients to update the paramters of the network.

A more detailed explanation of backpropagation can be found in (Nielsen, 2015).

The partial derivatives by backpropagation are used to decide how much and in which direction to change each parameter in the network in order to minimize the error. The way in which the parameters are changed using gradient descent is explained in the following section.

2.4.6. Gradient Descent

Backpropagation calculates the gradient of the error function with respect to parameters of the network. It is an indicator of how changing the value of each parameter affects the error. Gradient descent is an algorithm that uses the calculated gradients to tweak the parameters in order to minimize the error. It can be applied in three different ways.

Batch Gradient Descent

In batch gradient descent, the parameters are updated once at each run for all the examples in the dataset using Equation 2.26.

$$\mathbf{W} = \mathbf{W} - \eta \nabla_{\mathbf{W}} E(\mathbf{W}) \tag{2.26}$$

Where $\mathbf{W}$ is the network parameters and ∇ and E denote the gradient operation and error function, respectively. η is referred to as the learning rate determining the magnitude of the steps to take towards the minimum error.

Batch gradient descent is very slow as it performs one update for the whole dataset. For batch gradient descent, the whole dataset needs to fit into memory as well.

Stochastic Gradient Descent

In Stochastic Gradient Descent (SGD), the parameter is updated for each example in the dataset at each run.

$$\mathbf{W} = \mathbf{W} - \eta \nabla_{\mathbf{W}} E(\mathbf{W}; x_i; y_i) \tag{2.27}$$

Where x_i and y_i are the i^{th} input and output pair in the dataset.

SGD is faster than the batch gradient descent as the parameters are frequently updated, but this also leads to fluctuation in the error graph making it harder to reach the exact minimum.

Mini-batch Gradient Descent

To overcome the shortcomings of both batch and stochastic gradient descent, mini batch gradient descent comes to play. Instead of updating the parameters once for the whole dataset (like batch gradient descent) or once for each training example (like SGD), it performs one update for a small batch of n examples from the training data. This is usually referred to as the *batch size* in deep learning research.

$$\mathbf{W} = \mathbf{W} - \eta \nabla_{\mathbf{W}} E(\mathbf{W}; x_{i:i+n}; y_{i:i+n}) \tag{2.28}$$

Thus, the loss does not fluctuate, like it does for SGD, resulting into a more stable convergence. Additionally, it takes advantage of computational resources such as GPUs to make fast matrix multiplications for the mini batch, leading to faster training.

2.4.7. Gradient Descent Optimization Algorithms

The gradient descent variants explained in Section 2.4.6 need a hyperparameter, the learning rate η, to be defined in advance. The performance and speed of the model highly depends on the value of the learning rate. If it is too small, the model will take a very long time to converge. On the other hand, if it is too big, the model may never converge and jump over the minima. It could also get stuck in a suboptimal local minima or a saddle point that it may not be able to get out of. To deal with such challenges, there are a number of optimization algorithms, some of which are explained in this section.

Mini Batch Gradient Descent with Momentum

In mini batch gradient descent with momentum, instead of using only the gradients at the current pass, an exponentially decaying average of the gradient in all the previous passes are used to update the parameters. This helps accelerate towards the minima and reduces the fluctuations.

$$\mathbf{V}_t = \beta_1 \mathbf{V}_{t-1} + (1 - \beta_1) \nabla_{\mathbf{W}} E(\mathbf{W}) \tag{2.29}$$

Where $\mathbf{V}_t$ is the exponentially decaying average of gradients until time step t. β_1 is a hyperparameter for the momentum usually set to 0.9. The parameters are then updated as follows.

$$\mathbf{W} = \mathbf{W} - \eta \mathbf{V}_t \tag{2.30}$$

Root Mean Square Propagation

Root Mean Square Propagation (RMSProp) uses the exponentially decaying average of squared gradients to update the parameters which results in an even smoother path towards the minima and reduced fluctuations in the error.

$$\mathbf{S}_t = \beta_2 \mathbf{S}_{t-1} + (1 - \beta_2)(\nabla_{\mathbf{W}} E(\mathbf{W}))^2 \tag{2.31}$$

Where S_t denotes the exponentially decaying average of the gradients squared. The parameters are then updated as follows.

$$\mathbf{W} = \mathbf{W} - \eta \frac{\nabla_{\mathbf{W}} E(\mathbf{W})}{\epsilon + \sqrt{\mathbf{S}_t}} \tag{2.32}$$

Adaptive Moment Estimation

Adaptive Moment Estimation (Adam) (Kingma and Ba, 2015) uses a combination of both techniques in gradient descent with momentum and RMSProp to update the parameters. It keeps exponentially decaying averages of the past gradients as well as the past gradients squared.

$$\mathbf{W} = \mathbf{W} - \eta \frac{\frac{\mathbf{V}_t}{1-\beta_1}}{\epsilon + \sqrt{\frac{\mathbf{S}_t}{1-\beta_2}}} \tag{2.33}$$

Where β_1 and $\mathbf{V}_t$ are the hyperparameter and exponentially decaying average of the past gradients from Equation 2.29, and β_2 and $\mathbf{S}_t$ are the hyperparameter and exponentially decaying average of the past gradients squared from Equation 2.31.

2.4.8. Supervised Learning

In supervised learning, the dataset, $\mathcal{D}$, is composed of N inputs and labels, $\mathcal{D} = \{(x_1,y_1), (x_2,y_2),...,(x_{N-1},y_{N-1}),(x_N,y_N)\}$ and the goal is to train a model that associates each input example to its corresponding label, $f: \mathbf{X} \rightarrow \mathbf{Y}$.

Based on the type of target values, supervised learning is divided in to two categories: classification and regression. In classification, the target value, y_i, is one of C discrete classes. The model is trained to produce output vectors, $\vec{\hat{y}}$, of length C representing probability distributions where the probability, $\hat{y}_{y_i}$ at y_i^{th} index is maximized. In such tasks, *softmax* is generally used as the activation function for the final layer of the network to squash the predictions and imitate a probability distribution (in the case of two classes, the sigmoid activation function is commonly used). The suitable objective function is cross entropy. Some examples of classification tasks are recognizing handwritten digits in images, assigning a label to an input image, and sentiment analysis from textual data, among others.

In regression, the target value, y_i, is a continuous real number. The model is trained to produce an output, $\hat{y}_i$, which is as similar to the label as possible. Based on the range of possible values

for the labels, suitable activation functions for the final layer of the networks used in regression are sigmoid (for values between 0 and 1), Tanh (for values between −1 and 1), ReLU (for positive real numbers) or *linear*, i.e., identity function (for the range $(-\infty,\infty)$). Objective functions for such tasks are binary cross entropy (for sigmoid as the final activation function), MSE or MAE. An example of regression task is predicting house prices taking the house specifications such as number of rooms, location, etc., into consideration.

Common supervised learning tasks in computer vision are image classification, semantic/instance segmentation and object detection, among others, some of which are explained below.

Image Classification

Image classification is defined as the task of assigning a label (from a set of predefined categories) to a given input image. The application of DL models for image classification or other visual tasks gained its popularity after Krizhevsky et al. (2012) won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015) with a record breaking performance. The model they used is commonly known as *AlexNet* and is composed of 5 convolutional, max pooling and dropout layers followed by 3 dense or fully connected layers. It is illustrated in Figure 2.21. It uses two GPUs during training to speed up computation and uses data augmentation and dropout to avoid overfitting. Data augmentation refers to increasing the size of training dataset using already existing examples. For training *AlexNet*, they randomly crop images, apply random translation and flipping which leads to an increase in dataset size by a factor of 2048. Dropout, shown in Figure 2.22, refers to dropping out some of the neurons in a layer at random during training. This avoids the neurons in a layer from co-adapting and leads to better generalization (Srivastava et al., 2014). The model achieved a top 5 error rate of 15.4%, where the next best method by the Intelligent Systems and Informatics (ISI) team from the University of Tokyo achieved a top 5 error rate of 26.2%.

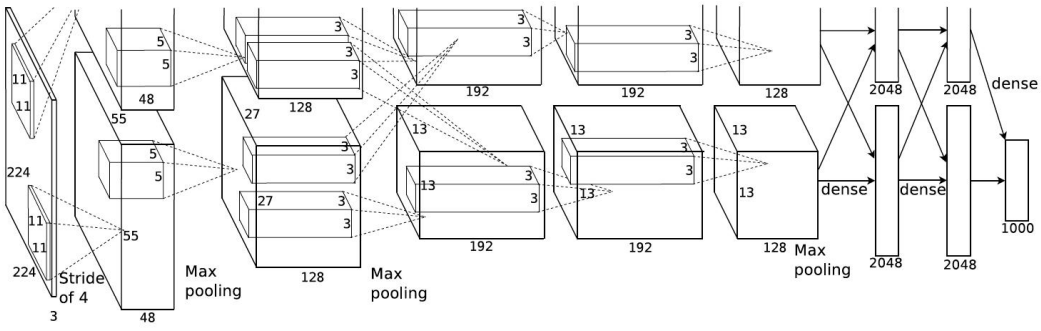


Figure 2.21.: Architecture of AlexNet (Krizhevsky et al., 2012).

Another image classification network named VGGNet was proposed by Simonyan and Zisserman (2015) that allowed higher number of layers due to filters of size 3×3 as opposed to 11×11 and 5×5 in the initial convolutional layers resulting into improved classification accuracy.

Instead of stacking layers sequentially, Szegedy et al. (2015) proposed the inception module which uses 3 convolutional layers with filter sizes of 1×1, 3×3 and 5×5 and a max pooling layer in parallel and concatenates their outputs. The 1×1 convolution, inspired by Lin et al. (2014), learns fine-grained representations while the 5×5 convolution learns abstract representations. As the number of convolutional layers grow, it is common practice to increase the depth, i.e., the size of the feature map and decrease the spatial dimensions of the outputs. Multiple convolution and max

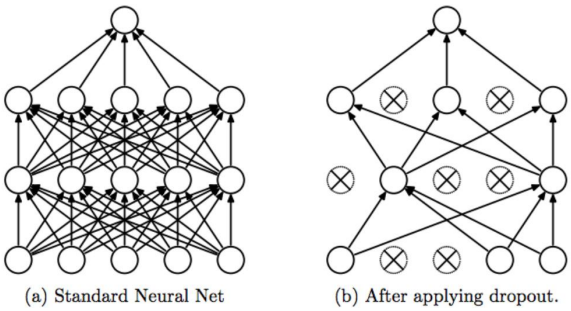


Figure 2.22.: Dropout (Srivastava et al., 2014).

pooling layers in parallel lead to a high number of parameters, consequently causing overfitting, and making the computations costly. To avoid this problem, an extra 1×1 convolution is added before the 3×3 and 5×5 convolutions and after the max pooling layer in order to reduce the depth, and hence the number of parameters. The complete inception module is illustrated in Figure 2.23

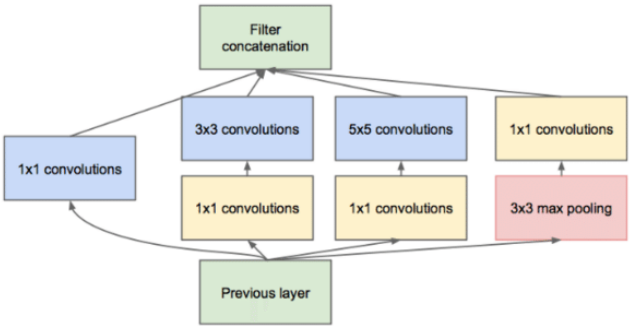


Figure 2.23.: The inception module (Szegedy et al., 2015)

As the number of layers in DL models grow, it becomes difficult to train and optimize them. He et al. (2016) show that the training error increases with the size of a network. However, they create a Deep Convolutional Neural Network (DCNN) called Residual Network (ResNet) that uses the so-called residual modules. They argue that adding extra layers to a model should either increase the model's performance or keep it the same, but not decrease it. Residual modules are composed of weighted layers and nonlinear activation functions (i.e., ReLU) and an identity function. The idea is that if a smaller network is already optimal, the weighted layers are pushed to output zeros, effectively keeping the final output the same. Otherwise, the weighted layers help improve performance. The residual module is illustrated in Figure 2.24

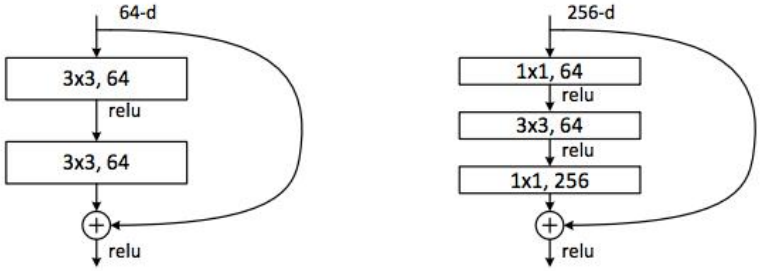


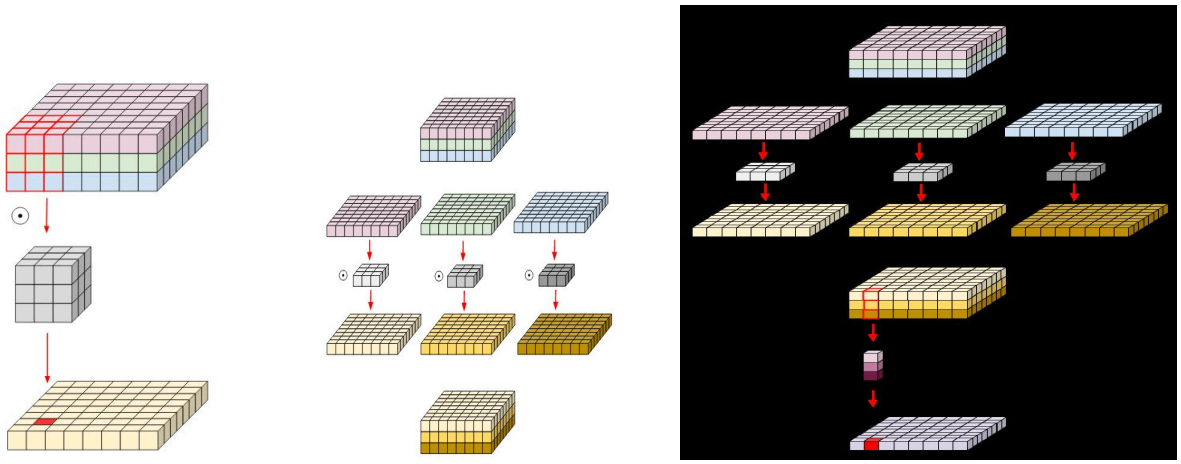
Figure 2.24.: The residual modules (He et al., 2016). The left module is used for smaller networks while the right one is used in bigger networks

Xception (Chollet, 2017) is another well known DCNN that is inspired by the Inception (Szegedy et al., 2016) and ResNet (He et al., 2016) models. It uses a combination of depthwise separable convolutions (Sifre and Mallat, 2014) and residual modules. While regular convolution applies a kernel through the spatial dimensions and the channel dimension or depth of the input volume at once and simultaneously learns cross-channel and spatial correlations, the inception module (Szegedy et al., 2015, 2016) performs this operation in two steps. It first applies a 1×1 convolution across the channel dimension and then a 3×3 or 5×5 convolution across the spatial dimensions. The Xception model uses a more extreme version of the inception module, i.e., the convolution along the spatial dimension is applied separately for every channel. Moreover, the order of operations is inverse, i.e., first the spatial convolution over each input channels is performed, and then a pointwise convolution is applied. This leads to fewer number of parameters which in turn results in faster computation and improvement in generalization. The depthwise separable convolution is shown in Figure 2.25. For an input volume of dimensions $128 \times 128 \times 3$ and an output feature map of size 16, a normal convolution with a kernel size of 5×5 would require:

- $3 \times 5 \times 5 \times 16 = 1200$ parameters.
- $3 \times 5 \times 5 \times 128 \times 128 \times 16 = 19.7$ million operations.

For the same input and output, a depthwise separable convolution would require:

- $3 \times 5 \times 5 + 3 \times 16 = 123$ parameters.
- $3 \times 5 \times 5 \times 128 \times 128 + 128 \times 128 \times 3 \times 16 = 2$ million operations.



(a) Normal Convolution *(b) Depthwise Convolution* *(c) Depthwise Separable Convolution*

Figure 2.25.: Normal, Depthwise and Depthwise separable convolutions. Image Source: eli.thegreenplace.net

Other well known image classification models include MobileNets (Howard et al., 2017; Sandler et al., 2018), DenseNet (Huang et al., 2017), and EfficientNet (Tan and Le, 2019), among others.

Classification models take input images and learn abstract representations through multiple weighted layers. Commonly, the initial layers are thin with big spatial dimensions and deeper layers get wider and have smaller spatial dimensions. The final layer is designed to output vectors representing class probabilities for the given input. Outputs of layers before the final classification layer are referred to as *features* in deep learning literature. For other vision tasks where the goal is not to assign a label to the input image, but to learn pixel-wise labels or detect bounding box locations of objects in the input, deep learning classifiers discussed previously can be used as *feature extractors*. The goal is to learn and extract discriminative features of images using the

classification models and use them as input to further layers designed for semantic segmentation, instance segmentation or object detection, explained as follows.

Object Detection

Object detection refers to finding the location and the class label for each object in the image. The locations are encoded as rectangular bounding boxes. The general steps in object detection methods are identifying informative regions in the input, extracting relevant features and classifying them. A well-known object detection framework is called Region-Based Convolutional Neural Networks (RCNN) (Girshick et al., 2014) with three distinct stages. The first stage is for identifying informational regions in the input and proposing possible candidate bounding boxes that may contain an object. This is done using the selective search algorithm (Uijlings et al., 2013). The selective search algorithm initially partitions the input regions and then iteratively groups similar regions based on color, texture, size or shape. The final bounding boxes are then passed on to the next stage. The second stage, i.e., feature extraction, makes use of Convolutional Neural Networks (CNNs). The regions proposed by the selective search algorithm in the previous stage are resized to a fixed size before being fed to the CNN for feature extraction. CNN outputs, which are learned discriminative features for each proposed region, are then fed to the final stage. The final stage has two branches. The first branch contains multiple classifiers, i.e., one for each of the predefined categories, which perform a binary classification, i.e., object versus background, using SVMs. The second branch is a regression module which learns bounding box coordinates for the object in the corresponding proposed region using the features learned by the CNN. The RCNN framework is illustrated in Figure 2.26.

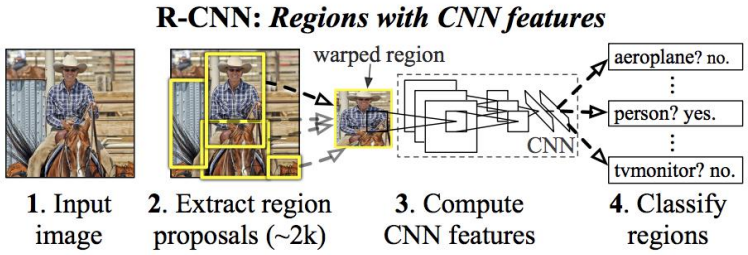


Figure 2.26.: RCNN Framework (Girshick et al., 2014)

Instead of computing features for each proposed region separately, Girshick (2015) developed Fast RCNN that extracts features from images once and then applies the region proposal algorithm on the computed features. A Region of Interest (ROI) pooling layer is then used to extract fixed-length feature vectors from the proposed regions and feed them to fully connected layers and eventually to the final branch for bounding box regression and classification. Instead of SVM classifiers, the final classification module is a multiclass softmax classifier. Fast RCNN is illustrated in Figure 2.27.

The region proposal stage in RCNN and Fast RCNN is a bottleneck for the previously explained frameworks and takes a lot of time compared to the remaining stages. Ren et al. (2016) proposed Faster RCNN, replacing the region proposal algorithm with a Region Proposal Network (RPN). The RPN takes an input image of any size and produces rectangular outputs each of which are paired with an objectness score representing the probability of the rectangular region belonging to an object or the background. The RPN network shares computation with convolutional layers from the CNN model for feature extraction in the first step. The output feature map of the last shared convolutional layer is fed to a small network for region proposal generation. A sliding window approach is used to map an $n \times n$ spatial window from the feature map into a lower dimensional

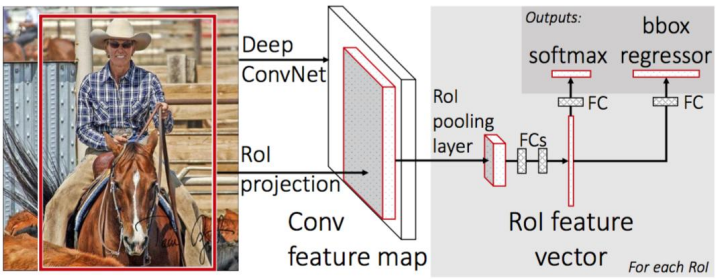
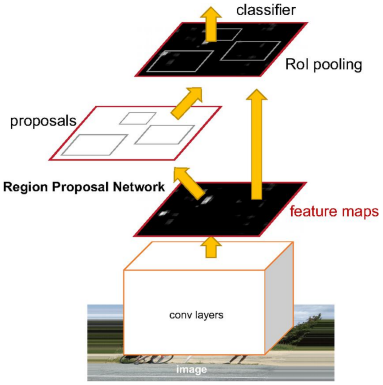


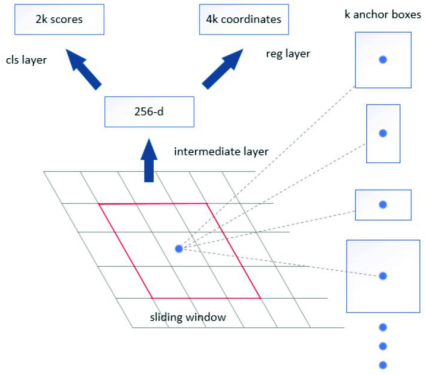
Figure 2.27.: Fast RCNN Framework (Girshick, 2015)

feature vector. The vector is then fed to the following layers for classification and bounding box regression.

The Faster RCNN framework and the RPN module that it uses are illustrated in Figure 2.28.



(a) Faster RCNN



(b) RPN Module used in Faster RCNN

Figure 2.28.: Faster RCNN and the RPN module (Ren et al., 2016)

Objects in images come at different scales essentially making them hard to detect. A standard solution to this is using feature pyramids based on image pyramids (Adelson et al., 1984). Lin et al. (2017b) introduce Feature Pyramid Network (FPN) which creates feature pyramids in a two steps. In the first step called the bottom-up pathway, the input image is processed by a DCNN and feature maps at different scales are stored to be used for creating feature pyramids. The second step is called the top-down pathway. In this this step, the final output feature map with a small spatial resolution but high level semantic information is upsampled and merged with the corresponding feature maps from the bottom-up pathway. The process is repeated until the feature map with the biggest spatial resolution in the bottom-up pathway is merged with the last upsampled feature map from the top-down pathway. The final merged feature maps at different resolution are passed on to the RPN. FPN, illustrated in Figure 2.29, is architecture-independent and can use any DCNN such as ResNet, VGGNet and others. Additionally, it can be incorporated in any of the previously explained object detection frameworks, e.g., Fast RCNN or Faster RCNN.

Instance Segmentation

Object detection frameworks explained previously are designed to learn bounding box coordinates and class labels for objects contained in an input image. The bounding box represents a coarse, rectangular location for an object, but if a precise outline for each instance in the input image is desired, a method called instance segmentation is applied. In addition to bounding box

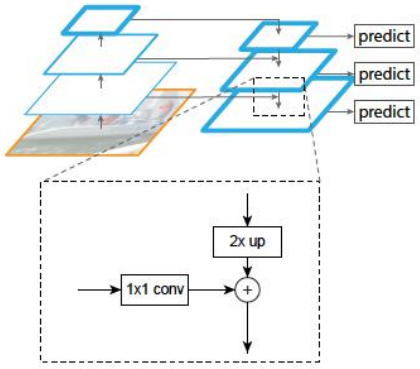


Figure 2.29.: FPN Structure (Lin et al., 2017b)

coordinates and class labels, instance segmentation models also learn pixel-wise segmentation masks. A well-known architecture for this task is called Mask RCNN (He et al., 2017), illustrated in Figure 2.30. It is built on top of Faster RCNN (Ren et al., 2016) with an additional branch for segmentation. Moreover, the ROI pooling layer in Faster RCNN is a max pooling operation on a discrete grid based on a region proposal. It is used to create fixed-length feature vectors for regression and classification layers in the detection framework. While this does not hurt the classification because of its robustness to small translations, it has a negative effect on pixel-wise segmentation mask predictions. To alleviate this problem, the ROI pooling is replaced with a so-called RoIAlign operation in Mask RCNN. RoIAlign uses bilinear interpolation in order to preserve the exact spatial correspondence between the pixels. This small change in the architecture results into more accurate segmentation masks.

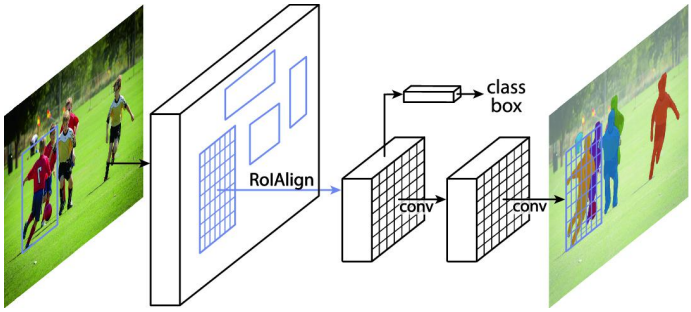


Figure 2.30.: Mask RCNN (He et al., 2017)

Other examples of object detection (including instance segmentation) frameworks include You Only Look Once (YOLO) (Redmon et al., 2016; Redmon and Farhadi, 2017, 2018; Bochkovskiy et al., 2020), Spatial Pyramid Pooling Network (SPPNet) (He et al., 2015), Region-based Fully Convolutional Network (R-FCN) (Dai et al., 2016), and Path Aggregation Network (PANet) (Liu et al., 2018), among others.

Semantic Segmentation

Semantic segmentation networks are end-to-end frameworks that learn pixel-wise labels for an input image. While instance segmentation models are good at localizing objects using a rectangular bounding box and learning their segmentation masks, they are generally suitable for areal objects. For linearly elongated structures, e.g., roads or streams, that could be situated diagonally in an input image, the bounding box detected by an instance segmentation model is basically the whole

image. Semantic segmentation assigns a label for a pixel, regardless of which object instance in the input image it belongs to, making such frameworks suitable for objects of any shape or size.

Fully Convolutional Network (FCN)(Long et al., 2015) is the first fully convolutional network that takes an input image of arbitrary size and predicts dense pixel-wise labels. It is built on top of well known deep classifiers such as VGGNet, AlexNet or GoogLeNet among others. The final classification layer and all the dense layers in the classifiers are discarded. A 1×1 convolution with the feature map size equal to the number of categories in the dataset, e.g., 21 for PASCAL VOC dataset (Everingham et al., 2015), is added. Finally a deconvolution, i.e., backwards convolution or transposed convolution layer is added at the end to upsample and match the dimensions of the input image. The model, illustrated in Figure 2.31, is trained end-to-end and achieved state-of-the-art results in segmentation tasks in 2014.

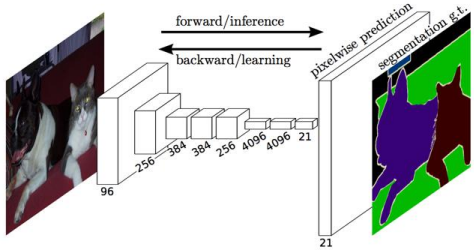
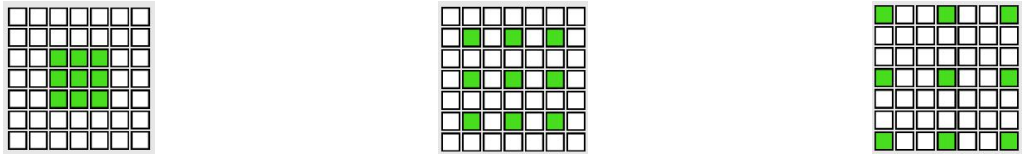


Figure 2.31.: FCN architecture for semantic segmentation (Long et al., 2015)

DCNNs take an input image and extract high level representations through multiple convolutional layers. The striding and max pooling operations decrease the spatial resolution of the features. For semantic segmentation tasks, it is desirable to have high resolution outputs in order to make accurate pixel-level predictions. Chen et al. (2017b) propose removing the max pooling operations in the final layers of DCNNs and using atrous convolutions in order to keep the spatial resolution as high as possible. Atrous convolution, also known as dilated convolution, is based on the 'hole' algorithm (Mallat, 1999) and is similar to the regular convolution operation with an extra parameter called dilation rate. Dilation rate is the spacing between the elements in the convolutional kernel as shown in Figure 2.32. Dilated convolution provides a bigger field of view compared to regular convolution without additional computational cost.



(a) *(b)* *(c)*

Figure 2.32.: Regular (a) and dilated (b & c) convolution with a 3×3 *kernel and dilation rates of* 1, 2 *and* 3

Additionally, DCNNs are invariant to translations in the input and learn abstract representations. While such invariance is beneficial for image classification tasks, it is detrimental to semantic segmentation, specially the final upsampling layers, where the goal is precise pixel-level labeling. To overcome this, Chen et al. (2017b) propose combining the final feature maps in the DCNN with fully connected Conditional Random Field (CRF) (Krähenbühl and Koltun, 2011). CRF is computationally efficient and excels at capturing fine edge details and long range dependencies (in this specific case). Combining it with DCNN boosts pixel-level predictions and refines the outputs

of DL models as shown in Figure 2.33. The whole architecture termed DeepLab is illustrated in Figure 2.34.

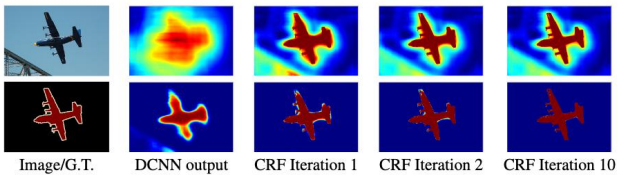


Figure 2.33.: Example prediction by DeepLab with CRF (Chen et al., 2017b)

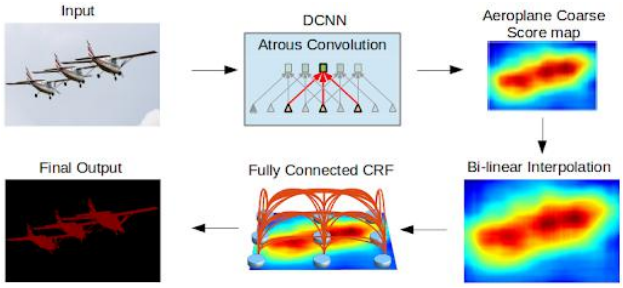


Figure 2.34.: Architecture of DeepLab (Chen et al., 2017b)

Objects of the same category may appear at different scales in images. To account for this variation, an improved version of DeepLab termed DeepLab V2 which makes uses of Atrous Spatial Pyramid Pooling (ASPP) is proposed by Chen et al. (2017b). ASPP is referred to fusing feature maps of multiple atrous convolution with different dilation rates in order to improve prediction accuracy for objects at different scales. It is visualized in Figure 2.35.

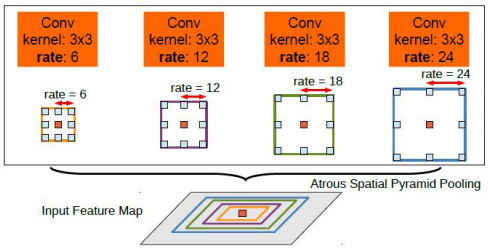


Figure 2.35.: Atrous Spatial Pyramid Pooling used in DeepLab V2 (Chen et al., 2017b)

To improve performance and eliminate the need for CRF post-processing of DCNN outputs, Chen et al. (2017a) propose DeepLab V3. It uses atrous convolution in the final layers of DCNNs followed by ASPP. Additionally, the output of the ASPP module is concatenated with low level image features from the initial convolution layers in order to encode global context. The concatenated features are passed on to to the following layers and finally, predictions are made directly and without the use of CRF. The authors show that including multi-scale contextual information through ASPP and global context help make more precise predictions and lead to improved prediction accuracy compared to previous DeepLab versions. The architecture for DeepLab V3 is shown in Figure 2.36.

Finally, an even more improved version termed DeepLab V3+ is proposed by Chen et al. (2018). It is an extension of DeepLab V3 with a refined decoder module for better prediction along object boundaries. Additionally, the convolutional layers including those in the ASPP module use depthwise separable convolution, as in the Xception network (Chollet, 2017). This results into a faster model with improved performance. The architecture is shown in Figure 2.37.

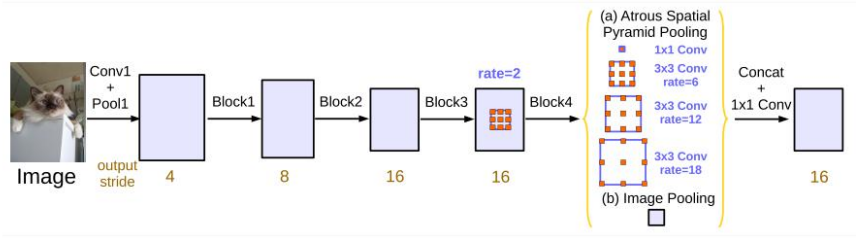


Figure 2.36.: DeepLab V3 (Chen et al., 2017a)

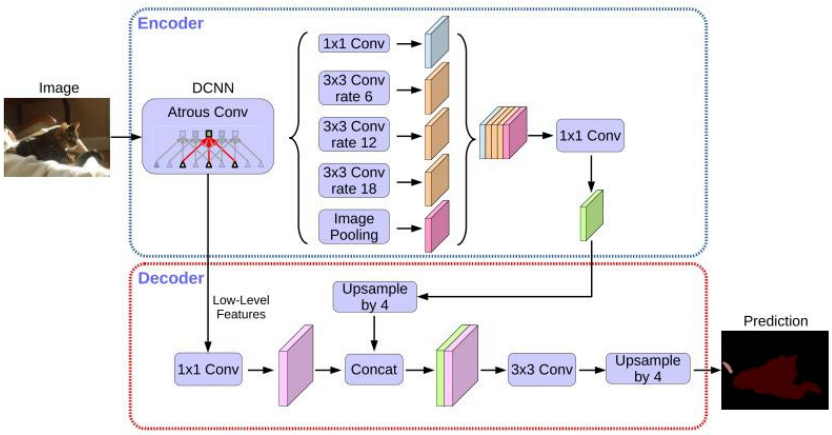


Figure 2.37.: DeepLab V3+ (Chen et al., 2018)

The feature extractor or encoder part of the DeepLab variants could incorporate DCNNs such as ResNet, VGGNet or Xception models originally developed for image classification.

Previous DCNNs for semantic segmentation follow a general pattern of learning image representations by gradually decreasing the spatial resolution. The low-resolution representation is then used in the decoder part of the model to recover high resolution representations by upsampling using different methods such as transposed convolution, bilinear upsampling, and/or non-linear upsampling using pooling indices, among others. In contrast, High Resolution Network (HRNet) (Sun et al., 2019; Wang et al., 2020) maintains the high resolution representation throughout the network and gradually branches out parallel blocks of convolutional layers (eventually 4 parallel branches) with different spatial resolution and regularly exchanging information across the parallel branches. The resulting network is one that outputs semantically richer and spatially more precise representations. The general architecture for HRNet is illustrated in Figure 2.38 and the final outputs from the 4 branches can be used to extend the model for different vision tasks as shown in Figure 2.39.

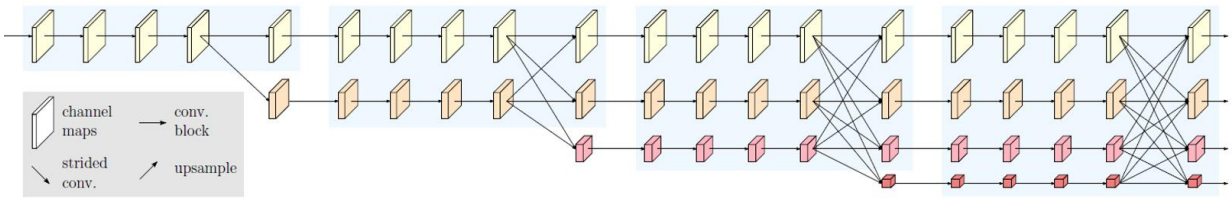


Figure 2.38.: HRNet architecture (Sun et al., 2019; Wang et al., 2020)

Other well known architectures for semantic segmentation include UNet (Ronneberger et al., 2015) and SegNet (Badrinarayanan et al., 2017), among others.

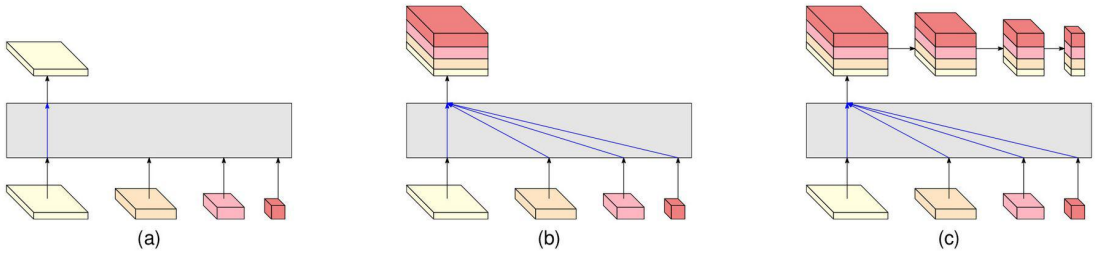


Figure 2.39.: HRNet architecture extended for classification (a), semantic segmentation (b) and feature extraction for object detection tasks (c) (Sun et al., 2019; Wang et al., 2020)

2.4.9. Transfer Learning

DNNs rely on large volumes of labeled data for training in order to generalize and perform well on unseen examples. It is time-consuming and laborious to create labeled data for every task. A solution to this is transfer learning. A DNN trained on another large dataset, such as ImageNet (Russakovsky et al., 2015) is used for a new task with a dataset of small size. The pretrained DNN is either used as an initialization and finetuned or simply used as a fixed feature extractor for another small linear classifier, e.g., SVM (Cortes and Vapnik, 1995). Sharif Razavian et al. (2014) use a DNN by (Sermanet et al., 2014) trained on ILSVRC data (Russakovsky et al., 2015) as a feature extractor for vision tasks such as image classification, object detection and visual instance retrieval, and report superior results compared to the state-of-the-art methods. Yosinski et al. (2014) finetune or retrain the weighted layers of the pretrained model on the new dataset and show that the performance depends on how similar or dissimilar the base dataset (large dataset the model was pretrained on) and the the small target dataset are.

Transfer learning is a dominant pretraining approach and works well when a large annotated dataset for a similar domain to the small dataset is available. However, unsupervised or self supervised pretraining, explained in the following sections, are promising pretraining methods (Yang et al., 2020a) that demonstrate comparable results and are specially useful when large labeled datasets in the similar domain are not available. In this research, SSL pretraining is utilized due to the large volumes of unlabeled DTM data and lack of annotated datasets in the domain of this research.

2.4.10. Unsupervised Learning

In unsupervised learning, there are only input examples and no manual annotations. It can be used to detect and discard redundant and non-essential aspects of the data (Masci et al., 2011). The goal is to find some structure and extract useful features from the data that can help solving another task or answer some questions regarding the data. Unsupervised learning can be grouped into tasks such as clustering, association, and anomaly detection.

In clustering, the model learns to extract features and group similar examples together. Examples include grouping customers by purchasing behavior, grouping images of handwritten digits into their corresponding category, and identifying fake news by analyzing the content and grouping similar news together.

Association is the task of identifying unknown patterns in the data by finding correlations with other examples using known attributes. For example, if person P_1 has read and liked a book B_1, and the task is to find out whether or not they will like another book B_2, it can be solved by looking at other readers, $P_{i\neq1}$, who have read B_1 and B_2 and liked B_1. In general, if $P_{i\neq1}$ has liked B_2 as well, by association, so will P_1. Applications of association tasks include recommending products

one is most likely to buy based on similar previous purchases between buyers and the similarities of products.

Anomaly detection refers to analyzing the data and finding irregularities or data points that are different compared to other examples in the dataset. Applications of anomaly detection include fraud detection in banking, health monitoring in hospitals, and surveillance systems, among others. Anomaly detection can be used to detect outliers in the dataset.

Two other common unsupervised learning methodologies: autoencoders and Generative Adversarial Networks (GANs) are explained in details in the following.

Autoencoders

Autoencoders are a special case of unsupervised learning where the goal is to map an input $\mathbf{x} \in \mathbb{R}^N$ into a lower dimensional latent space $\mathbf{h} \in \mathbb{R}^{N'}$ using a function f with parameters θ and then reconstruct the input from the latent space representation h using another function g with parameters θ'.

$$\begin{aligned} \mathbf{h} &= f_\theta(\mathbf{x}) \\ \mathbf{y} &= g_{\theta'}(\mathbf{h}) \end{aligned} \tag{2.34}$$

The model is trained to minimize the difference between original input $\mathbf{x}$ and the reconstructed $\mathbf{y}$. This unconstrained autoencoder simply learns identity mapping (Goodfellow et al., 2016). To learn useful representations that apply on new examples and generalizes well, other versions of autoencoders are developed, some of which are detailed below.

- **Denoising Autoencoders**: In denoising autoencoders, the input data is corrupted by adding some noise. The model maps the corrupted input to the latent space and learns to reconstruct the original non-corrupted version of the input. Examples of noise added to input data can be removing random pixels from an input image or adding Gaussian noise.
- **Convolutional Autoencoders**: As explained previously, convolutional layers have the advantage of weight sharing and translation invariance compared to fully connected layers. Hence they work better on 2D image data.
- **Stacked Autoencoders**: Multiple layers are stacked together to learn latent space representation of the input first. Similar number of layers are then used to reconstruct the original input from the latent space representation. This can be normal fully connected layers or convolutional layers.

A convolutional autoencoder model proposed by Guo et al. (2017) is illustrated in Figure 2.40 which learns reconstructing and clustering MNIST data (LeCun et al., 1998).

In general, autoencoders are trained to compress inputs to a latent space representation and reconstruct the original input. However, all the variants of autoencoders can be trained such that the encoder part learns hidden representation of the input, but the decoder uses this representation to predict not the original input but other relevant information. Examples are learning to reconstruct color images from grayscale images, reconstructing high resolution images from low resolution counterparts, translating between different data modalities, e.g., aerial photographs to Digital Terrain Models (DTMs) for the same region, and more. Since the reconstructed output is not the original input in these cases, a more appropriate term is the encoder-decoder networks.

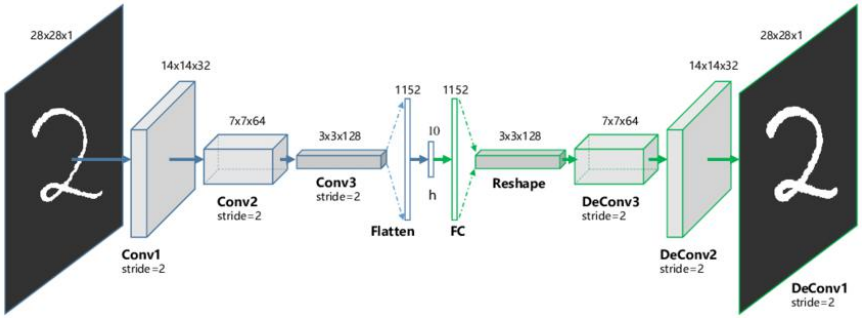


Figure 2.40.: Convolutional autoencoder by Guo et al. (2017)

Generative Adversarial Networks (GANs)

GANs are specific application of generative modeling. Generative models learn internal representation and distribution of the input and as such can be used to sample and generate new examples from the learned distribution. GANs, first introduced by Goodfellow et al. (2014), are generative approaches that frame a task with unlabeled data as a supervised learning problem. A GAN consists of a *generator* and a *discriminator* model. The generator samples random noise from a Gaussian distribution and uses it as an input to generate new plausible examples for the task domain. The discriminator on the other hand takes input examples from the task domain and the new examples generated by the generator, and its task is to determine whether a given input is real (i.e., from the task domain) or fake (i.e., created by the generator). The two models are trained jointly in what is called as the *zero-sum game* in literature. The generator tries to generate realistic examples to fool the discriminator and the discriminator tries to separate real examples from fake. It is called *the zero-sum game* as the reward for either one, the generator or the discriminator, is a loss for the other. In most cases, the training stops when the generator can generate examples that could fool the discriminator at least 50% of the time. The generator can then be used in other applications for generating synthetic data. The general pipeline for GANs is illustrated in Figure 2.41

Mathematically, training GAN models is defined by the following objective function.

$$\mathcal{L}(G,D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_z[\log(1 - D(G(z))] \tag{2.35}$$

Where $D(y)$ denotes the predicted probability by the discriminator, D, that the input, y, is real. $\mathbb{E}_y$ is the expected value over all examples in the dataset. $G(z)$ represents the generated fake example by the generator, G, given a random noise, z, as the input. $\mathbb{E}_z$ is the expected value over all inputs to the generator.

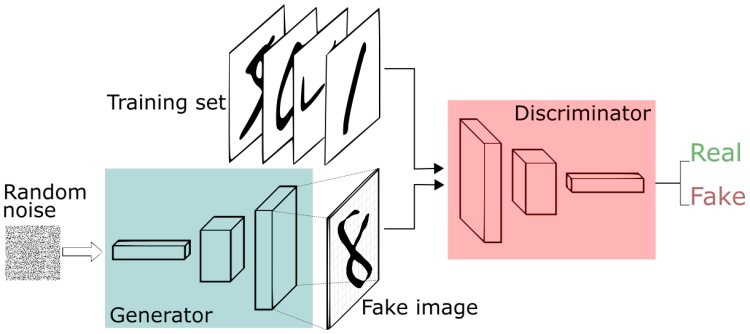


Figure 2.41.: Example pipeline for Generative Adversarial Networks

A specific type of GANs is the conditional GANs (Mirza and Osindero, 2014) in which the generator and/or the discriminator is conditioned on an extra input in addition to the random noise (for the generator) and the fake and real samples (for the discriminator). The condition can be the gender (e.g., male or female) for a GAN model that tries to generate photos, or time (e.g., day or night, winter or summer) for generating images. In conditional GANs, an image y is generated conditioned on the noise vector z, as in the case of general GANs, but it is additionally conditioned on an observed image x. The objective function for conditional GANs is hence modified and shown in Equation 2.36.

$$\mathcal{L}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x,G(x,z)))] + \lambda \mathcal{L}_{L_2}(G) \tag{2.36}$$

Where the first two expressions denote the general GAN objective with the addition of input image x as the condition. The last expression is the $L2$ distance loss, $\mathcal{L}_{L_2}$, and is meant to train the generator to not only fool the discriminator but also produce outputs that are similar to ground truth images.

A well known example of conditional GANs is the Pix2Pix model (Isola et al., 2017). Pix2Pix is a general-purpose image-to-image translation model that learns mapping from an input image to output image. It can be used for tasks such as photo generation, semantic segmentation, image colorization, edge-to-photo translation, sketch-to-photo translation, day-to-night translation of photos, and map-to-aerial photo translation, among others. The random noise vector z in conditional GANs is to avoid producing deterministic outputs when learning a mapping from input x to output y. However, authors of Pix2Pix report that the generator simply learns to ignore the noise. Therefore, the generator input in Pix2Pix is only the input image x and noise is only included in the form of dropout in the network layers. An illustration of the Pix2Pix approach is given in Figure 2.42. Other examples of conditional GANs are CycleGAN (Zhu et al., 2017), and StackGAN (Zhang et al., 2017a), among others.

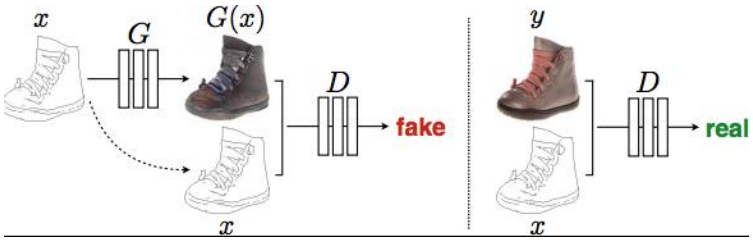


Figure 2.42.: Conditional GANs for edge-to-photo translation (Isola et al., 2017).

GANs have a wide range of applications such as image-to-image translation (Isola et al., 2017; Peters and Brenner, 2020), text-to-image translation (Zhang et al., 2017a; Zhu et al., 2017), video generation (Vondrick et al., 2016), photo blending (Wu et al., 2019a), inpainting (Pathak et al., 2016), image super-resolution (Ledig et al., 2017), and cartoon generation (Jin et al., 2017), among others. In addition to direct applications of GANs in generating realistic examples, they are helpful in improving predictions in supervised tasks. Kumar et al. (2017) use GANs in a semi-supervised learning framework to improve classification accuracy on MNIST (LeCun et al., 1998) and SVHN (Netzer et al., 2011) datasets. A trained discriminator is used as a feature extractor for image classification (Radford et al., 2016), and a trained generator is used to create synthetic annotated datasets for cases where limited training data is available (Shrivastava et al., 2017).

2.4.11. Self Supervised Learning

Self Supervised Learning (SSL) makes use of unlabeled and labeled data. The unlabeled data is first utilized in the first task called *pretext* to train a model that learns useful features from the data. The learned features and the trained model are then leveraged in a second step called the supervised *downstream* task that uses labeled data. The pretext phase uses an architecture composed of a DCNN block (i.e., a deep neural network with multiple convolutional layers) followed by a task specific block (i.e., convolutional and/or fully connected layers) that learns from unlabeled data using implicit supervision signals. Implicit supervision signals refer to the labels that are automatically created from the raw unlabeled data without manual annotation and used to train deep learning models. Examples of implicit supervision signals include rotations applied on input images, parts of input images cropped out, and gray-scale counterparts of color images, among others. Automatically creating such supervision signals or labels from tons of unlabeled images facilitate training deep learning models in the pretext phase. Models trained in this manner can be used in the second phase, i.e., downstream task (Erhan et al., 2010). The downstream phase uses the same DCNN block from the pretext phase initialized with the already learned parameters or weights. It is then followed by a task specific block that learns from annotated examples. Depending on the type of annotations, the task specific block can be classification, segmentation or object detection networks. The general pipeline for SSL is illustrated in Figure 2.43. The inputs to the pretext task are images from a large unlabeled dataset and the labels used for training are the implicit supervision signals automatically created, e.g., rotations applied to input images, or parts of inputs images that are cropped and used as labels. The inputs to the downstream task are images from a manually annotated dataset, and the supervision signals used for training the model are manual annotations indicating either class labels, segmentation masks, or bounding box coordinates. Mathematically, the SSL pretext step is shown Equation 2.37.

$$\hat{y}_p = g(f(x_u)) \tag{2.37}$$

Where x_u and $\hat{y}_p$ denote the unlabeled input data the output in the pretext. f is the DCNN module shown in Figure 2.43 and g indicates task specific layers in the pretext task.

Equation 2.38 represents the downstream step in SSL.

$$\hat{y}_d = h(f^*(x_l)) \tag{2.38}$$

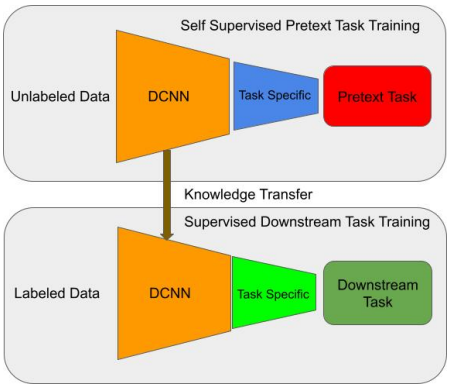


Figure 2.43.: Pipeline for self supervised learning. The DCNN block for both pretext and downstream tasks is the same. The following blocks or layers are task specific. In downstream task, the DCNN block uses the pretrained weights from the pretext task for supervised finetuning.

Where x_l and $\hat{y}_d$ denote the labeled input data and the output in the downstream step. f^* is the DCNN module shown in Figure 2.43 which is pretrained in the pretext step as indicated by the $*$ symbol. h indicates task specific layers in the supervised downstream task.

Autoencoders, GANs, and variants of DCNNs are suitable for the pretext task of learning useful properties and extracting features in the dataset. In this research, an encoder-decoder model and a GAN-based approach are used for the pretext. An encoder-decoder is similar to the stacked autoencoders, with the difference that the output is not a reconstruction of the given input, but an approximation of another superivision signal, e.g., relief rasters for DTMs. An example of SSL pretext task is training a deep neural network that learns image rotations. Unlabeled images are rotated at 90, 270, 180 and 270 degrees and the model is trained to learn the applied rotation. Such a pretraining and incorporating the trained model in supervised vision tasks is reported to prove useful (Gidaris et al., 2018).

Doersch et al. (2015) train a model to take two patches from a 3×3 image grid and predict the used configuration, as shown in Figure 2.44. The trained model learns image representation by predicting such context and boosts performance on supervised vision tasks such as image classification and object detection. The pretext task forces the model to learn feature embeddings for the image patches. The learned feature embeddings for visually similar patches are close to each other in the embedding space while the embeddings for dissimilar patches are far from each other. This helps the model in the downstream task perform better.

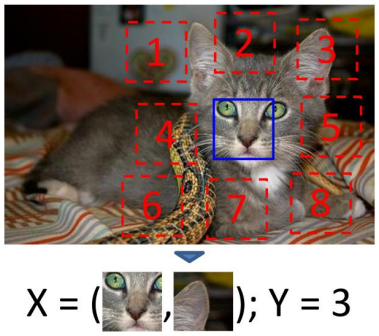


Figure 2.44.: Unsupervised pretraining by learning to predict the context. The DL model is trained to take two patches from the 8 possible configurations and predict the sampled configuration (Doersch et al., 2015)

Other examples of unsupervised representation learning include learning to solve jiggsaw puzzle in images (Noroozi and Favaro, 2016), and image colorization (Zhang et al., 2016c) (Figure 2.45), among others.

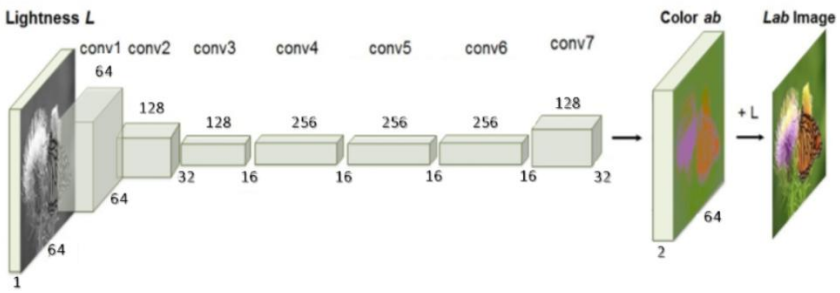


Figure 2.45.: Unsupervised pretraining by image colorization (Zhang et al., 2016c)

Deep learning models trained on the pretext task with unlabeled data can be loaded and finetuned on a downstream task with limited amount of labeled data. This is true for downstream tasks

with limited amounts of annotated data and a similar domain to the pretext task. Examples of downstream tasks are image classification, semantic segmentation, instance segmentation, object detection, and human action recognition, among others.

3. Related Work

This chapter describes related methods in detection of archaeological terrain structures in DTMs. Section 3.1 explains general applications of remote sensing technology in archaeology. Section 3.2 lists common applications of deep learning in remote sensing research in general. Section 3.3 focuses on deep learning applications in point cloud data, specially ALS point clouds and its derived rasters. Section 3.4 includes related work in deep learning used for archaeological research.

3.1. Remote Sensing in Archaeology

Remote sensing is defined as the technique for acquiring information about objects by collecting data through sensors and instruments in a nondestructive and noninvasive manner. This includes data collected by airborne or spaceborne sensors such as LiDAR, RADAR, aerial photography, and satellites or ground based sensors such as ground-penetrating radar and terrestrial laser scanning systems (Campana, 2017). Archaeology is the approach for revealing information about the human history and its environment through field surveys, excavations, and data analysis (Hadjimitsis et al., 2013). Remote sensing techniques and data collected through remote sensing systems are used in archaeology for accomplishing many tasks. These include acquiring information about objects of interest to archaeologists residing on the surface of the earth or buried under ground, monitoring cultural heritage and archaeological sites, and documenting changes through time. Collected remote sensing data are also used to store, analyze and visualize archaeological information with the help of GIS tools. This section outlines different types of remote sensing systems and their applications in archaeology.

The first application of remote sensing in archaeology is the use of aerial photography. Aerial photographs were used for documentation of archaeological sites in Iran (Stolze and Nöldeke, 1882), United Kingdom (Capper, 1907; Bewley, 1999, 2002), Italy (Myers et al., 2002), and Poland (Rączkowski, 2005). During the first World War, aerial photography was used by governments as a source of intelligence. This led to a large storage of aerial photographs that were later utilized for the purpose of archaeological studies (Campana, 2017; Rączkowski, 2001). The second World War generated even more aerial photographs that are utilized by archaeologists as well (Going, 2002). Main techniques for aerial photography and their applications in archaeology are discussed by Crawford and Keiller (1928) and Deuel (1969). Aerial photography is used to survey, analyze and document archaeological sites and landscapes (Campana, 2017; Horne, 2011; Palmer, 2007; Doneus, 2001; Bewley, 2003b).

Aerial photographs are as useful as the visibility of the surface that they capture, and the visibility depends on many natural phenomena, e.g., weather condition. Multispectral and hyperspectral scanners are remote sensing tools that address the shortcomings of aerial photographs. They are sensitive to lights in the infrared and ultra-violet lights which are not visible to the human eye and could acquire more information than the aerial photographs. They are more sensitive to changes in vegetation and temperature and better at discriminating soil moisture (Donoghue, 2001; Shell, 2002; Beck and Cowley, 2011). Multispectral remote sensing, like aerial photography, is used for archaeological research, in addition to other purposes in the industry or the research community. It is used to analyze archaeological crop stress (Moriarty et al., 2019; James et al., 2020), and archaeological remains in Italy (Gennaro et al., 2019). Other applications of multispectral remote

sensing include archaeological land use characterization (Villalon-Turrubiates and Llovera-Torres, 2011), archaeological prospection in alpine alluvial plain (Brivio et al., 2000), and detection of moats around medieval settlements in South India (Rajani and Kasturirangan, 2014), among others.

Multispectral and hyperspectral sensors can be airborne systems mounted on an aircraft and capturing data or they can be spaceborne and mounted on satellites. Satellite imagery is also used for archaeological purposes even if they are originally meant differently, e.g., military purposes. Examples of satellites that were used for military purposes are CORONA, argo, Lanyard and COSMOS (Hadjimitsis et al., 2013; Parcak, 2009). Other examples of satellite images are the KVR images from Russian space program, Landsat by National Aeronautics and Space Administration (NASA) and United States Geological Survey (USGS), IKONOS by the Space Imaging organization, CHRIS Proba by the European Space Agency (ESA), EO-1 Hyperion by NASA, QuickBird, WorldView, and GeoEye-1. Images from the aforementioned satellite systems come in different spatial and temporal resolutions and are utilized in archaeology in addition to other research or commercial purposes. Images from the CORONA and KVR satellites were used for archaeological purposes (Fowler and Fowler, 2005), mapping of geomorphological features (Grosse et al., 2005), and monitoring cultural heritage sites (Kostka, 2002). Landsat images are used for detection of Mayan settlements (Vaughn and Crawford, 2009) and neolithic settlements in Greece (Alexakis et al., 2009), and monitoring archaeological sites (Barlindhaug et al., 2007) and monuments in Cyprus (Hadjimitsis et al., 2009). EO-1 Hyperion images which are of high resolutions are used in detection of archaeological crop marks (Agapiou et al., 2012). QuickBird imagery is also utilized for archaeological prospection (Lasaponara and Masini, 2006b) and identification of archaeological buried remains (Lasaponara and Masini, 2006a).

As explained in Chapter 2, multispectral and hyperspectral scanners are passive remote sensing systems that use natural light from the sun for illumination or radiation. The disadvantage is that such systems cannot be used all the time, e.g., there is no reflected sun energy at night. RADAR is an active remote sensing system that has its own source of energy, can penetrate through cloud, smoke and light precipitation conditions, facilitating measurements at any time (Campana, 2017). RADAR remote sensing, airborne, spaceborne or ground-based, has been utilized in many archaeological projects. Examples include detection of ancient Maya settlements and archaeological structures within the settlements (Adams, 1980; Pope and Dahlin, 1989; Garrison et al., 2011), analysis and monitoring of archaeological sites (Tapete et al., 2013; Stewart et al., 2013; Moore et al., 2006), and detection of the Great Wall in north-western China (Xinqiao et al., 1997), among others.

Terrestrial laser scanning is another active remote sensing technique that is useful for many research fields such as architecture, earth science and archaeology (Forte and Campana, 2016). Terrestrial laser scanning data and methods are used for documentation of archaeological monuments (Neubauer et al., 2005; Lerma et al., 2010) and landscapes (Forte et al., 2005). They are also used for recording, monitoring, and preservation of archaeological heritage (Grussenmeyer et al., 2012; Lercari, 2019; Castagnetti et al., 2012; Wei et al., 2010). Other example applications of terrestrial laser scanning in archaeological research include damage detection in historical buildings (Armesto-González et al., 2010), virtual reconstruction of destroyed historical structures (Bitelli et al., 2017; Lindstaedt et al., 2011), and monitoring of wet-preserved archaeological wood (Lobb et al., 2010), among others.

Finally ALS or airborne LiDAR is a remote sensing technique that measures elevations of the ground surface relative to a reference point. It produces high resolution digital terrain models for large landscapes that are otherwise unattainable by photogrammetry (Campana, 2017). LiDAR has also had the most significant impact in archaeological remote sensing (Bewley, 2003a). It is used in archaeological research for mapping archaeological landscapes (Chase et al., 2011),

detecting archaeological features under woodland canopies (Devereux et al., 2005), identifying archaeological sites and landscapes (Masini et al., 2011), detection and monitoring of cultural heritage (Trier and Zortea, 2012; Risbøl et al., 2015), forest management (Roman et al., 2017), discovery of amazonian villages (Iriarte et al., 2020), revealing hidden Mayan structures under forest canopy in Guatemala (Canuto et al., 2018), uncovering previously unknown Great War sites in Belgium (Gheyle et al., 2018), uncovering arhchaeological landscapes at Angkor (Evans et al., 2013), and the discovery of medieval fortified settlements in southern Italy (Masini et al., 2018), among others. LiDAR rasters, different types of which were explained in Chapter 2, are utilized for better interpretability and visualization of archaeological landscapes, features and artifacts. Yokoyama et al. (2002) and Doneus (2013) use openness for visualization and interpretation of DTM data. Kokalj et al. (2013) address archaeological interpretation of LiDAR data with different relief models including analytical hillshading, trend removal, slope, SVF and elevation differentiation, among others. Kokalj and Hesse (2017) give an extensive list of LiDAR rasters and discuss the types of archaeological structures each of them are suitable for. This thesis specifically focuses on using LiDAR data: LD, SLRM, slope, SVF, POS and NEG, in addition to DTM for the purpose of self supervised detection of archaeological monuments.

3.2. Deep Learning in Remote Sensing

Deep learning techniques are used in remote sensing applications for many tasks including land cover classification, semantic and instance segmentation, scene classification, image registration, point cloud registration, change detection, data fusion and more. This section lists example applications of deep learning models in remote sensing relevant to this research.

Land Use and Land Cover Classification

Land Use and Land Cover Classification (LULC) is a common task in remote sensing. It refers to image classification or semantic segmentation tasks carried out for remote sensing data such as multispectral satellite images. DCNNs have dominated the LULC task in remote sensing. Helber et al. (2019) create EuroSAT, a benchmark dataset for LULC tasks using deep learning techniques. They use well-known DCNNs such as ResNet and GoogleNet and a shallow three-layer CNN to classify Sentinel-2 satellite images from the EuroSAT dataset into one of ten LULC classes including industrial buildings, residential buildings, annual crop, permanent crop, river, sea and lake, herbaceous vegetation, highway, pasture and forest. Zhang et al. (2018) combine the ideas for residual learning, atrous convolution and spatial pyramid pooling to create a UNet-like (Ronneberger et al., 2015) model for urban land use and land cover classification based on high spatial resolution satellite imagery. Kussul et al. (2017) compare DCNNs with traditional fully connected models and random forest, and show the superiority of DCNNs for classification of land and crop types such as maize, soybeans, wheat, sunflower and sugar beet using Landsat-8 and Sentinel-1A satellite images. MarsNet (Palafox et al., 2017) is another example application of multiple DCNN classifiers used in parallel for automated detection of geological landforms on Mars. Other examples of DCNNs applied in LULC include the works of Zhang et al. (2019a), Tracewski et al. (2017), and Marcos et al. (2018), among others.

Scene Classification

Image scene classification is another import task in the remote sensing community which has profited from the advances in deep learning. Cheng et al. (2017a) create a benchmark dataset called "NWPU-RESISC45" for this task. The dataset contains more than 30 thousand images covering

45 scene categories. The authors report classification results using different classification models such as AlexNet, VGGNet and GoogLeNet. Another benchmark dataset named Aerial Image Dataset (AID) is created by Xia et al. (2017). It is a dataset with more than 10 thousand annotated aerial scene images. Hu et al. (2015) use transfer learning and finetune classifiers trained on ImageNet data in order to classify scences in high resolution remote sensing images. Nogueira et al. (2017) also explore the effect of finetuning deep classifiers pretrained on ImageNet dataset on remote sensing scene classification task. They use GoogLeNet, VGGNet, and AlexNet models, among others, with randomly initialized weights and also ImageNet weights to perform scene classification on datasets such as UCMerced (Yang and Newsam, 2010), RS19 (Xia et al., 2010), and Brazilian Coffee Scenes (Penatti et al., 2015). Lu et al. (2017) use unsupervised representation learning to improve scene classification accuracy. They first use deconvolution networks to extract a set of feature maps for each image using multiple filters and minimize the reconstruction error between the input image and the output feature maps. The learned feature maps are then aggregated using spatial pyramid pooling and passed on to an SVM classifier. Other examples of DL models in remote sensing scene classification include research works by Cheng et al. (2017b), Cheng et al. (2018), Li et al. (2017a), and Zhang et al. (2019d), among others.

Object Detection

Object detection in remote sensing refers to identifying and localizing objects in a given remotely sensed data such as satellite images (Cheng and Han, 2016). Deep learning models have shown great success in the remote sensing object detection tasks as well. A benchmark dataset for optical remote sensing object detection named Detection in Optical Remote sensing (DIOR) is created by Li et al. (2020a). Deng et al. (2018) use concatenated ReLUs and inception modules to create a feature extractor. They then create a multi-scale object proposal network to generate region proposals which are consequently merged with the extracted features and fed to the final object detection network. Their proposed multi-scale object detection framework is evaluated on datasets such as NWPU-VHR-10 (Cheng et al., 2014), Aerial-Vehicle (Liu and Mattyus, 2015) and aircraft dataset (Zhang et al., 2016a). The authors report better detection results compared to other object detection models, e.g., Faster RCNN (Ren et al., 2016) and YOLO (Redmon et al., 2016; Redmon and Farhadi, 2017), evaluated on the same datasets.

Objects in optical remote sensing images appear at different rotation angles, making them hard for DL models to predict. To alleviate this problem, Cheng et al. (2016) propose a model termed Rotation Invariant Convolutional Neural Network (RICNN) for object detection. Their model uses existing CNN architectures, but is additionally trained to produce similar feature representations for multiple rotations for the same input. Their object detection model is evaluated on on the NWPU-VHR-10 (Cheng et al., 2014) dataset and achieves better results compared to other detection frameworks trained under the same conditions. Other examples of DL models for object detection in remote sensing data include the works of Li et al. (2017b), Li et al. (2018b), Diao et al. (2016), Ding et al. (2018), and Zhang et al. (2019c), among others. Extensive overviews of object detection in remote sensing are written by Li et al. (2020a), Zhang et al. (2016b) and Cheng and Han (2016).

Semantic Segmentation

Semantic segmentation of remote sensing images is also an important task. Similar to other major research topics in remote sensing, deep learning techniques have found their way into segmentation of remote sensing imagery as well. Examples of benchmark datasets in remote sensing image segmentation tasks include the Vaihingen and Potsdam datasets by ISPRS (Rottensteiner et al.,

2012), Zurich dataset (Volpi and Ferrari, 2015), and EvLab-SS dataset (Zhang et al., 2017b). Acquiring large annotated datasets for segmentation tasks in remote sensing imagery is costly and time-consuming. The previously mentioned datasets have either sacrificed high spatial resolution images for higher number of class labels or included more categories by a trade-off in the image resolution. Kemker et al. (2018) created a dataset called RIT-18 which contains high resolution multispectral images and includes higher number of categories, i.e., 18 different classes including road marking, trees, buildings, vehicles, woode panels, vegetations, water, and rocks, among others. It is difficult to use the dataset for training DL models due to imbalanced class distribution. To alleviate this problem, Kemker et al. (2018) first use the Digital Imaging and Remote Sensing Image Generation (DIRSIG) modeling software to generate large volumes of synthetic annotated multispectral images. The synthetic dataset is then used for training a DCNN that learns pixel-wise classification or semantic segmentation. The trained model is finetuned on the RIT-18 dataset. The authors use adapted versions of SharpMask (Pinheiro et al., 2016) and RefineNet (Lin et al., 2017a) as the DCNN for semantic segmentation and report superior results compared to traditional methods, e.g., SVM and Multi Layer Perceptrons (MLPs), trained on the RIT-18 dataset.

Kampffmeyer et al. (2016) use DCNNs for semantic segmentation in the ISPRS Vaihingen dataset (Rottensteiner et al., 2012) but incorporate uncertainty maps in the network using Monte Carlo dropout (Gal and Ghahramani, 2016) in order to improve segmentation accuracy. Rustowicz et al. (2019) create a dataset for semanatic segmentation of crop types in in Ghana and South Sudan. The authors use a DNN called 2D UNet + CLSTM, which is a hybrid of UNet (Ronneberger et al., 2015) and Convolutional Long Short Term Memory (LSTM) (Shi et al., 2015), and achieves great accuracy scores on their own dataset and outperforms previous leading methods on the dataset from Munich, Germany (Rußwurm and Körner, 2018). Other examples of DL models for semantic segmentation in remote sensing images include the works of Marmanis et al. (2016), Xu et al. (2018b), Audebert et al. (2017), Wurm et al. (2019), Pan et al. (2018), and Diakogiannis et al. (2020), among others.

3.3. Deep Learning in Point Clouds and Digital Terrain Models

Similar to to their success on 2D image data, deep learning techniques solve various problems using 3D point cloud data and 2D rasters derived from ALS point clouds. Major tasks in deep learning for 3D point cloud data include 3D shape classification, object detection and point cloud segmentation (Guo et al., 2020).

Based on how the input data are processed, shape classification techniques are categorized into three different methods. The first method is multi-view based in which the unstructured point cloud is projected into 2D images on which the model is trained. Examples of such methods include MVCNN (Su et al., 2015), MHBN (Yu et al., 2018), and View-GCN (Wei et al., 2020), among others. The second method is based on 3D volumetric representations of the point cloud data. A point cloud is first voxelized into 3D grids and then fed to 3D CNNs for shape classification. Examples of such methods include VoxNet (Maturana and Scherer, 2015), OctNet (Riegler et al., 2017), and PointGrid (Le and Duan, 2018), among others. The last method is called point-based where deep learning models directly consume raw point clouds for feature extraction and prediction tasks. Different models are applied using the point-based approaches including MLP, CNN and graph-based models. Examples of MLP-based approaches include PointNet (Qi et al., 2017a), PointNet++ (Qi et al., 2017b), and PointASNL (Yan et al., 2020), among others. Examples of CNN-based approaches include RS-CNN (Liu et al., 2019), PointConv (Wu et al., 2019b), GeoConv (Lan et al., 2019), PointCNN (Li et al., 2018a), and KPConv (Thomas et al., 2019), among others.

Finally examples of graph-based methods include DGCNN (Wang et al., 2019), KCNet (Shen et al., 2018), G3D (Dominguez et al., 2018), and PointGCN (Zhang and Rabbat, 2018), among others.

The second major task in deep learning for 3D point clouds is object detection. Deep learning models in this task are divided into two categories based on the workflow. The first category is based on region proposals. First, regions with the possibility of containing objects are proposed and then features are extracted for each region in order to predict a class label. Examples of region proposal-based models include PointRCNN (Shi et al., 2019), F-PointNets (Qi et al., 2018), and the works of Vora et al. (2020), Lang et al. (2019), Yang et al. (2019b), and Xu et al. (2018a), among others. The second category of deep learning models for object detection in 3D point clouds is called the single shot method. In the single shot methods, a single stage is used to predict class probabilities and bounding boxes for objects in the input. Examples of such methods include VeloFCN (Li et al., 2016), 3D-FCN (Li, 2017), 3DSSD (Yang et al., 2020b), and LaserNet (Meyer et al., 2019a), among others.

Finally, the third major task is semantic segmentation of point clouds using deep learning. The goal is to predict a label representing the semantic meaning of each point. Similar to shape classification, there are multiple categories in this approach as well based on how the input is processed. Examples of multi-view methods include the works of Lawin et al. (2017), Boulch et al. (2017), and Tatarchenko et al. (2018), among others. Example methods using volumetric representations include SEGCloud (Tchapmi et al., 2017), and ScanComplete (Dai et al., 2018), among others. Finally, examples of point-based methods for this task include the works of Engelmann et al. (2018), Zhang et al. (2019e), and Hu et al. (2020), among others.

Other example applications of deep learning on 3D point cloud include photo-realistic point cloud rending with Conditional GANs (Peters and Brenner, 2020), point cloud to image translation Milz et al. (2019), semantic segmentation with multi view outlier detection (Peters et al., 2020), automatic generation of point cloud data for training (Peters and Brenner, 2019), and the works of Shu et al. (2019), Yang et al. (2019a), and Sauder and Sievers (2019), among others.

Methods explained above on general 3D point clouds are also applicable to ALS point clouds which are the initial data source for the rasters used in this research. An adapted version of PointNet (Qi et al., 2017a,b) is used for semantic labeling of ALS point cloud (Winiwarter et al., 2019). Yang et al. (2017) first transform each point in the 3D space into a 2D image using the geometric and full-waveform features of its surrounding points and then feed the image to a CNN for classification. Qin et al. (2019) propose VPNet which uses a combination of volumetric and point based representation for semantic labeling of ALS point clouds. Li et al. (2020b) use a model for this task (i.e., semantic labeling of ALS point clouds) taking into consideration three distinct properties of ALS point clouds including geometry of instances, variations in scale for different categories and the discrepancies in the elevations. Zhang et al. (2021) propose DPCC-Net which uses DCNNs in combination with k-means clustering for unsupervised 3D terrain scene clustering. The DCNN is used for extracting features and the k-means algorithm is used for clustering, the results of which are used as pseudo labels for training the DCNN. Politz et al. (2020) use height distributions within a grid for a rasterized point cloud as inputs to a DCNN for classification. ALS data are also used alongside dense image matching for semantic segmentation with DCNNs (Politz and Sester, 2018, 2019). Other examples of deep learning for ALS point clouds include the works of Zhao et al. (2020), Zhao et al. (2018), Zhang et al. (2018), Soilán et al. (2019), and Politz et al. (2018), among others.

ALS point clouds are usually used to create DTMs first and then DCNNs are trained on the DTMs or other DTM derivatives. Torres et al. (2018) use DTMs to identify mountain summits with deep learning. Lee (2019) exploit deep learning and DTM data for detection of craters on

Mars. Marmanis et al. (2015) use DTMs and deep learning to filter points above the ground. Heo et al. (2020) train a CNN model on DTM data to search for high solar energy regions. Another example of deep learning application using DTMs is identification of river defences (Wood et al., 2021). Researchers also derive other rasters from DTMs and then use them to train deep learning models. Examples include classification of ancient Maya settlements with DCNNs trained on relief visualization rasters (Somrak et al., 2020), automated mapping of cultural heritage in SLRM rasters (Trier et al., 2021), mapping topographic features of mining related valley fills in slope rasters (Maxwell et al., 2020b), mapping of industrial heritage using relief rasters such as SLRM, POS and NEG rasters (Gallwey et al., 2019), among others.

3.4. Deep Learning in Archaeology

Traditional machine learning and computer vision tools and methods have been applied in cultural heritage and archaeological research for many tasks. van der Maaten et al. (2006) use content-based retrieval system for classification of historical glass and edge detection algorithms for medieval coin classification. Meyer et al. (2019b) use Object Based Image Analysis (OBIA) techniques on DTM data for automated detection of field monuments such as ridge and furrow areas, burial mounds, and mote-and-bailey castles in Westphalia, Germany. Kersten and Lindstaedt (2012) automatically reconstruct 3D models of cultural heritage objects and archaeological structures from image data. Pavlidis et al. (2007) make use of 3D acquisition and digitization techniques to document and display cultural heritage artworks. Such digitization can also help in monitoring cultural heritage objects as small deformations and cracks can easily be detected (Beraldin et al., 1999). Gomez-Lahoz and Gonzalez-Aguilera (2009) use the Computer Aided Design (CAD) software to create 3D models of archaeological sites. Computer vision photogrammetry techniques are applied for recording underwater archaeological sites in low visibility environments (Van Damme, 2015). Yaman (2019) use multinomial logistic regression for classification of arrowheads made of iron and bronze found in an excavation project in Karamattepe, Turkey. Makridis and Daras (2013) automatically classify archaeological pottery sherds using classical computer vision and image processing techniques such as Kirsch edge detection (Kirsch, 1971), Local Binary Patterns (LBP) (Ojala et al., 1994), Bag of Words (BoW) (Wallach, 2006), K-Nearest Neighbor (KNN) (Aha et al., 1991), and SVM (Chang and Lin, 2011). Klassen et al. (2018) use a semi supervised approach with multiple linear regression and graph-based models to predict the chronology of medieval archaeological sites in Angkor, Cambodia. Orengo and Garcia-Molsosa (2019) automatically detect potsherds in high resolution drone imagery using Random Forest (RF). Traviglia and Torsello (2017) detect patterns in archaeological landscapes using remote sensing images and classical filtering method, Gabor filters (Feichtinger and Strohmer, 2002), thresholding and and feature extraction algorithms.

Classical methods for archaeology explained previously require discriminative features to be extracted based on informed decisions by an expert in the field. The extracted features are then processed by automated machine learning algorithms for classification or detection of interesting patterns. Selection and extraction of such features manually is an expensive task and costs a lot of time and money specially with large volumes of data. Similar to their applications in the other domains explained in previous sections, deep learning techniques are applicable to research fields in archaeology and cultural heritage management as well. Researchers have leveraged DL methods for detection and classification of archaeological objects in natural images, aerial and satellite imagery, and laser scanning data, among others. Brenner et al. (2018) use CNNs to automatically detect bomb craters in World War II aerial images. Bundzel et al. (2020) use DL models to detect areas of ancient construction activity and find remnants of ancient Maya building. The authors train UNet and Mask RCNN models on LiDAR data and report

satisfactory detection rates, specially for objects of medium size. Trier et al. (2021) automatically map cultural heritage sites in Norway training Fast RCNN on LiDAR data. They report detection results for structures such as grave mounds, pitfall traps, and charcoal kilns. Other examples of DL applications in archaeological research include the works of Soroush et al. (2020), Lambers et al. (2019), Maxwell et al. (2020b), Maxwell et al. (2020a), and Gallwey et al. (2019), among others.

In general, there is a lack of benchmark datasets for archaeological tasks. Researchers in this domain create their own labeled datasets which are not made publicly available. These datasets are small in comparison to datasets in other domains, e.g., natural images. Therefore, in an attempt to improve the performance, transfer learning from other domains are used. For example Trier et al. (2019) use models pretrained on ImageNet, and Verschoof-van der Vaart and Lambers (2019) use models pretrained on PASCAL VOC (Everingham et al., 2015) to finetune them on their own archaeological datasets, respectively. The goal of this research is to utilize the recent promising method, i.e., Self Supervised Learning (SSL). SSL pretext uses unlabeled data for the same domain, i.e., DTM data to pretrain deep learning models. The pretrained models are then finetuned for suprvised downstream tasks on annotated datasets to detect archaeological monuments and historical terrain structures.

4. Datasets

This chapter includes details of the datasets used in this research. Section 4.1 discusses the DTM data collected from the Lower Saxony State in Germany. Section 4.2 describes archaeological monuments and man-made terrain structures in the Harz including two multi-class datasets: one with areal structures and one with linear structures, and one single-category dataset. Data preprocessing for deep learning tasks are explained in Section 4.3

4.1. Digital Terrain Model and Relief Visualization Dataset

The focus of this thesis is automated detection of man-made terrain structures related to historical mining and archaeology in the Harz mountains. The Harz mountains are located in Lower Saxony, Germany with a maximum altitude of 1141 meters. The region is home to ore and mineral deposits found and processed over thousands of years (Segers-Glocke et al., 2000; Bartels and Klappauf, 2012; Malek, 2017). The ore found in this region was the main source for minting coins in the Middle Ages. The historical town of Goslar, which was the residence for German kings and emperors in the 11^{th} century, is also located in this region. Water drainage and technical innovations in the 15^{th} century led to the possibility of smelting silver with the help of lead ores and added to the significance of the region. Goslar, the ore mines of Rammelsberg, and Upper Harz Water Management Systems that are listed as UNESCO World Heritage Sites are also located in this region.

Annotated datasets for a small region of the Harz mountains are available as detailed in the following sections. However, unlabeled DTM data for the whole state of Lower Saxony is leveraged in the pretext phase of SSL, i.e., the main methodology in this dissertation. The DTM data is created from LiDAR using TIN (explained in Chapter 2) with linear interpolation. It has a resolution of 0.5 meters per pixel and encompasses a total area of $47\,000\,\text{km}^2$. The DTM contains elevation values of up to 1000 meters and visualizing the whole region as such would just yield a grayscale image mostly filled with dark color. For better visualization, the shaded relief for the DTM is illustrated in Figure 4.1.

As explained in Chapter 2, various relief visualizations can be calculated from the DTMs which normalize the pixel values within a fixed range and are transformed into a more visually perceivable format. Examples of such relief rasters, as explained in Chapter 2, include Simple Local Relief Model (SLRM), Local Dominance (LD), Sky View Factor (SVF), Positive Openness (POS), Negative Openness (NEG) and slope among others. While the relief rasters are originally created for visualization, analysis and presentation (in the form of maps) of structures on the terrain, they are used in this thesis for pretraining deep learning models in the Self Supervised Learning (SSL) pretext phase. The RVT software (Kokalj and Somrak, 2019) is used to calculate the previously mentioned relief rasters for the unlabeled DTM data. Examples of DTM regions and the corresponding relief rasters are visualized in Figures 2.3-2.15 in Chapter 2.

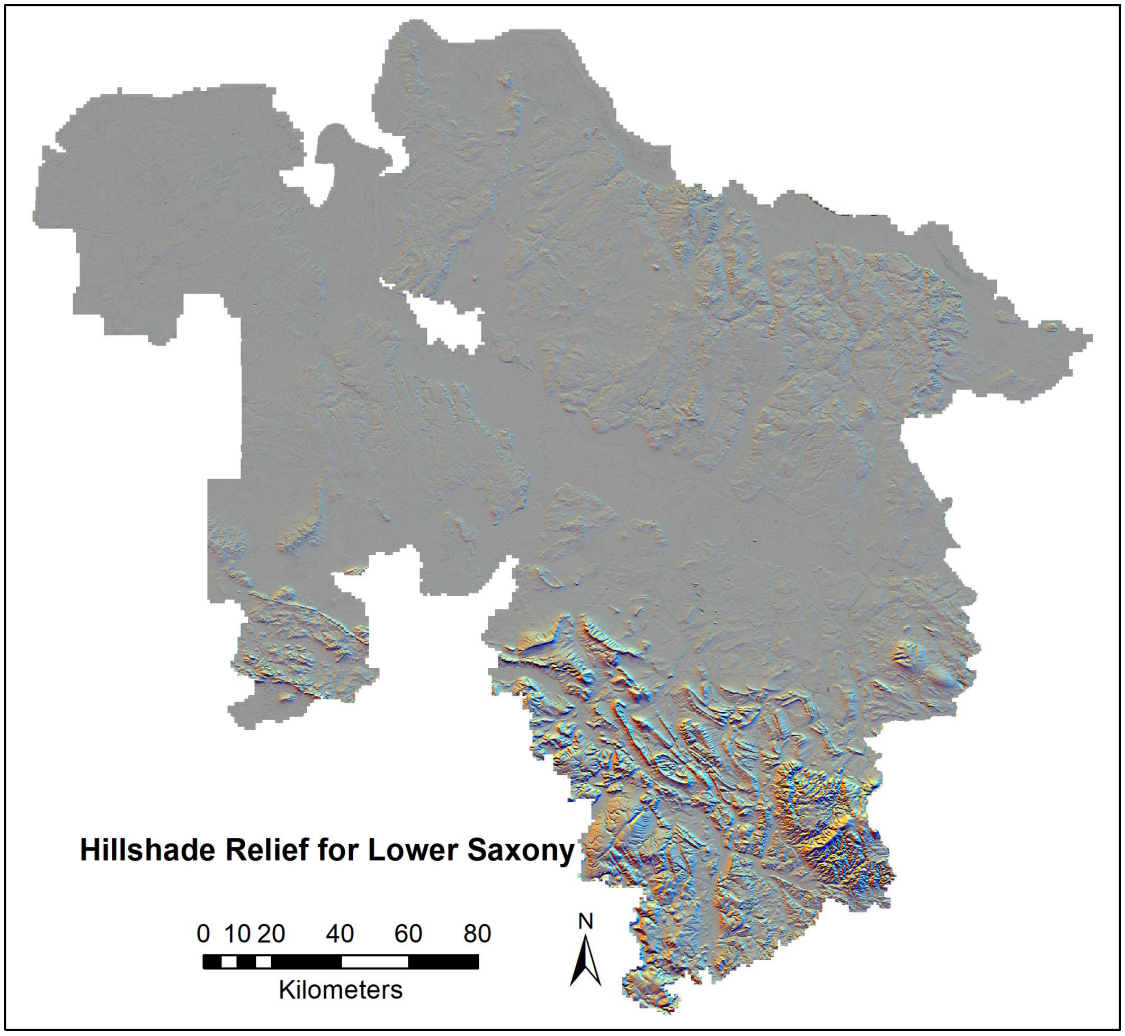


Figure 4.1.: Hillshade relief visualization for DTM data from Lower Saxony.

4.2. Archaeological Monuments in the Harz

This section describes the archaeological monuments and man-made terrain structures studied in this research. There are two datasets with objects of areal and linearly elongated shapes. The first dataset includes bomb craters, charcoal kilns, burial mounds and mining holes. The second dataset contains examples of ditches, paths, roads, and hollow ways. Another dataset with annotated examples of stone quarries is also investigated. The stone quarries also have closed areal shapes, but due to the huge difference in their sizes compared to the objects in the first dataset, they are investigated separately. Annotations for all the datasets are created in ArcGIS by archaeologists or with their guidance using the locations of already known monuments and structures. Details of each dataset are given in the following sections. However, since some of structures studied in this research may not be known to everyone, a brief explanation of some of them are given here.

- **Charcoal kilns:** Charcoal kilns are structures built for creating coal from wood which were used as fuel for smelting ores and production of metals (Deforce et al., 2021). An example of charcoal kiln is shown in Figure 4.2a.

- **Bomb craters:** bomb craters are a type of crater or structure formed by the bombs dropped from air craft during World War II. An example of a bomb crater is shown in Figure 4.2b.

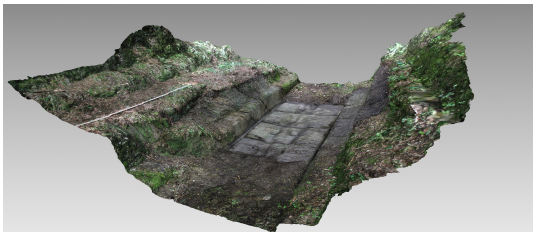
- **Burial mounds:** Accumulated earth and and stones over a grave are called burial mounds. An example of burial mound is shown in Figure 4.2c.
- **Mining holes:** Mining holes or sinkholes are a kind of depression on the terrain caused by mining. An example of mining hole is shown in Figure 4.2d.
- **Hollow ways:** Hollow ways are ancient paths and tracks that are sunken due to humans and animals walking on them over time, and they now exist in the form of linear depressions on the terrain. An example of hollow ways is shown in Figure 4.2e.
- **Ditches:** Ditches are structures that were created to channel water for irrigation and water supply management. An example of ditches is shown in Figure 4.2f.
- **Stone quarries:** Stone quarries are mining structures from which rocks and minerals are extracted. An example of stone quarries is shown in Figure 4.2g.

(a) *Example charcoal kiln. Taken from Figure 5 in (Di Fazio et al., 2010)*

(b) *Example bomb crater. Taken from Figure 10 in (Passmore et al., 2014)*

(c) *Examples of burial mounds. Taken from Figure 1 in (Dayaratne, 2012)*

(d) *Example mining hole. Image by Katharina Malek, NLD.*



(e) *3D-rendering of a hollow way. Image credit: Georg Drechsler and Katharina Malek, NLD.*

(f) *Example of a ditch from the Upper Harz Water management System. Credit: Katharina Malek NLD*

(g) *Communion quarry near the mine of Rammelsberg, Goslar (Credit: Torsten Schröpfer, NLD).*

Figure 4.2.: Example images for the structures studied in this research.

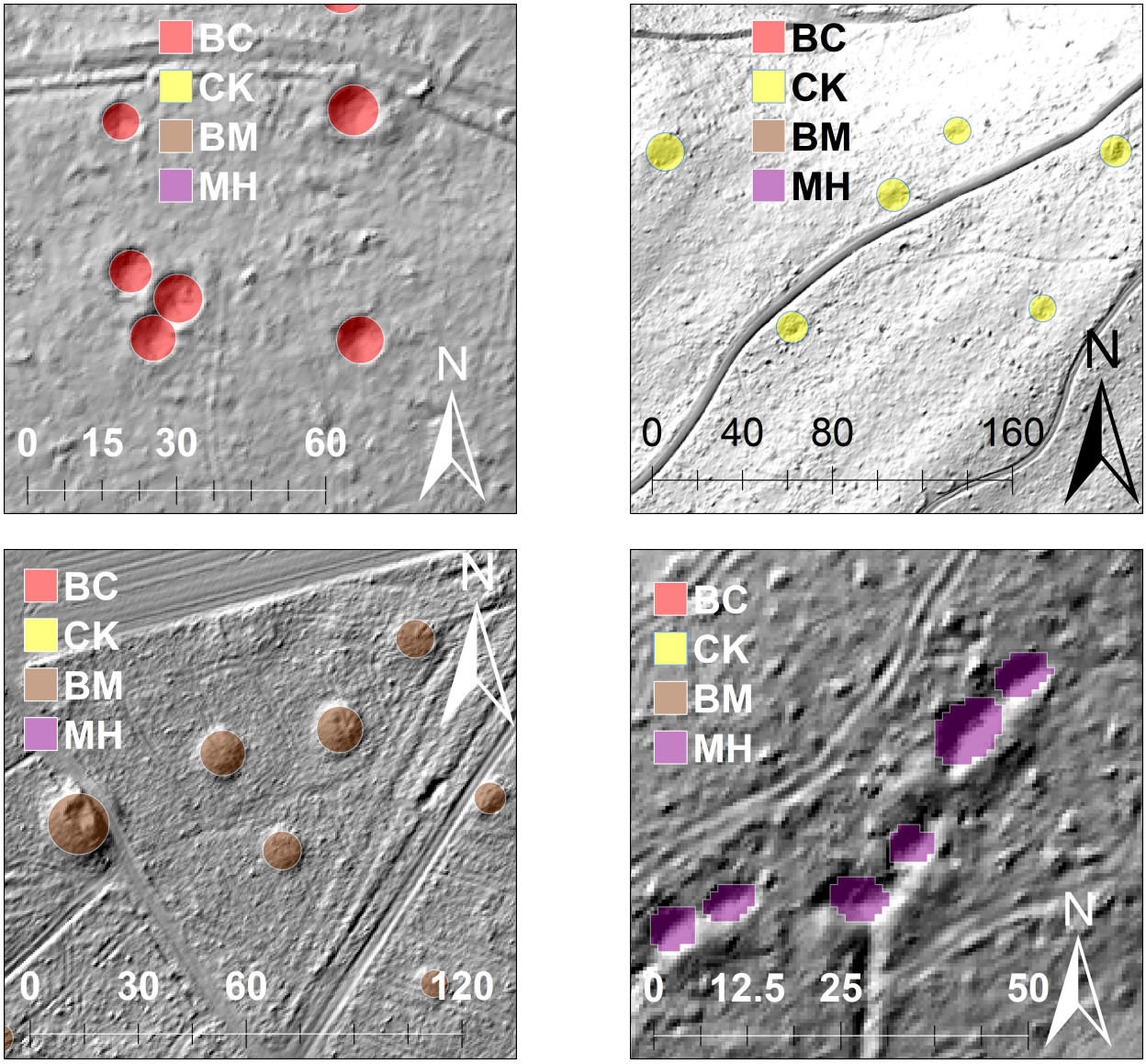


Figure 4.3.: Example annotations for the Harz Areal Dataset.

4.2.1. Areal Dataset

This dataset contains annotated examples of 4 kinds of areal structures including bomb craters, charcoal kilns, burial mounds and mining holes. They have areal shapes, and some of them are manually annotated as such. However, annotating the exact shape of all the instances is a time-consuming task. Knowing the central pixel location of each instance, the ArcGIS software is used to create circular polygons as annotations for some of the 4 structures. This results into missing parts of each object instance in some cases and covering parts of the background pixels with the object in other cases. Examples of annotations for these structures are shown in Figure 4.3. Statistics for the structures in this dataset are given in Table 4.1.

	No. examples	Min. diameter	Avg. diameter	Max. diameter
Bomb Craters	617	1.3 m	7.4 m	38 m
Charcoal Kilns	2543	6.3 m	15.3 m	24.4 m
Burial Mounds	1410	4.5 m	14.8 m	37.7 m
Mining Holes	2986	1.2 m	8 m	63 m

Table 4.1.: Statistics for Harz Areal Dataset.

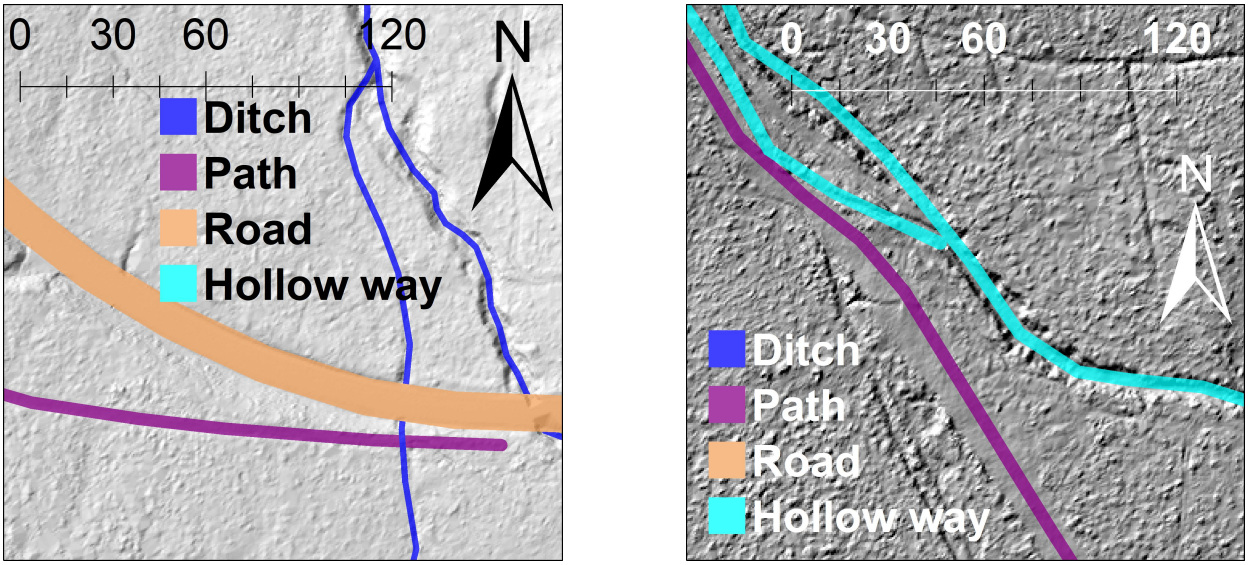


Figure 4.4.: Example annotations for the Harz Linear Dataset.

4.2.2. Linear Dataset

The second dataset in this research includes linearly elongated structures such as ditches, paths, roads and hollow ways. They are initially annotated as line features in ArcGIS. To approximate the actual width or each structure and also to be able to apply deep learning techniques (especially semantic and instance segmentation), polygon features are calculated from the line features using the ArcGIS Buffer operation. The buffer distance on each side of the line features are set based on the type of the structure. For ditches, a buffer distance of 1 meter on each side is used, resulting in polygon features of width 2 meters for each ditch. A buffer distance of 2 meters is used for paths and hollow ways, resulting in polygon features of width 4 meters for these two structures. Finally for the road features, different buffer distances are applied based on the type of each road. A buffer distance of 4 meters is used for county roads. For state roads and federal roads, the applied buffer distances are 6 and 8 meters, respectively. Example annotations for this dataset are shown in Figure 4.4. Statistics for the structures in this dataset are given in Table 4.2.

	No. segments	Min. area	Avg. area	Total area	Max. area
Ditches	837	17.57 m^2	2030 m^2	1.69 km^2	0.63 km^2
Paths	1183	44.80 m^2	4469 m^2	5.28 km^2	0.06 km^2
Roads	174	440.11 m^2	17 606 m^2	3.06 km^2	0.02 km^2
Hollow ways	756	28.41 m^2	54 479 m^2	1.39 km^2	0.05 km^2

Table 4.2.: Statistics for Harz Linear Dataset.

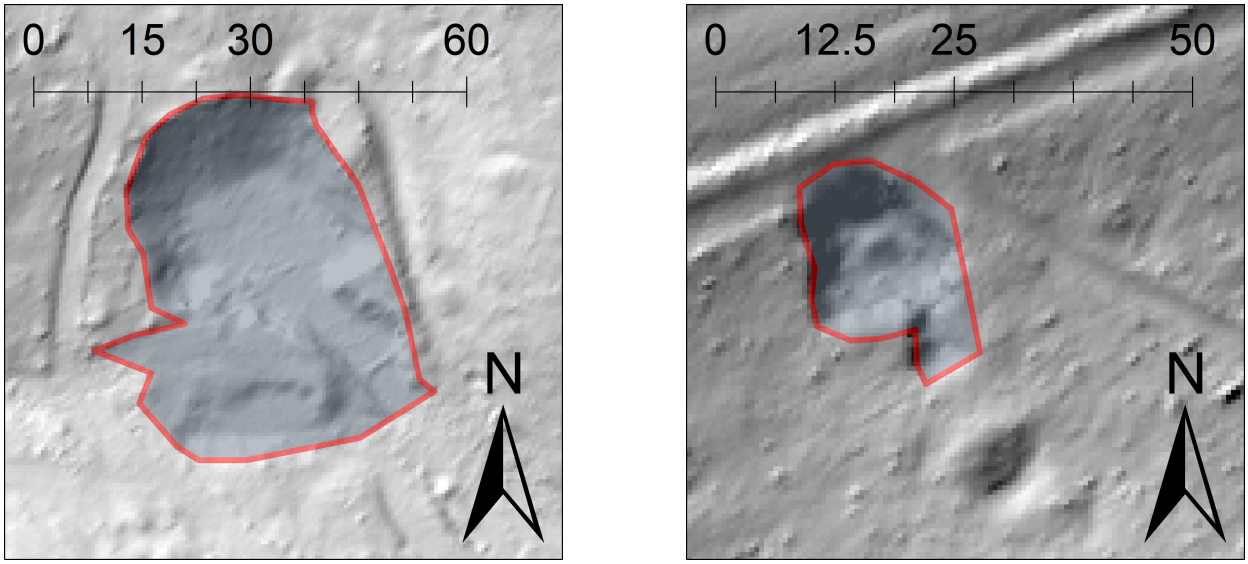


Figure 4.5.: Example annotations for the Stone Quarries (SQ) Dataset.

4.2.3. Stone Quarries Dataset

The final dataset in this research includes annotations of only one kind of structure: historical stone quarries. They are manually annotated as polygons of different sizes and shapes. Example annotations for stone quarries are shown in Figure 4.5 and the statistics are given in Table 4.3. As observed in the statistics shown in Tables 4.1 and 4.3, examples of stone quarries are very large compared to the structures in the areal dataset explained in Section 4.2.1 previously. Therefore, this dataset is used separately even though the stone quarries also have a closed shape and can in principle be included in the areal dataset.

No. examples	Min. area	Avg. area	Max. area
3082	20.9 m^2	6494 m^2	3 km^2

Table 4.3.: Statistics for Stone Quarries Dataset.

4.3. Data Preparation for Deep Learning Models

This section describes the data processing steps to prepare training, validation and test sets for deep learning experiments conducted in this research. Details of unlabeled data processing steps for the SSL pretext task are given in Section 4.3.1. In the second phase of the SSL approach followed in this research, three downstream tasks are carried out: classification, instance segmentation and semantic segmentation, and the processing details are described in Sections 4.3.2, 4.3.3 and 4.3.4, respectively.

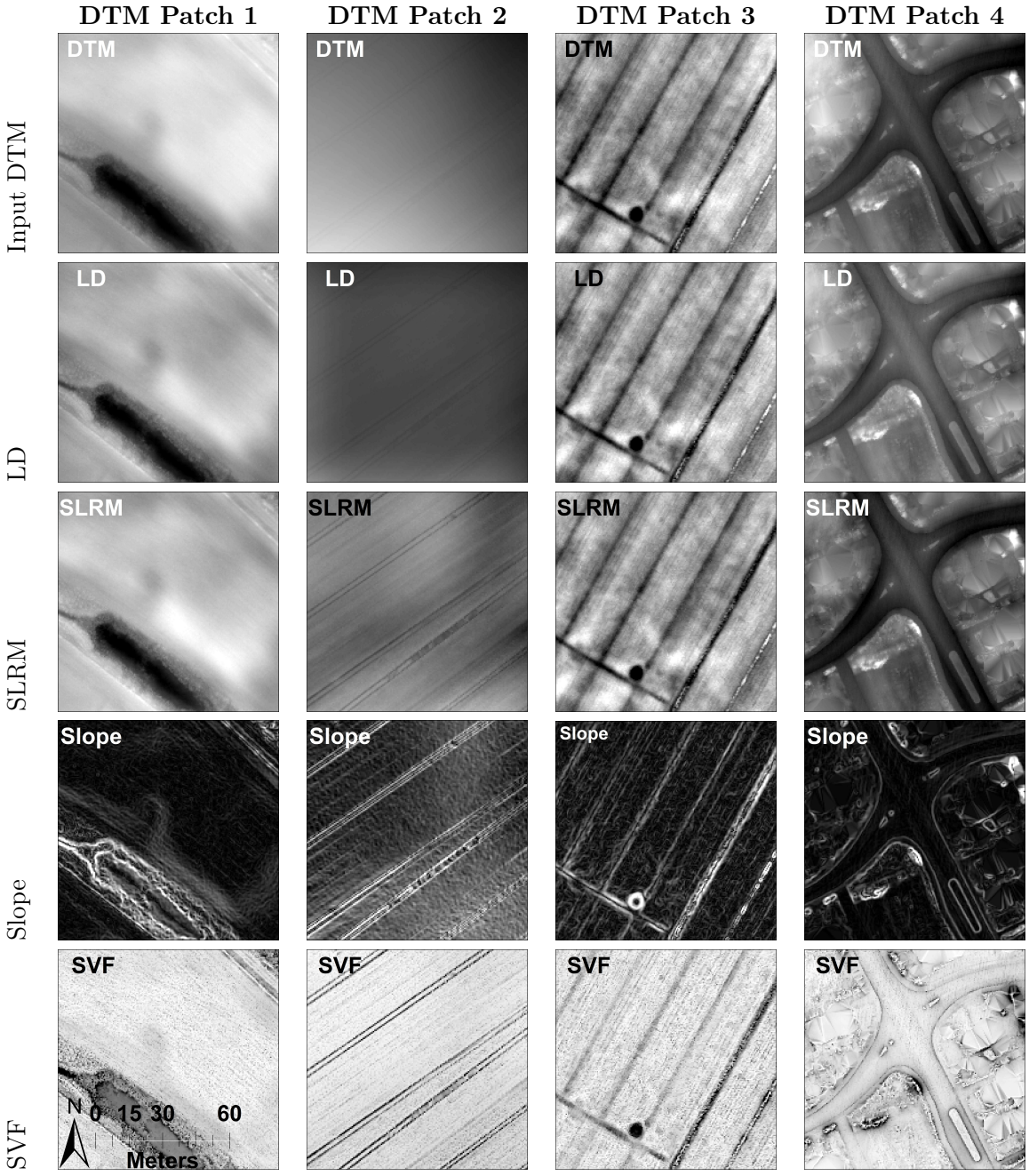
4.3.1. Data Processing for Self Supervised Learning Pretext

Unlabeled DTM data for Lower Saxony is leveraged in the first phase in SSL, namely the pretext phase. To create training data, 200 000 random DTM patches of size 224×224 pixels are cropped from the region. For each of the DTM patches, relief rasters such as LD, SLRM, slope, SVF, POS and NEG, explained previously, are calculated using the RVT software (Kokalj and Somrak, 2019).

Pixels in the relief rasters are calculated relative to neighboring pixels within a search radius of 64 pixels each for 16 directions. For more details regarding the search radius and direction parameters, please refer to Section 2.3.4 or the user manual for RVT (Kokalj and Somrak, 2019). Illustrations for four selected DTM patches with their corresponding relief rasters from the prepared dataset are given in Table 4.4. Input DTM patches and the relief rasters are scaled locally to have pixels in the range 0 and 1 using Equation 4.1

$$X = \frac{X - \mathrm{MIN}(X)}{\mathrm{MAX}(X) - \mathrm{MIN}(X)} \tag{4.1}$$

Where X denotes a raster patch and MIN and MAX indicate the minimum and maximum operators.



Continued on next page

Table 4.4 – continued from previous page

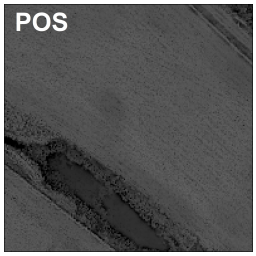
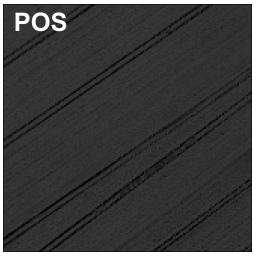
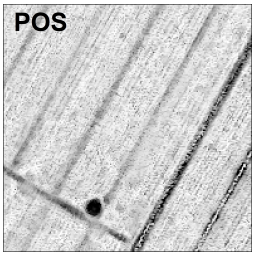
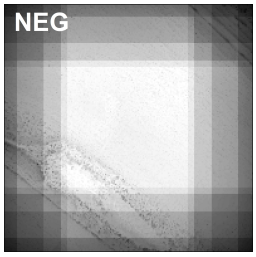
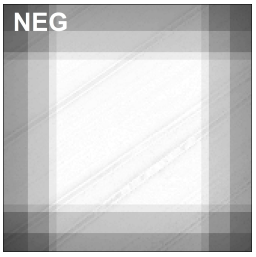
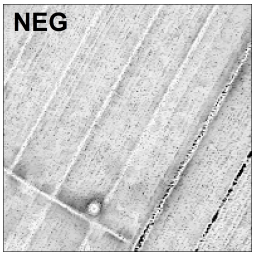
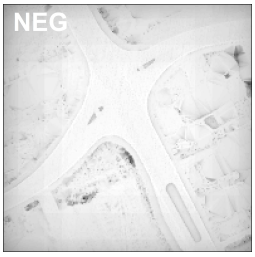
	DTM Patch 1	DTM Patch 2	DTM Patch 3	DTM Patch 4
POS				
NEG				

Table 4.4.: Four training examples for SSL pretext. First row shows DTM inputs, the remaining 6 rows show relief rasters used as labels.

4.3.2. Data Processing for Classification

Classification models take input DTM patches and produce a probability for the input belonging to a category from a predefined set of categories. To train a classifier with inputs of $n \times n$ pixels, an $n \times n$ DTM patch centered at each annotated object is cropped from the region and assigned the corresponding object's label. For this experiment, classifiers with input sizes $n \in \{32,64,96,128,224\}$ are trained. An example DTM input for an instance of burial mounds is shown in Figure 4.6.

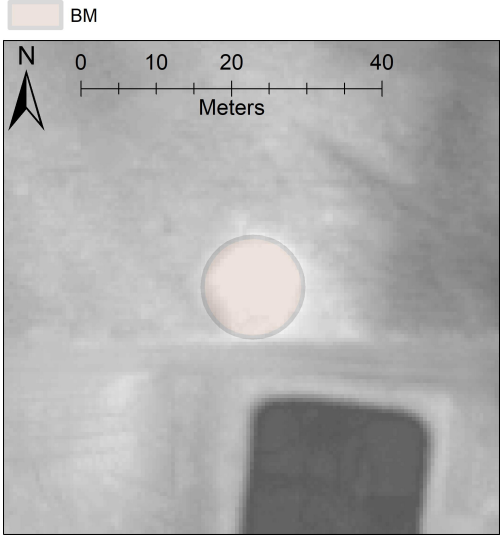
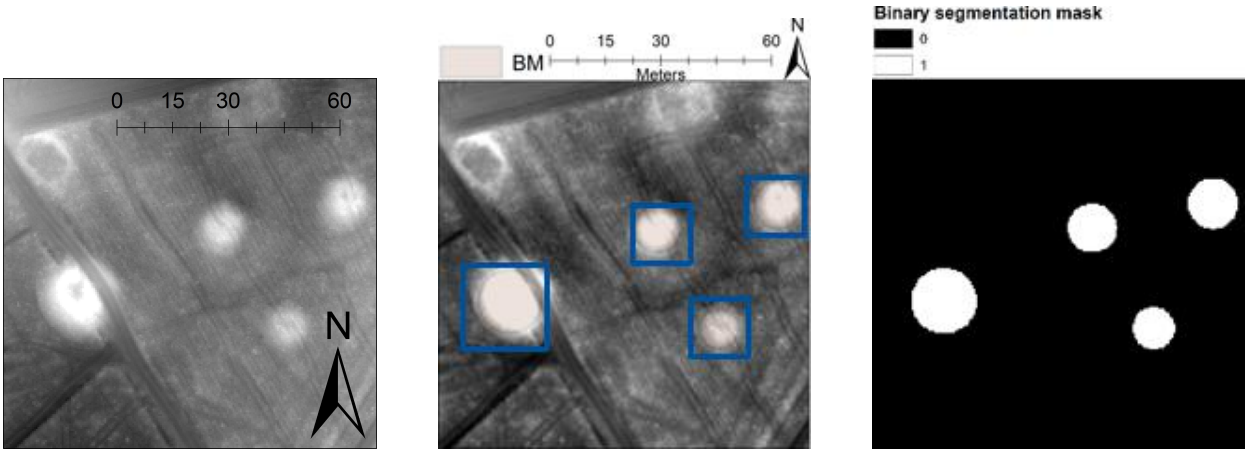


Figure 4.6.: Example input and label for classification with input size of 128×128 *pixels.*

4.3.3. Data Processing for Instance Segmentation

Instance segmentation models predict bounding box coordinates, binary segmentation masks and class labels for object instances in a given input DTM and therefore need training data prepared as such. For this approach, random DTM patches are cropped for each object making sure the

object is included in the patch, but not necessarily centered. Thus, multiple training examples can be created for each object and this helps increase the size of training data. Example DTM input and annotations for instance segmentation model is shown in Figure 4.7.

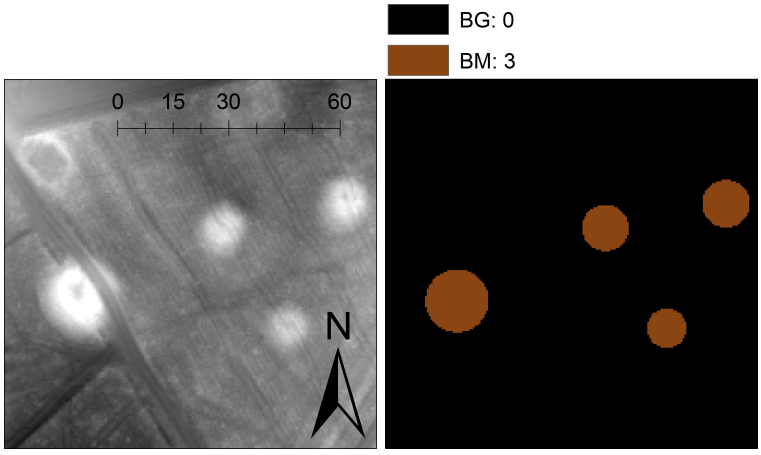


(a) DTM input. *(b) Bounding box.* *(c) Segmentation mask.*

Figure 4.7.: Training example for instance segmentation.

4.3.4. Data Processing for Semantic Segmentation

In semantic segmentation, the predicted output for every input is the pixel-wise class labels. Similar to instance segmentation, training data for this approach is also created by random crops of DTM around each object making sure the object is within the patch and not necessarily centered. Example DTM input and annotations for semantic segmentation model is shown in Figure 4.8.



(a) DTM input. *(b) semantic segmentation label.*

Figure 4.8.: Training example for semantic segmentation.

The same datasets are used for classification, instance segmentation and semantic segmentation. The difference is in how the inputs and labels are prepared for each task as explained previously.

5. Methodology

This chapter discusses the deep learning technique technique, i.e., Self Supervised Learning (SSL), used for recognition of archaeological terrain structures in DTMs created from LiDAR or ALS data. SSL, as explained in Chapter 2, is composed of two steps. The first step is called the pretext, in which unlabeled data are used to train deep learning models for representation learning and feature embedding extraction. The learned and extracted feature embeddings are close to each other in the embedding space for similar inputs. The knowledge from the pretext step is then transferred to the second step called the downstream. In the downstream tasks, models are initialized with the pretrained weights from the pretext step and then finetuned on annotated datasets for different supervised tasks, i.e., classification, object detection and semantic segmentation. In the pretext phase in this research, models are trained on unlabeled DTM data (and automatically calculated relief rasters are used as implicit supervision signals) to generate relief visualization rasters. The pretrained models are then customized for supervised downstream tasks to detect archaeological monuments. The DTM patches alone can be used for the pretext with random rotations or cropping applied to each DTM patch as the supervision signal. However, the relief rasters are used as the supervision signals in this task since they are shown to be effective in training deep learning models (Kazimi et al., 2019a). The goal is to train deep learning models that learn generating such rasters from DTMs and by doing so help improve detection performance in supervised downstream tasks. Section 5.1 describes the SSL pretext approaches exploited in this research and Section 5.2 gives details of the supervised downstream tasks for detection of archaeological monuments and man-made terrain structures. The general pipeline for the SSL technique used in this research is shown in Figure 5.1

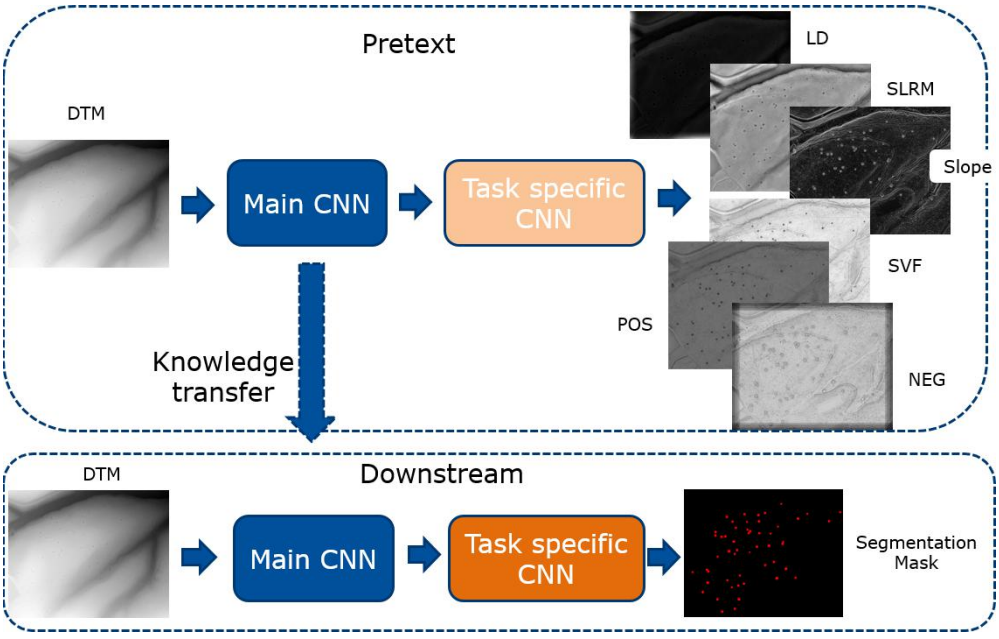


Figure 5.1.: Training pipeline in Self Supervised Learning.

5.1. Pretext Methods

DTM data represent the elevation of points on a terrain with respect to a reference height using continuous values. The values can range from 0 (in some cases it could be negative as well) and go up to thousands or more depending on the region. This is different to natural images which have a fixed gray-scale range, e.g., 0 to 255. There are methods to convert raw DTM data into other raster formats, referred to as relief visualizations, each of which has a certain fixed range of values and are suitable for interpreting different structures. Among them are LD, SLRM, slope, SVF, POS, and NEG, as explained in Chapter 2. While such relief visualization rasters are mathematically calculated from the DTM, in this research, deep learning models are trained in the SSL pretext phase to automatically generate them given an input DTM. Two different methods are explored as the pretext tasks in this research. The first method follows an encoder-decoder approach to generate the corresponding relief visualization rasters for a DTM patch. Hence, it is termed as the Relief Visualization Network (RVNet). The encoder-decoder model learns to map input DTM patches to their corresponding relief rasters. Since the goal is to use the pretrained encoder-decoder model to improve detection performance in supervised downstream tasks, it is desired that the model does not learn the mapping (from input DTM patches to relief rasters) perfectly. It should only approximately generate outputs that resemble the original relief rasters. Thus, the model is forced to learn important properties of the data (Goodfellow et al., 2016). This constraint can be introduced by adding random noise to the input DTM patches or incorporating noise to the model architecture, e.g., in the form of dropout layers (Isola et al., 2017). A well-known family of architectures for this is Generative Adversarial Networks (GANs). Therefore, the second pretext method in this research is based on GANs and used for the same purpose, i.e., generating relief visualizations. It is therefore termed as the Relief Visualization GAN (RVGan). Both methods use the unlabeled DTM data and the automatically calculated relief visualization rasters explained in Chapter 4 for training. Detailed explanations of the two approaches are given in the following sections.

5.1.1. Relief Visualization Network (RVNet)

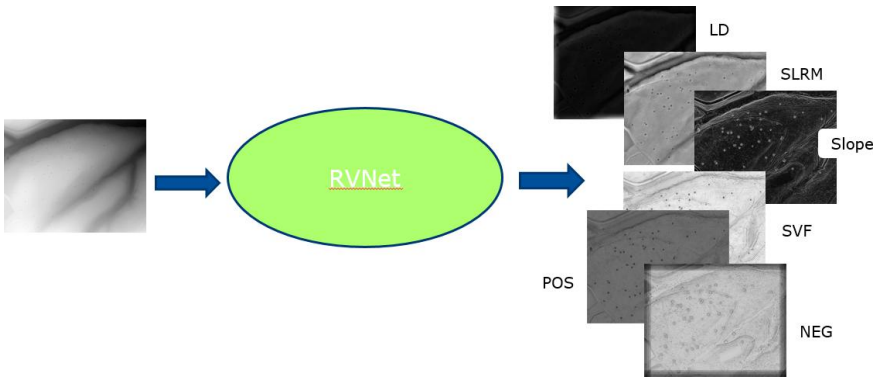


Figure 5.2.: Relief Visualization Net (RVNet)

As shown in Figure 5.1, the pretext phase is composed of a main CNN module and a task specific module. The downstream phase uses its own task specific module which is different to that of the pretext, but it uses the same architecture in the main CNN module as in the pretext phase making the knowledge transfer possible. Many different deep learning architectures can be used in the pretext phase, as long its main CNN module is compatible to be incorporated in the downstream phase. In this section, the well-known HRNet model (Sun et al., 2019; Wang et al., 2020) is selected. This choice is made based on HRNet's performance in the supervised classification task compared

to another well-known deep learning architecture (He et al., 2016). It is also compared to ResNet as the backbone in Mask RCNN for instance segmentation. Finally, HRNet for semantic segmentation is compared to DeepLabV3+ (Chen et al., 2018) with the ResNet backbone. In general, HRNet performs better (as reported in the following sections) and hence it is selected as the main model in this research. The last layer in the pretext phase is composed of convolutional layers that output a feature map with 6 channels each representing LD, SLRM, slope, SVF, POS and NEG rasters, respectively. Thus, the first SSL method is called the Relief Visualization Network (RVNet) and illustrated in Figure 5.2.

5.1.2. Relief Visualization GAN (RVGan)

The second pretext method in this research is based on conditional GANs, specifically the Pix2Pix model (Isola et al., 2017). A generator model is trained to take input DTMs and produce the corresponding relief visualization rasters. A discriminator model is simultaneously trained to take input DTMs and the corresponding relief rasters, either the already calculated ones or those produced by the generator, and its task is to tell them apart. The architecture for this approach, called Relief Visualization GAN (RVGan) is illustrated in Figure 5.3. While the discriminator model is similar to that of the Pix2Pix (Isola et al., 2017), the generator model is selected to be HRNet (Sun et al., 2019; Wang et al., 2020) similar to the RVNet model.

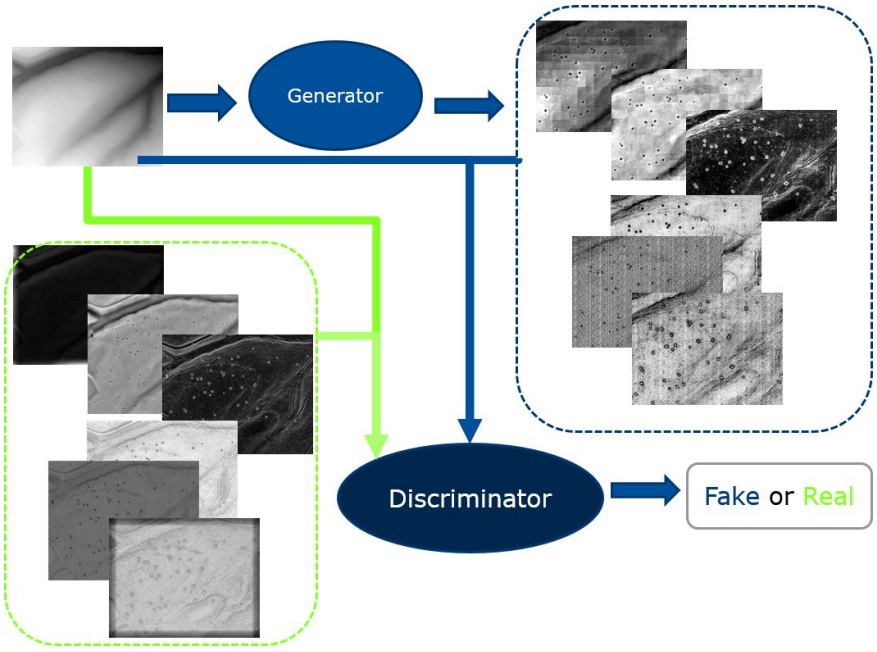


Figure 5.3.: Relief Visualization GAN (RVGan)

5.2. Downstream Methods

The second phase in SSL is called downstream. In this phase, the lower main CNN module as shown in Figure 5.1 is initialized with the learned weights of the main CNN module from the pretext phase and the task specific module is customized to supervised tasks and finetuned on annotated datasets. Three different tasks are explored in this phase for detection of archaeological monuments and man-made terrain structures in the Harz region. These are classification, instance segmentation and semantic segmentation explained in the following sections.

5.2.1. Classification of Archaeological Monuments and Terrain Structures

Classification is the task of assigning a label from a list of predefined categories to a given input. The classification models in this research are trained to take input DTMs and predict the probability of archaeological monuments and terrain structures in the given input region. The pipeline for classification of archaeological structures including Charcoal Kilns (CK), Burial Mounds (BM), and Mining Holes (MH) and man-made structures such as Bomb Craters (BC) in addition to Background class (BG) is shown in Figure 5.4.

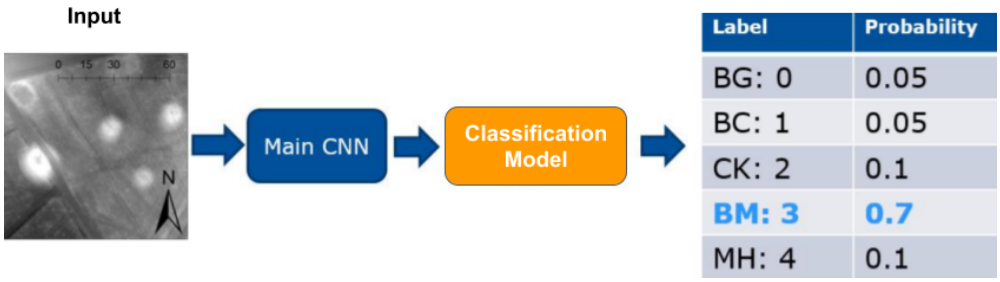


Figure 5.4.: Pipeline for classification of archaeological monuments and man-made terrain structures. The input is a DTM patch. The label is an integer indicating which type of object it contains.

The main CNN module in the pipeline for classification is the HRNet model which is also used in the SSL pretext methods (RVNet and RVGan). The pipeline can be trained with random weight initialization, but to take advantage of the SSL pretraining, the main CNN module can be initialized with the pretrained parameters of RVNet or RVGan in order to train faster and produce better results. As explained previously the ResNet (He et al., 2016) model is also trained for classification on the same dataset and compared to the HRNet.

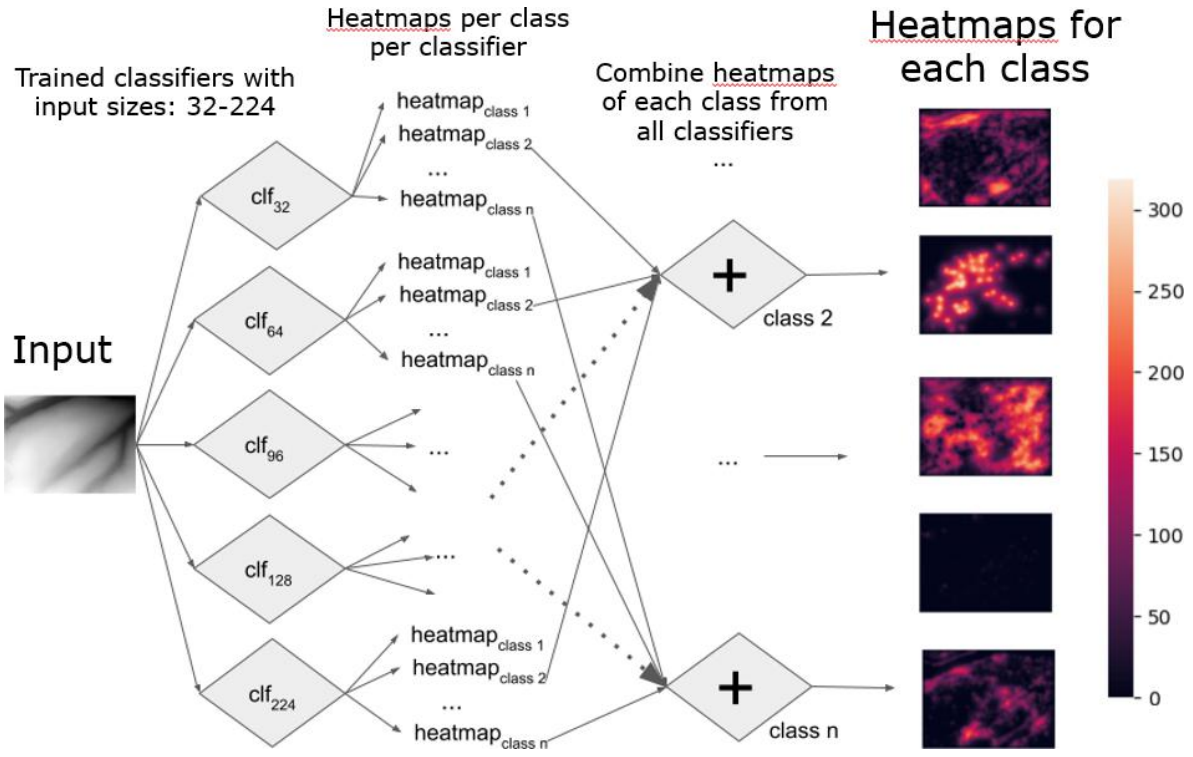


Figure 5.5.: Pipeline for inference. clf_h denotes trained classifier with input size $h \times h$

Objects may appear at different sizes and scales. In order to account for these variations, classifiers with different input sizes are trained. Additionally, in order to make use of the trained classifiers, inspired by the works of Palafox et al. (2017), they are used together to make predictions on large regions following the methodology shown in Figure 5.5 and Equation 5.1.

$$\mathrm{HM}_n = \sum_{c=i}^{C} \mathrm{HM}_{c,n} \text{ for } n = 1,2,...,N \tag{5.1}$$

Where $\mathrm{HM}_{c,n}$ denotes the predicted heatmap for class n by classifier c using Algorithm 1, N and C denote the number of classes and number of classifiers (each trained with different input sizes previously mentioned), respectively, and the final heatmaps by the classifiers are summed for each class and optionally masked for values greater than a desired threshold. The final heatmap for each class represent the presence or absence of the corresponding class in the DTM.

Algorithm 1 *The procedure for generating heatmaps using a classification model*

Require:
$\mathrm{classifier}_{H\times W}$: classifier model trained with input size $H \times W$
$\mathrm{DTM}_{\mathrm{input}}$: Input DTM of size $P \times Q$
N: Number of classes the the classifier model is trained to detect.
s: Stride size for the sliding window approach.

procedure GetHeatmaps($\mathrm{classifier}_{H\times W}$, $\mathrm{DTM}_{\mathrm{input}}$, N, s)
  HM $\leftarrow$ Zeros($N \times P \times Q$) ▷ a matrix of zeros of shape ($N \times P \times Q$)
  for each item i in range(0, $P - H$, step $= s$) **do**
    for each item j in range(0, $Q - W$, step $= s$) **do**
      currentPatch $\leftarrow \mathrm{DTM}_{\mathrm{input}}[i : i + H, j : j + W]$
      $n \leftarrow \mathrm{classifier}_{H\times W}$(currentPatch) ▷ Class label predicted for the current patch
      $\mathrm{HM}[n, i : i + H, j : j + W]$ += ones($H \times W$)
    end for
  end for
  return HM ▷ a heat map of size P x Q for each class
end procedure

5.2.2. Instance Segmentation of Archaeological Monuments and Terrain Structures

The classification approach explained in the previous section produces heatmaps with a rough indication of regions containing structures of interest. They are also very slow at inference scanning large DTM regions cropping DTM patches to make predictions. The cropping of DTM patches need to be overlapping for a reasonable heatmap at the end. To alleviate such issues and make faster predictions, instance segmentation approaches are studied on the same dataset and for the same task, i.e., automated detection of archaeological monuments and man-made structures in DTMs. Instance segmentation models produce class labels, bounding box coordinates and binary segmentation masks for every instance of objects in a given input DTM as shown in Figure 5.6. Thus, while scanning a large region using the instance segmentation models, it is not necessary to crop overlapping patches as the instances of objects in each patch are detected individually. This leads to fast prediction for larger regions. For instance segmentation, the well-known Mask RCNN (He et al., 2017) framework is used. It is first trained using ResNet as the feature extractor backbone as in the original version. Then, the ResNet backbone is replaced with HRNet and the model is trained on the same dataset. Similar to the classification approach, the main CNN module

in instance segmentation framework (with HRNet backbone) is also initialized with the pretrained weights from RVNet or RVGan and finetuned on the same dataset.

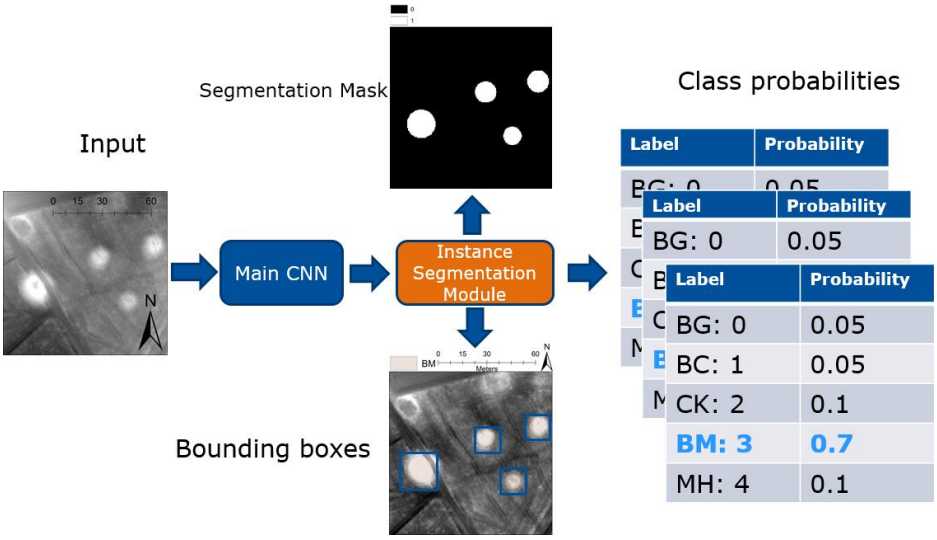


Figure 5.6.: Pipeline for instance segmentation of archaeological monuments and man-made terrain structures. The main CNN module can be initialized with random weights or pretrained weights from the main CNN modules in the SSL pretext, i.e., RVNet and RVGan's generator.

5.2.3. Semantic Segmentation of Archaeological Monuments and Terrain Structures

Instance segmentation models work well when objects in the input have an areal structure and covers big blobs of the input region. For linearly elongated thin structures or small objects not covering many pixels, the predictions are not as desired. With these considerations, another approach called semantic segmentation is studied. Semantic segmentation refers to pixel-wise labeling of inputs to a predefined set of categories. The predictions are given for every pixel regardless of the shape or size of objects. The pipeline for semantic segmentation is illustrated in Figure 5.7. It is the HRNet architecture as used in the RVNet and RVGan models in the pretext phase. Its main CNN module is also initialized with random weights or the pretrained weights from the main CNN module in RVNet and RVGan's generator. Additionally, as explained previously, the DeepLabV3+ (Chen et al., 2018) model with the ResNet backbone is also trained for semantic segmentation on the same data.

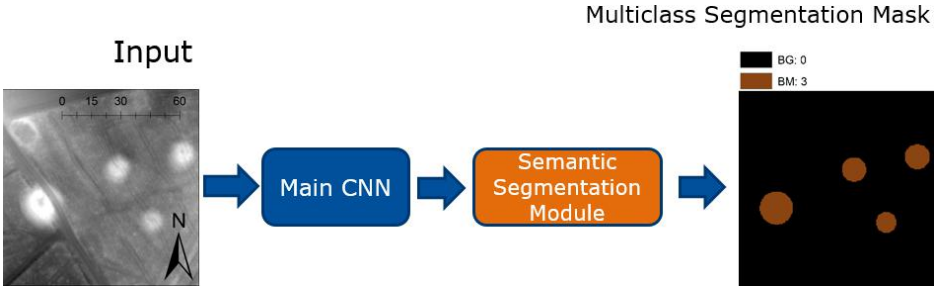


Figure 5.7.: Pipeline for semantic segmentation of archaeological monuments and man-made terrain structures. Similarly, the main CNN module can be initialized with random weights or pretrained weights from the main CNN modules in the SSL pretext, i.e., RVNet and RVGan's generator.

6. Experiments and Results

This chapter describes the experiments and evaluation results for this research. Section 6.1 contains conducted experiments for the SSL pretext with unlabeled DTM data. Sections 6.2-6.4 give details of quantitative evaluations for supervised downstream tasks including classification, instance segmentation and semantic segmentation. An overview of experiments with each method and each dataset is given in Talbe 6.1. Section 6.5 includes comparisons for predictions by deep learning models and manual annotations by three different people evaluated against the initial ground truth annotations. Section 6.6 compares the three supervised approaches qualitatively and a summary of experiments and results is given in Section 6.7.

Model	Backbone	Weights	Input	Output	Task category	Dataset	Section
RVNet	HRNet	-	DTMs	relief rasters	Pretext	DTMs & relief rasters	6.1
RVGan	HRNet	-	DTMs	relief rasters	Pretext	DTMs & relief rasters	6.1
HRNet	-	- / RVNet / RVGan	DTMs	Class labels	Classification	Areal	6.2
ResNet	-	-	DTMs	Class labels	Classification	Areal	6.2
Mask RCNN	ResNet	-	DTMs	Class labels & Coordinates & Masks	Instance segmentation	Areal / Linear	6.3
Mask RCNN	HRNet	- / RVNet / RVGan	DTMs	Class labels & Coordinates & Masks	Instance segmentation	Areal / Linear	6.3
HRNet	-	- / RVNet / RVGan	DTMs	Segmentation maps	Semantic segmentation	Areal / Linear / Stone quarries	6.4
DeepLabV3+	ResNet	-	DTMs	Segmentation maps	Semantic segmentation	Areal / Linear / Stone quarries	6.4
HRNet	-	-	DTM / LD / SLRM / SVF / Slope / POS / NEG / Combined	Segmentation maps	Semantic segmentation	Areal	6.4

Table 6.1.: Overview of methods and datasets used in the experiments linked to the corresponding sections.

6.1. Self Supervised Learning Pretext Experiments

In the pretext phase, 200 000 DTM patches of size 224×224 pixels are randomly cropped from the DTM for Lower Saxony and the corresponding relief visualization rasters, namely LD, SLRM, slope, SVF, POS and NEG are calculated. Both SSL pretext methods: RVNet and RVGan are trained to learn producing the said relief visualization rasters from the DTM patches. The experiments are done in Python and the Keras (Chollet et al., 2015) deep learning framework. Both approaches are trained for 50 epochs with a batch size of 20 examples using the Adam (Kingma and Ba, 2015) optimization algorithm. 80% (160 000) of the examples are used for training, 10% (20 000) for validation and 10% (20 000) for testing. RVNet is trained using the Binary Cross Entropy (BCE) function shown in Equation 6.1.

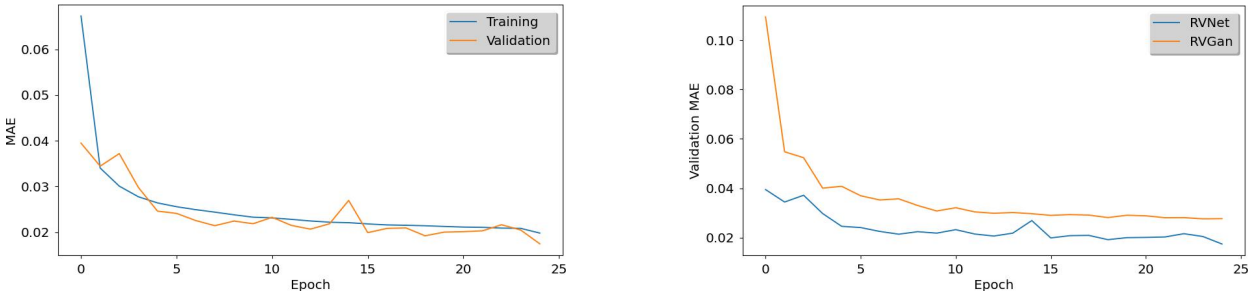
$$BCE = -\sum_{i=1}^{H}\sum_{j=1}^{W} y_{ij}\log(\hat{y}_{ij}) + (1 - y_{ij})\log(1 - (\hat{y}_{ij})) \tag{6.1}$$

Where H and W denote the height and width of the output rasters. $\hat{y}$ and y denote the predicted and original relief rasters, respectively.

RVGan is trained using the GAN objective function shown in Equation 2.36. Both methods are evaluated based on how close their predicted relief rasters are to the original relief rasters. This is quantified by the Mean Absolute Error (MAE) score explained in Section 2 and shown in Equation 2.15.

Model	BCE	MAE
RVNet	**0.493**	**0.017**
RVGan	0.496	0.027

Table 6.2.: Test results for RVGan and RVNet on the test set. Better results are in ***bold****.*



(a) MAE by RVNet for training and validation.

(b) MAE comparison for RVNet and RVGan.

Figure 6.1.: MAE plots for RVNet and RVGan.

The Mean Absolute Error (MAE) plot during training for the RVNet approach and the validation MAE for both RVNet and RVGan are shown in Figure 6.1. The BCE loss and MAE scores on the test set are listed in Table 6.2. BCE is the loss function the models try to minimize, hence the lower BCE indicates the better performance. MAE, as explained previously, shows how close the predicted pixels by the models are to the original relief rasters. The best possible value for both BCE and MAE is 0 and is achieved only if the predicted and original relief rasters are completely identical. Since the range of values for predictions and the original relief rasters is from 0 to 1, the best MAE score, i.e., 0 is achieved if the original relief rasters and the predicted rasters have all zeros or all ones in every pixel. The worst MAE score is 1 and is achieved if the original rasters have all ones and the predictions have all zeros in every pixel or vice versa. For BCE, the best possible score is 0 and the worst possible score is 15.43. Moreover, in addition to minimizing the objective function to get the lowest possible BCE and MAE score in reconstructing relief rasters, the generator in RVGan is trained to fool the discriminator by generating realistic examples. In the beginning of the training, the outputs of both generator and discriminator are random, hence the the generated relief rasters are random noise and the accuracy of discriminator is around 50%. As the training progresses, the discriminator learns to tell apart original relief rasters from those produced by the generator, thus scoring a high accuracy. Then, the generator learns to generate realistic examples and fool the discriminator bringing its accuracy down. The validation accuracy plot for the discriminator, depicting this phenomenon, is shown in Figure 6.2. After 20 epochs, the generated relief rasters are realistic enough that the discriminator thinks they come from the original data distribution.

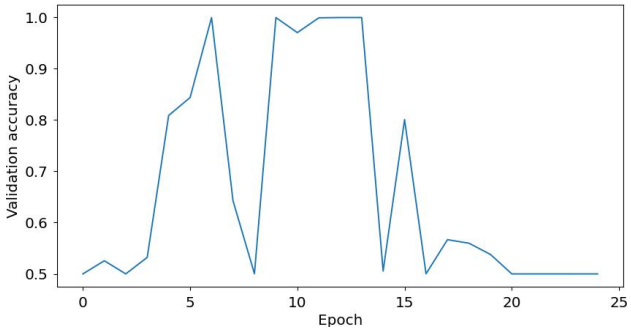
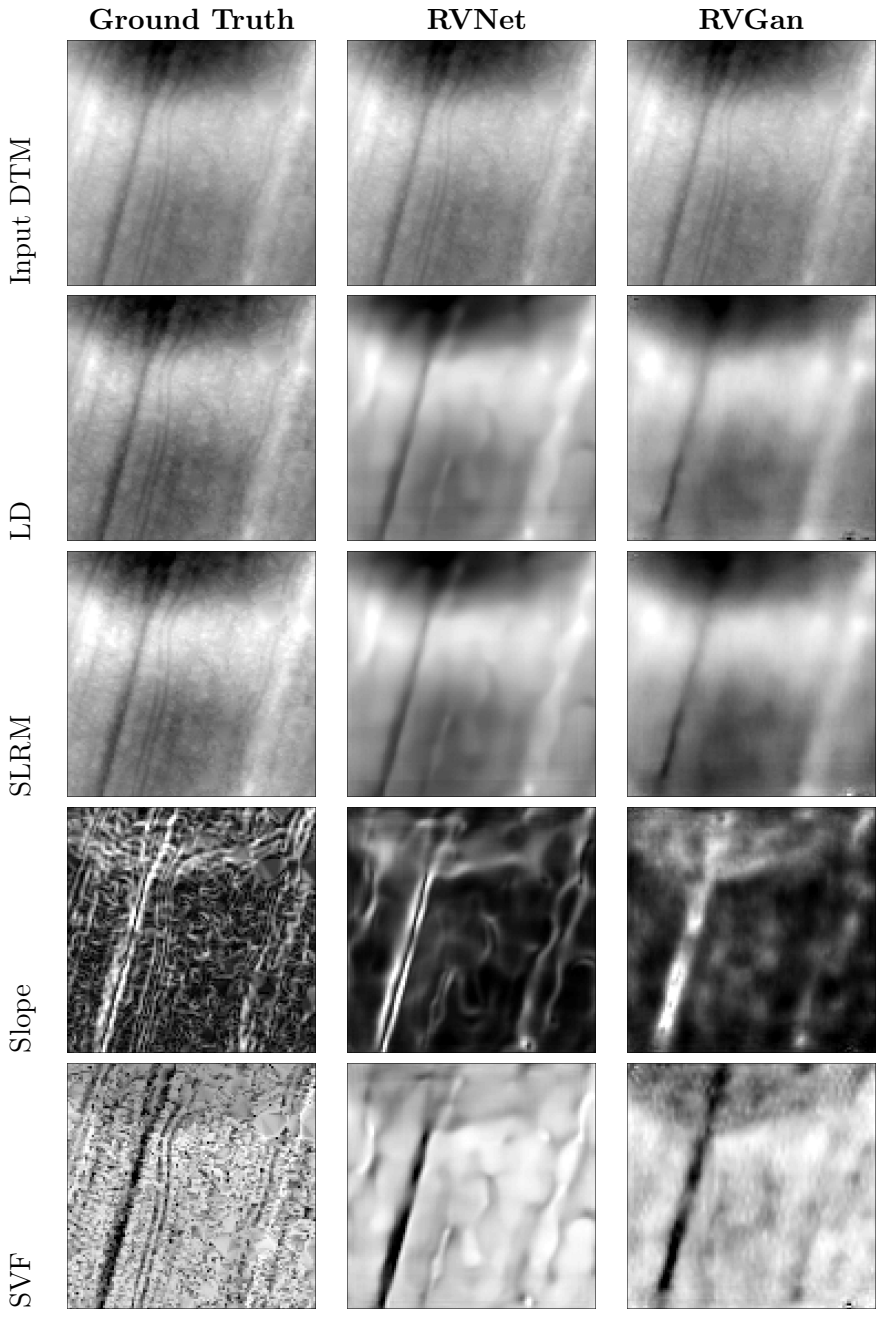


Figure 6.2.: Validation accuracy plot for the discriminator model.



Continued on next page

Table 6.3 – continued from previous page

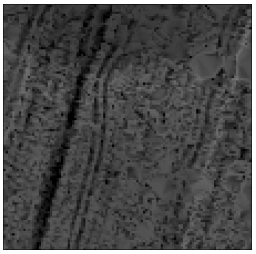
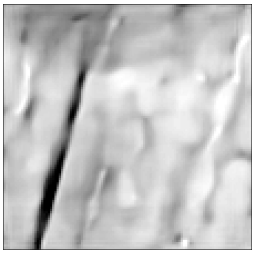
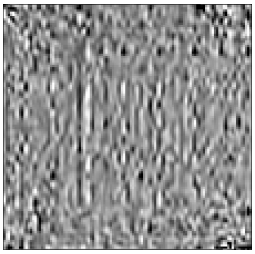
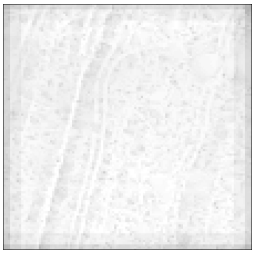
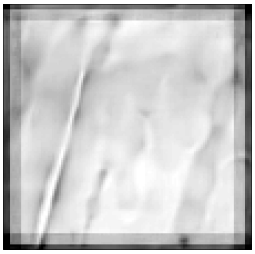
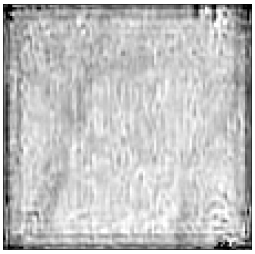
	Ground Truth	RVNet	RVGan
POS			
NEG			

Table 6.3.: Examples of relief visualization rasters by RVNet and RVGan

Examples of relief rasters generated by the RVNet and RVGan are illustrated in Table 6.3. As expected, the encoder-decoder approach (RVNet) learns the mapping from DTM patches to the relief rasters better than the GAN-based approach (RVGan). For example the POS and NEG rasters produced by the generator look like random noise. It is also consistent with the lower (better) BCE and MAE scores by RVNet shown in Table 6.2. This is by design, as explained in Chapter 5, since the ultimate goal is not generating perfect relief visualizations, but learning useful properties of the data that can help with the performance in supervised downstream tasks. The impact of RVGan is thus more significant in detection scores by the downstream tasks as discussed in the following sections.

Further examples of relief rasters by RVNet and RVGan are additionally shown in Appendix A.

6.2. Classification

For the classification task, 5 models are trained with inputs of height and width 32, 64, 96, 128 and 224 pixels separately. They are trained on the areal dataset to classify an input DTM into 5 categories including background (BG), bomb craters (BC), charcoal kilns (CK), burial mounds (BM) and mining holes (MH). The dataset is created from three separate regions for training, validation and testing. Number of examples for each category in each set is listed in Table 6.4. Similar to the pretext phase, experiments are conducted using Python and Keras (Chollet et al., 2015). The training is carried out for 50 epochs with batches of 20 examples, minimizing cross entropy using the Adam (Kingma and Ba, 2015) optimization algorithm. The objective function for training the classifiers is the categorical cross entropy function shown in Equation 2.16. Table 6.5 shows quantitative evaluation results on the test set using different input sizes for the HRNet and the ResNet models used as the classifiers. The best scores are achieved by the HRNet model with input dimensions of 64 pixels. The maximum diameter for an object in the dataset is 38 meters as shown in Table 4.1. For the DTM with a resolution of half a meter per pixel, 64 pixels would contain all the objects with a smaller diameter than 32 meters. Thus, it is expected for the model with input size of 64×64 to give the highest F1-score compared to the others.

	Bomb craters	Charcoal kilns	Burial Mounds	Mining holes	Background
Training	314	1560	833	1741	3351
Validation	169	479	357	481	-
Testing	134	504	220	764	

Table 6.4.: Number of examples for each category in the regions for training, validation and testing. For training the classifiers, random patches with none of the 4 categories are also created to help the models learn when an input does not contain any of the objects/structures.

Model	Input Dim.	F1-score	Precision	Recall
HRNet	32	**85.13**	**85.28**	**84.97**
ResNet	32	76.13	76.74	75.53
HRNet	64	**87.49**	**87.51**	**87.47**
ResNet	64	80.40	81.99	78.88
HRNet	96	**85.40**	**85.47**	**85.32**
ResNet	96	73.13	73.86	72.41
HRNet	128	**83.30**	**83.38**	**83.22**
ResNet	128	73.47	73.97	72.97
HRNet	224	**80.38**	**80.93**	**79.84**
ResNet	224	80.06	80.82	79.32

Table 6.5.: Evaluation results on the test set for HRNet and ResNet with random weights initialization on Harz areal dataset using different input dimensions.

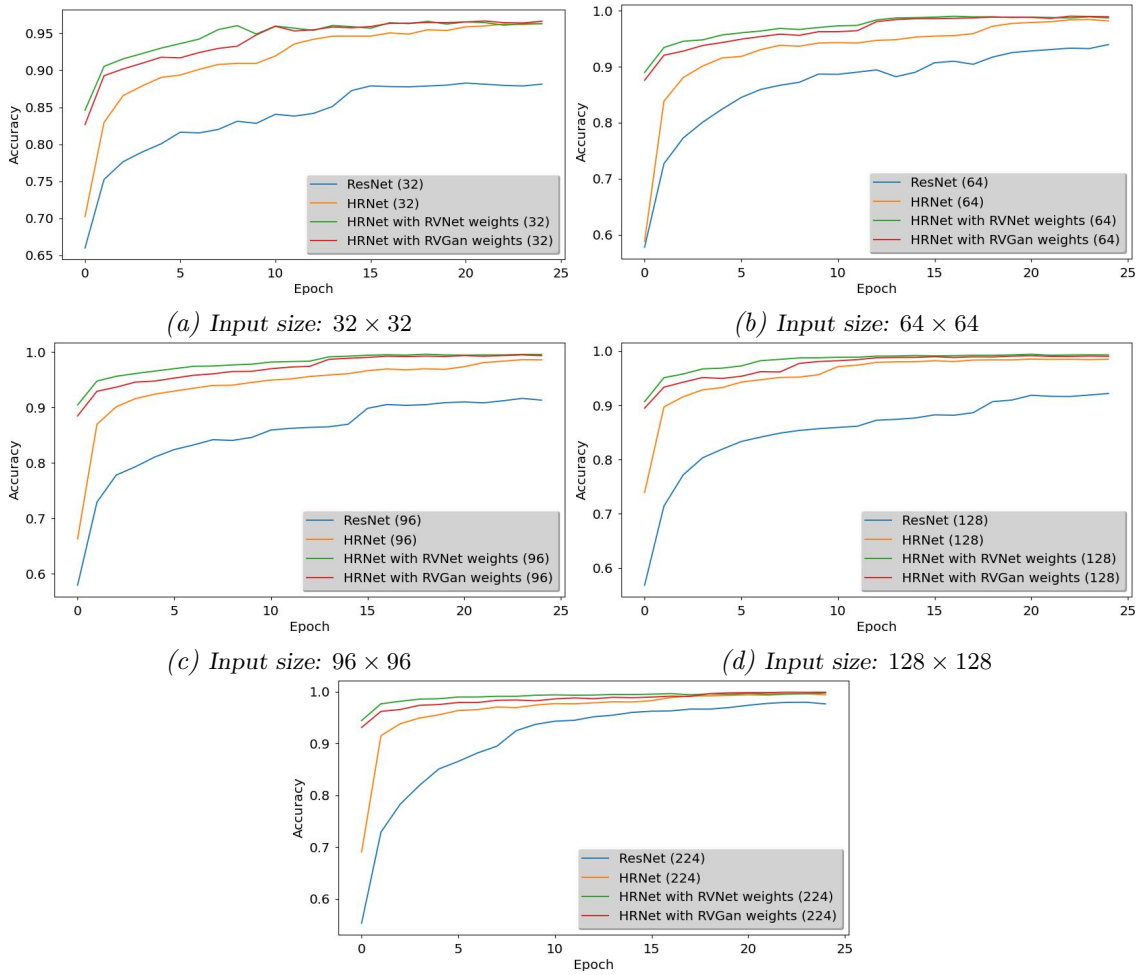
The HRNet model, as illustrated in Figure 5.4, is first trained with random weights. Additionally, the main CNN module in the architecture is initialized with the pretrained weights from RVNet and RVGan model and trained on the same dataset. The F1-scores for the classifiers with random weights, pretrained weights from RVNet and pretrained weights from RVGan are shown in Table 6.6. For smaller input sizes, the SSL pretraining does not help improve performance scores, but as the input sizes grow, the performance scores for HRNet using pretrained weights of RVNet and RVGan grow, and are higher than HRNet with random weight initialization. This is intuitive as the SSL pretraining is done with higher input sizes, i.e., 224×224 pixels.

	Pretrained weights	**32**	**64**	**96**	**128**	**224**
HRNet	-	**85.13**	**87.49**	**85.40**	**83.3**0	80.38
HRNet	RVNet	81.62	84.85	76.76	82.00	**88.23**
HRNet	RVGan	83.63	84.72	84.38	81.75	86.39

*Table 6.6.: Effect of SSL pretext on performance of HRNet with different input dimensions. Values denote $F1$ scores on the areal data test set. Better scores by different models are in **bold** and the highest score overall is shown in blue.*

The accuracy plots for training the models with different input sizes are shown in Figure 6.3. It is observed that using pretrained weights from the SSL pretext methods, i.e., RVNet and RVGan help the models converge faster compared to using randomly initialized weights. The lowest accuracy scores are gained by ResNet with each input size, and the performance of HRNet with random weight initializations reach those initialized with the weights from the SSL pretext methods only after approximately 20 epochs.

To further evaluate the performance of training from scratch with random weights and fine-tuning with pretrained weights from the SSL pretext method, i.e., RVGan, for input sizes of 64, 32 and 224 are shown in Tables 6.7-6.9. Confusion matrices for ResNet and HRNet with pretrained weights from RVNet, and input sizes of 96 and 128 are included in Appendix A. In general, the model trained from scratch is better at predicting bomb craters (BC) while for the rest of the objects,



(a) Input size: 32×32 *(b) Input size:* 64×64

(c) Input size: 96×96 *(d) Input size:* 128×128

(e) Input size: 224×224

Figure 6.3.: Accuracy plots for classification models with different input sizes.

it alternates depending the size of the input DTM patches. Bomb craters are very similar to mining holes and therefore, they are interchangeably mislabeled by the models. Due to the high number of examples for mining holes (1741) and smaller number of examples for bomb craters (314) during training, the models make more false predictions for bomb craters than mining holes. Burial mounds and charcoal kilns have more distinct structures and are therefore not mislabeled as much. 5 different models trained with different input sizes are then used by the pipeline shown in Figure 5.5 and Algorithm 1 to scan a large region and produce heatmaps for each object class. Results of the qualitative evaluation are shown in Section 6.6.

Actual \ Predicted	BG	BC	CK	BM	MH
BG	94.93	0.67	0.40	1.73	2.27
BC	31.51	41.10	2.28	0.46	24.20
CK	2.04	0.00	89.22	0.00	8.51
BM	7.44	0.00	0.41	90.50	1.03
MH	4.87	5.23	0.54	0.90	88.10

(a) Random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	94.27	0.53	1.07	1.33	2.80
BC	32.42	36.99	2.28	1.37	26.48
CK	2.38	0.00	79.00	0.11	18.27
BM	8.26	0.21	0.41	89.26	1.24
MH	3.16	4.69	1.44	1.26	89.09

(b) RVGan weights

Table 6.7.: Confusion matrix for the HRNet with input dimensions of 64 and different weight initializations. Values are in percent.

Actual \ Predicted	BG	BC	CK	BM	MH
BG	90.00	0.53	1.73	2.40	5.33
BC	20.09	47.03	1.83	0.00	30.59
CK	7.60	0.11	83.54	0.23	8.29
BM	8.47	0.21	1.45	88.84	0.41
MH	4.15	5.68	1.26	0.81	87.74

(a) Random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	91.20	0.53	1.20	2.93	4.13
BC	29.22	44.29	3.65	0.46	21.92
CK	7.60	0.00	78.66	0.45	13.05
BM	7.64	0.21	0.21	91.12	0.21
MH	6.67	6.04	0.99	0.81	85.12

(b) RVGan weights

Table 6.8.: Confusion matrix for the HRNet with input dimensions of 32 and different weight initializations. Values are in percent.

Actual \ Predicted	BG	BC	CK	BM	MH
BG	80.40	1.33	5.07	5.33	7.87
BC	12.79	52.97	6.39	0.91	26.48
CK	9.53	0.68	59.82	1.48	28.26
BM	10.12	0.21	1.45	85.54	2.07
MH	4.24	8.48	2.16	0.63	84.13

(a) Random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	98.40	0.13	0.00	0.40	1.07
BC	55.71	16.89	0.46	0.00	26.48
CK	6.47	0.00	87.85	0.11	5.33
BM	8.68	0.21	0.21	89.67	0.62
MH	4.87	5.59	0.27	0.09	88.82

(b) RVGan weights

Table 6.9.: Confusion matrix for the HRNet with input dimensions of 224 and different weight initializations. Values are in percent..

6.3. Instance Segmentation

This section contains quantitative results for experiments with the instance segmentation approach on two different datasets. Section 6.3.1 lists the results for instance segmentation using the areal dataset, i.e., the same dataset used in the classification experiments as well and described in Section 4.2.1. Section 6.3.2 contains results for a dataset of linear structures as explained in Section 4.2.2. On both the datasets, the instance segmentation architecture, Mask RCNN, shown in Figure 5.6 is first trained with random weights using both HRNet and ResNet as the backbone. Additionally, it is trained after initializing the main CNN module of the architecture with pretrained weights from the pretext, namely, the RVNet and RVGan weights. While the rest of the parameters during training are specific to each dataset, the general objective function for instance segmentation is composed of three functions for the three branches, i.e., classification, bounding box regression and segmentation branches. It is defined by Equation 6.2.

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{mask}} \tag{6.2}$$

Where $\mathcal{L}_{\text{cls}}$ and $\mathcal{L}_{\text{mask}}$ are the cross entropy function shown in Equation 2.16, and $\mathcal{L}_{\text{box}}$ is the smooth L_1 loss shown in Equation 6.3 below.

$$\mathcal{L}_{\text{box}_{\alpha=1}} = \begin{cases} |y - \hat{y}| & : |y - \hat{y}| > \alpha \\ \frac{1}{\alpha}(y - \hat{y})^2 & : |y - \hat{y}| \leq \alpha \end{cases} \tag{6.3}$$

6.3.1. Areal Dataset

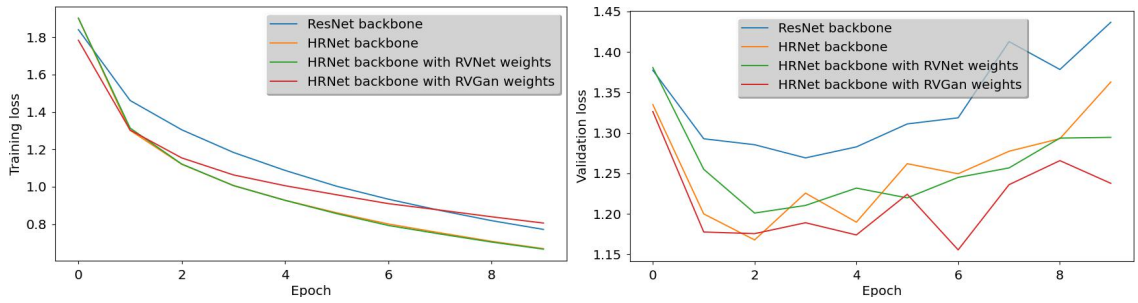
The areal dataset explained in Section 4.2.1 and used in the classification experiment is used to train instance segmentation models as well. The same tools (Python and Keras) and the same training, validation and testing regions are used for this approach as well. The input DTMs are

prepared with dimensions of 256 × 256 each and the batch size is set to 4 examples. The data contains examples of 4 classes including bomb craters (BC), charcoal kilns (CK), burial mounds (BM) and mining holes (MH). The Mean Average Precision (MAP) scores on the test data achieved by Mask RCNN with the ResNet and HRNet backbones (trained with random weights), and with the HRNet backbone finetuned with pretrained weights from the RVNet and RVGan are shown in Table 6.10. As observed in the table, Mask RCNN with HRNet backbone outperforms the same with ResNet backbone. Additionally, finetuning with the pretrained weights of RVNet and RVGan both achieve higher MAP scores compared to random weight initialization proving the positive impact of SSL pretraining with unlabeled data.

Backbone	Pretrained weights	MAP
HRNet	-	53.25
ResNet	-	46.15
HRNet	RVNet	**58.38**
HRNet	RVGan	58.29

Table 6.10.: Effect of SSL pretraining on Mask RCNN on Areal Dataset. Values range from 0 to 100 and higher values are better. The best score is shown in ***bold****.*

Training and validation loss plots for the Mask RCNN model with different backbones are shown in Figure 6.4. The models with the HRNet backbones converge faster and the validation loss for the model with pretrained weights from RVGan are more consistent and smaller (better). The smallest validations are achieved after the 5^{th} or 6^{th} epochs after which the models overfit, i.e., training loss decreases but the validation loss increases.



(a) Training loss *(b) Validation loss*

Figure 6.4.: Training and validation loss for instance segmentation with Mask RCNN on the areal dataset.

The trained models are used to make predictions on examples from the test set and the outputs are converted (from raster) to polygons and saved as vector files (i.e., shape files) that can be processed and visualized in ArcGIS. The predictions and ground truth are given as inputs to the *Spatial Join* tool in ArcGIS which joins features based on their spatial relationships. It can be used to determine the intersecting and non-intersecting polygons. Based on the outputs of the *Spatial Join* tool, the confusion matrices for predictions by each model are computed. Table 6.11 shows the confusion matrices for the number of predicted instances that intersect with the ground truth. It includes any prediction that intersects the ground truth. The percentage of intersection area between the predictions and the ground truth are listed in Table 6.12. In both tables *OP* indicates the percentage of over-predictions, i.e., predictions by the models that do not exist in the ground truth. *NC* stands for *Not Captured*, i.e., percentage of example objects that exist in the ground truth, but were not labeled by the models. *OP* and *NC* are explained visually in Figure 6.5. In general, the Mask RCNN model with the HRNet backbone outperforms that with the ResNet backbone. Additionally, the SSL pretraining has a positive impact on the performance as indicated by the results of Mask RCNN initialized with RVNet and RVGan weights. Examples

of bomb crater are mostly confused with mining holes. This is because the two structures look quite similar. The models tend to label more examples of bomb craters as mining holes than the other way around because of the imbalance in the data, i.e., the number of annotated mining holes (1741) outweighs that of bomb craters (314) during training. Examples of burial mounds and mining holes are confused with charcoal kilns. Charcoal kiln structures are eroded over time showing burial mound-like structures around the perimeters and mining hole-like structures in the middle part contributing to the confusion by the models. Qualitative evaluation results are shown in Section 6.6.

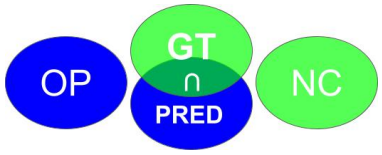


Figure 6.5.: Example illustration showing the intersection (∩) between predictions (PRED) and ground truth (GT) examples, ground truth examples not captured by the model (NC), and over-predictions (OP), i.e., predictions by the model that do not intersect any ground truth example.

Predicted

Actual	BC	CK	BM	MH	NC
BC	58.21	5.97	0.00	15.67	20.15
CK	0.00	90.48	0.40	7.14	1.98
BM	0.91	8.64	90.45	0.00	0.00
MH	2.88	24.21	6.41	62.17	4.32
OP	43.02	75.07	81.95	90.88	

(a) HRNet backbone and random weights

Predicted

Actual	BC	CK	BM	MH	NC
BC	48.51	22.39	0.00	11.94	17.16
CK	0.00	89.29	0.00	9.92	0.79
BM	0.91	25.00	72.73	0.45	0.91
MH	1.70	54.45	1.18	36.91	5.76
OP	53.49	82.50	56.89	96.10	

(b) ResNet backbone and random weights

Predicted

Actual	BC	CK	BM	MH	NC
BC	63.43	4.48	0.00	17.16	14.93
CK	0.00	95.04	0.00	4.17	0.79
BM	0.91	17.73	80.91	0.00	0.45
MH	4.32	22.64	1.70	66.36	4.97
OP	46.90	81.59	70.39	90.70	

(c) HRNet backbone and RVNet weights

Predicted

Actual	BC	CK	BM	MH	NC
BC	66.42	2.99	0.75	14.93	14.93
CK	0.00	96.03	0.00	2.98	0.99
BM	0.45	8.64	90.91	0.00	0.00
MH	3.80	8.25	0.39	82.20	5.37
OP	61.86	87.10	69.14	87.13	

(d) HRNet backbone and RVGan weights

Table 6.11.: Confusion matrix for Mask RCNN results on the areal data test set with different backbones and weight initializations. Values show percentage for number of predicted instances that intersect those of the ground truth. NC denotes percentage for number of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage of predicted instances by the models that do not intersect any ground truth.

Predicted

Actual	BC	CK	BM	MH	NC
BC	70.44	9.46	0	22.58	1.50
CK	0	75.53	1.08	14.42	2.40
BM	0.14	5.16	98.62	0	0
MH	4.47	15.81	5.09	85.20	2.75
OP	21.80	63.32	65.31	71.26	

(a) HRNet backbone and random weights

Predicted

Actual	BC	CK	BM	MH	NC
BC	52.6	70.27	0	74.89	0.79
CK	0	74.56	0	15.12	1.10
BM	0.08	19.78	96.06	0.45	0.23
MH	3.01	43.34	0.85	75.83	2.85
OP	32.32	70.23	36.4	77.3	

(b) ResNet backbone and random weights

Predicted

Actual	BC	CK	BM	MH	NC
BC	77.08	3.08	0	84.05	0.67
CK	0	72.01	0	9.57	0.86
BM	0.19	11.63	94.22	0	0.21
MH	5.94	15.11	1.1	86.64	2.48
OP	26.57	73.37	54.18	70.48	

(c) HRNet backbone and and RVNet weights

Predicted

Actual	BC	CK	BM	MH	NC
BC	83.59	1.99	1.77	77.58	0.63
CK	0	76.42	0	5.21	1.35
BM	0.09	6.76	95.82	0	0
MH	7.04	4.96	0.52	84.99	2.55
OP	31.57	79.52	49.15	69.10	

(d) HRNet backbone and RVGan weights

Table 6.12.: Percentages for the intersection area of predicted instances in the areal data test set by Mask RCNN and the ground truth. NC denotes percentage for the the area of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage for the area of predicted instances by the models that do not intersect any ground truth.

6.3.2. Linear Dataset

The second dataset used by the instance segmentation method, i.e., Mask RCNN, is the linear dataset explained in Section 4.2.2. It contains linear structures including ditches, paths, roads and hollow ways. Similar to the areal dataset, the training, validation and testing set for this dataset have also been prepared from three separate non-overlapping regions. The number of examples and the area for each category are listed in Table 6.13.

	Ditches		**Paths**		**Roads**		**Hollow ways**	
	Examples	**Area**	**Examples**	**Area**	**Examples**	**Area**	**Examples**	**Area**
Training	364	0.73 km^2	668	2.94 km^2	82	1.96 km^2	366	0.83 km^2
Validation	35	0.21 km^2	135	0.96 km^2	5	0.68 km^2	59	0.31 km^2
Test	40	0.23 km^2	102	0.90 km^2	4	0.22 km^2	23	0.20 km^2

Table 6.13.: Number of examples and total area for each category in the regions for training, validation and testing in the linear dataset.

Input DTM patches with sizes of 128 × 128 pixels and a batch size of 8 are used to train the instance segmentation segmentation models with random weights and also with the pretrained weights from the SSL pretext. The MAP scores for the model with the ResNet backbone, and the HRNet backbones with random weights, RVNet weights and RVGan weights on the test set for this data are listed in Table 6.14. As observed, SSL pretraining leads to better scores on this data as well. Similar to the experiments with the areal dataset, the *Spatial Join* tool in ArcGIS is used to create confusion matrices and also the percentages for the intersection area of predictions by the deep learning models and the ground truth. The confusion matrices are shown in Table 6.15 and the corresponding percentages of intersection area for predictions and the ground truth are listed in Table 6.16. Examples of ditches, roads and hollow ways are mostly confused with those of paths and the majority of path instances are falsely categorized as roads, as shown in Table 6.11. However, the correctly classified examples have the highest intersection with the ground truths for all the categories as shown in Table 6.12. Qualitative results are shown in Section 6.6.

Backbone	**Pretrained weights**	**MAP**
HRNet	-	48.77
ResNet	-	49.45
HRNet	RVNet	50.73
HRNet	RVGan	**54.95**

Table 6.14.: Effect of SSL pretraining on Mask RCNN on Linear Dataset. Values range from 0 to 100 and higher values are better. The best score is shown in ***bold****.*

The training and validation loss plots are shown in Figure 6.6. Consistent with the the preivous experiments and the MAP scores on the test set shown in Table 6.14, the Mask RCNN model using the pretrained weights from the RVGan model in the pretext converges faster and shows the smallest loss values in the training and validation sets. Optimal number of training epochs before the models start overfitting are 14 or 15 epochs.

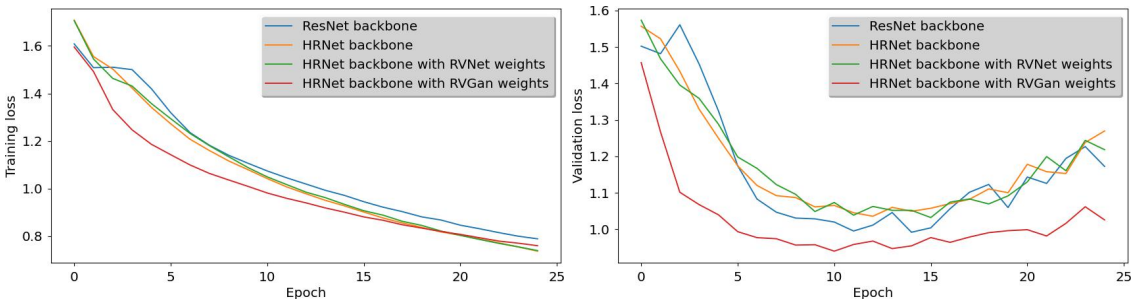


Figure 6.6.: Training and validation loss for instance segmentation with Mask RCNN on the linear dataset.

Predicted

Actual	Ditch	Path	Road	H-way	NC
Ditch	45.13	17.65	13.39	7.39	16.43
Path	7.43	60.56	16.07	5.36	10.58
Road	34.13	49.52	11.28	5.07	0.00
H-way	21.79	31.53	13.27	17.44	15.97
OP	11.39	4.50	14.38	18.38	

(a) HRNet backbone and random weights

Predicted

Actual	Ditch	Path	Road	H-way	NC
Ditch	40.26	13.83	21.04	10.17	14.70
Path	3.89	53.39	19.44	7.47	15.81
Road	20.51	51.58	22.38	5.53	0.00
H-way	20.23	40.87	15.15	12.78	10.97
OP	15.15	8.12	4.81	24.70	

(b) ResNet backbone and random weights

Predicted

Actual	Ditch	Path	Road	H-way	NC
Ditch	37.91	29.65	13.30	10.35	8.78
Path	5.27	60.04	12.18	14.08	8.42
Road	28.78	56.14	3.84	11.24	0.00
H-way	19.00	37.18	6.72	27.76	9.34
OP	12.72	8.50	17.56	20.71	

(c) HRNet backbone and RVNet weights

Predicted

Actual	Ditch	Path	Road	H-way	NC
Ditch	11.48	40.00	19.57	12.78	16.17
Path	2.51	55.12	8.55	20.56	13.26
Road	0.00	72.09	27.23	0.69	0.00
H-way	4.50	32.02	5.57	42.01	15.89
OP	14.63	2.29	4.31	21.0	

(d) HRNet backbone and RVGan weights

Table 6.15.: Confusion matrix for Mask RCNN results on the linear data test set with different backbones and weights initializations. Values show percentage for number of predicted instances that intersect those of the ground truth. NC denotes percentage for number of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage of predicted instances by the models that do not intersect any ground truth.

Predicted

Actual	Ditch	Path	Road	H-way	NC
Ditch	23.41	7.95	13.69	12.82	5.98
Path	0.28	47.44	48.74	2.81	3.88
Road	2.11	27.85	98.24	1.09	0
H-way	6.42	19.46	20.05	36.51	5.98
OP	13.32	3.22	4.90	7.05	

(a) HRNet backbone and random weights

Predicted

Actual	Ditch	Path	Road	H-way	NC
Ditch	18.12	8.14	20.18	12.55	7.66
Path	0.19	40.52	41.97	3.40	6.17
Road	0.60	13.59	96.89	1.01	0
H-way	3.42	20.82	36.50	22.46	4.04
OP	20.78	8.40	3.29	19.74	

(b) ResNet backbone and random weights

Predicted

Actual	Ditch	Path	Road	H-way	NC
Ditch	24.10	10.49	9.06	15.11	3.48
Path	0.39	58.89	38.62	5.59	2.38
Road	2.03	35.61	98.50	2.09	0
H-way	5.51	20.89	12.24	45.42	3.02
OP	14.50	6.10	4.07	14.59	

(c) HRNet backbone and RVNet weights

Predicted

Actual	Ditch	Path	Road	H-way	NC
Ditch	1.12	9.43	6.53	10.25	12.94
Path	0.07	57.48	37.95	5.3	4.04
Road	0	27.83	99.24	0.01	0
H-way	0.27	22.62	5.10	36.42	5.66
OP	26.40	33.90	1.06	19.45	

(d) HRNet backbone and RVGan weights

Table 6.16.: Percentages for the intersection area of predicted instances in the linear data test set by Mask RCNN and the ground truth in the linear dataset. NC denotes percentage for the the area of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage for the area of predicted instances by the models that do not intersect any ground truth.

6.4. Semantic Segmentation

Similar to instance segmentation, experiments on the two datasets are conducted using the semantic segmentation model shown in Figure 5.7 initialized with random weights, and also with

the pretrained weights from SSL pretext, namely, the RVNet and RVGan. Additionally, the semantic segmentation experiment is conducted on the stone quarries dataset explained in Section 4.2.3. Section 6.4.1 describes experiments on the areal dataset. Section 6.4.2 gives details of experiments on the linear dataset. Finally, Section 6.4.3 contains experiments and results on the stone quarries dataset. While the rest of the parameters during training are specific to the datasets, the objective function for training semantic segmentation models are the categorical cross entropy function defined in Equation 2.16.

6.4.1. Areal Dataset

The positive impact of relief visualization rasters on classification and instance segmentation tasks are already discussed in Kazimi et al. (2020) and Kazimi et al. (2019a). To understand the effect of different relief visualization rasters on semantic segmentation, the 6 previously mentioned relief rasters and the DTM patches for the areal dataset explained in Section 4.2.1 are separately used to train the HRNet model. Additionally, all the 6 relief rasters are combined in 6 channels in the order listed in Table 6.17 as the *combined* input for the semantic segmentation model. The raster patches are prepared with sizes of 224×224 pixels and used in batches of 20 examples to train the HRNet for 20 epochs separately for every input raster and/or the combinations. The MIOU and class-wise Intersection Over Union (IOU) scores for background (BG), bomb craters (BC), charcoal kilns (CK), burial mounds (BM) and mining holes (MH) achieved using different input rasters are shown Table 6.17. As observed, the original DTM is good for detecting mining holes and identifying background pixels that do not contain any structures/objects of interest. The LD raster is effective for bomb craters and burial mounds while the slope raster leads to the highest IOU score for charcoal kilns. Finally SVF raster achieves the top overal MIOU score.

Input raster	MIOU	BG	BC	CK	BM	MH
Combined	61.58	96.37	51.53	60.09	58.72	41.19
DTM	62.37	**96.43**	50.94	62.11	61.04	**41.33**
LD	61.82	96.25	**56.97**	57.05	**64.13**	34.70
SLRM	59.82	96.38	43.05	58.28	65.26	36.13
SLOPE	61.46	96.12	53.60	**62.72**	59.95	34.90
SVF	**62.83**	96.47	54.63	61.09	64.02	37.91
POS	61.10	96.42	47.77	61.94	63.77	35.60
NEG	54.93	96.04	40.56	51.46	58.43	28.18

Table 6.17.: IOU by HRNet on the test set with different relief raster as the input. Combined means all the relief rasters except DTM in 6 channels in the order listed in the table.

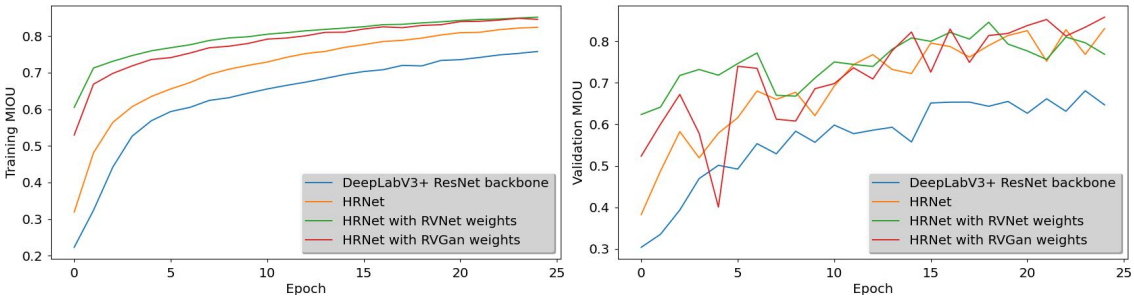
Model	Pretrained weights	MIOU	BG	BC	CK	BM	MH
HRNet	-	61.49	96.36	51.41	59.36	62.56	37.77
DeepLabV3+	-	57.50	96.08	38.75	63.53	57.39	31.73
HRNet	RVNet weights	62.68	**96.69**	42.23	**65.05**	**68.37**	41.03
HRNet	RVGan weights	**66.21**	96.65	**62.07**	63.13	67.88	**41.31**

Table 6.18.: Effect of SSL pretraining on semantic segmentation performance of the areal dataset. Values show the IOU scores on the test set.

Additionally, the semantic segmentation architecture shown in Figure 5.7 using HRNet is trained for 50 epochs with randomly initialized weights and also pretrained weights to evaluate the impact of SSL pretraining on semantic segmentation as well. HRNet is also compared to the DeepLabV3+ model (Chen et al., 2018) with the ResNet backbone, which is also trained for semantic segmentation on the same dataset. The MIOU scores and class-wise IOU scores for background (BG), bomb craters (BC), charcoal kilns (CK), burial mounds (BM) and mining holes (MH) on the test data by ResNet (i.e., DeepLabV3+ with ResNet backbone), and HRNet with random weights, RVNet

weights and RVGan weights are shown in Table 6.18. As it is observed in the table, SSL pretraining improves MIOU score from 61.49 percent with random weight initialization to 66.21 percent with SSL pretrained weights.

The training and validation MIOU plots for the semantic segmentation models are shown in Figure 6.7. Consistent with the previous experiments and the MIOU scores listed in Table 6.18, models using the pretrained weights from SSL pretext methods, i.e., RVNet and RVGan converge faster and show the highest MIOU values for training and validation sets at each epoch. The optimal number of epochs after which the performance scores stays constant is 25 (epochs $26 - 50$ are not shown in Figure 6.7).



(a) Training MIOU *(b) Validation MIOU*

Figure 6.7.: Training and validation MIOU plot for semantic segmentation on the areal dataset.

For the semantic segmentation experiments on the areal dataset, similar to the instance segmentation experiments, the *Spatial Join* tool in ArcGIS is used to calculate the confusion matrices and the percentages for the area of intersection between the predictions and the ground truth examples. As shown in Table 6.19 the HRNet models finetuned with weight initialization from the SSL pretext methods, i.e., RVNet and RVGan, score higher. Similarly, the intersection areas of predictions and ground truth for these models are higher than those of HRNet with random weights and DeepLabV3+ as listed in Table 6.20.

Additionally, the results by the semantic segmentation approach shown in Tables 6.19 and 6.20 are better than those by the instance segmentation approach shown in Tables 6.11 and 6.12. The instance segmentation approach also makes more over-predictions (OP) as indicated in the last row in each table. In other words, the instance segmentation models label a higher percentage of background pixels as the object classes. Looking at the diagonal values in Tables 6.19 and 6.20, it might appear that the randomly initialized models perform better, however, it can be observed from the NC columns that the percentage of ground truth examples not captured by models initialized with random weights are higher. Additionally, the OP columns show that in general, the percentage of over-predictions, i.e., background pixels labeled as objects by the models with the random weights are also higher, which is not desired. Qualitative evaluation results are shown in Section 6.6.

Predicted

Actual	BC	CK	BM	MH	NC
BC	64.93	0.75	0.75	9.70	23.88
CK	0.00	93.45	0.00	0.99	5.56
BM	0.00	0.00	96.82	0.00	3.18
MH	3.27	0.13	0.26	66.36	29.97
OP	11.81	2.07	8.09	15.32	

(a) HRNet with random weights

Predicted

Actual	BC	CK	BM	MH	NC
BC	72.39	0.00	0.00	2.24	25.37
CK	1.79	89.88	2.38	1.19	4.76
BM	0.91	0.00	95.91	0.00	3.18
MH	8.77	0.39	0.39	63.87	26.57
OP	12.50	2.56	7.38	24.24	

(b) DeepLabV3+ with random weights

Predicted

Actual	BC	CK	BM	MH	NC
BC	60.45	1.49	0.00	14.18	23.88
CK	0.00	96.63	0.00	1.19	2.18
BM	0.00	0.45	94.55	0.00	5.00
MH	3.53	0.52	0.00	71.99	23.95
OP	35.71	10.34	19.69	9.45	

(c) HRNet with RVNet weights

Predicted

Actual	BC	CK	BM	MH	NC
BC	72.39	0.00	0.00	7.46	20.15
CK	0.00	95.24	0.00	2.58	2.18
BM	0.00	0.00	96.36	0.00	3.64
MH	3.40	0.52	0.00	73.17	22.91
OP	6.82	2.81	2.75	21.46	

(d) HRNet with RVGan weights

Table 6.19.: Confusion matrix for semantic segmentation results on the areal data test set. Values show percentage for number of predicted instances that intersect those of the ground truth. NC denotes percentage for number of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage of predicted instances by the models that do not intersect any ground truth.

Predicted

Actual	BC	CK	BM	MH	NC
BC	75.48	1.84	0.37	9.10	3.55
CK	0	64.95	0	0.15	4.3
BM	0	0	92.52	0	3.23
MH	2.66	0.20	0.01	58.59	20.19
OP	3.18	0.19	2.22	8.75	

(a) HRNet with random weights

Predicted

Actual	BC	CK	BM	MH	NC
BC	86.46	0	0	4.22	1.74
CK	0.3	70.29	1.64	0.35	6.02
BM	0.16	0	92.94	0	1.76
MH	8.22	0.34	0.7	60.71	17.16
OP	3.07	0.28	1.14	7.76	

(b) DeepLabV3+ with random weights

Predicted

Actual	BC	CK	BM	MH	NC
BC	71.93	0.06	0	22.21	1.58
CK	0	70.91	0	0.25	1.91
BM	0	0.07	86.44	0	2.75
MH	2.44	0.82	0	66.13	14.18
OP	2.02	0.45	0.55	9.93	

(c) HRNet with RVNet weights

Predicted

Actual	BC	CK	BM	MH	NC
BC	82.71	0	0	7.97	1
CK	0	69.71	0	0.38	2.24
BM	0	0	90.53	0	1.72
MH	2.46	0.73	0	65.16	13.35
OP	1.77	0.27	0.89	9.04	

(d) HRNet with RVGan weights

Table 6.20.: Percentages for the intersection area of predicted instances by semantic segmentation models and the ground truth in the areal dataset. NC denotes percentage for the the area of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage for the area of predicted instances by the models that do not intersect any ground truth.

6.4.2. Linear Dataset

The linear dataset explained in Section 4.2.2 and used in the instance segmentation experiment in Section 6.3.2 is used in the semantic segmentation experiment as well. The input DTMs are prepared with sizes of 128×128 pixels each. Similar to previous experiments, the DeepLabV3+ with ResNet backbone, and the HRNet model with random weights, RVNet weights, and RVGan weights are trained for 50 epochs and a batch size of 20 examples. The MIOU scores and class-wise IOU scores for background (BG), ditch, path, road and hollow way classes in the test data are listed in Table 6.21. HRNet significantly outperforms DeepLabV3+ with the ResNet backbone and SSL pretraining is observed to have increased the performance scores compared to random weight initialization in this experiment as well, as indicated by the IOU scores.

Model	Pretrained weights	MIOU	BG	Ditch	Path	Road	Hollow way
HRNet	-	52.24	91.35	17.73	44.17	81.61	26.34
DeepLabV3+	-	44.15	89.45	14.22	30.91	73.12	13.05
HRNet	RVNet weights	**53.46**	91.21	**18.70**	44.98	81.68	**30.71**
HRNet	RVGan weights	53.04	**91.45**	17.52	**47.78**	**81.91**	26.57

Table 6.21.: Effect of SSL pretraining on semantic segmentation of the linear data test set. Values show IOU scores.

The training and validation MIOU plots for semantic segmentation on this datasets are shown in Figure 6.8. As observed, the convergence for models using pretrained weights from the SSL pretext methods, i.e., RVNet and RVGan are faster and the highest MIOU values for training and validation at each epoch belong to the model with RVGan weights.

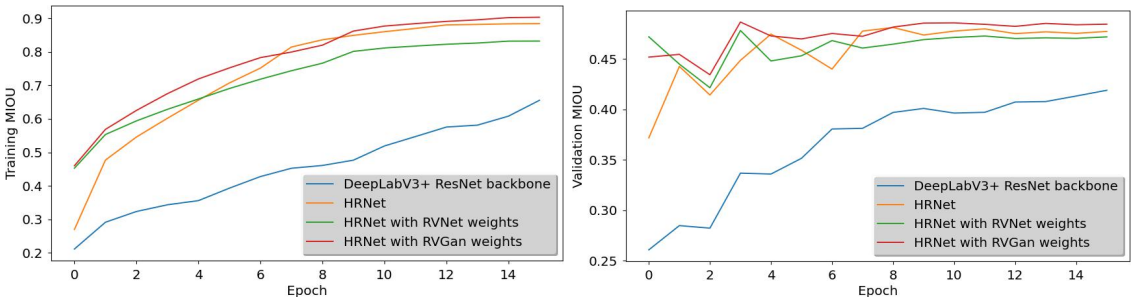


Figure 6.8.: Training and validation MIOU plot for semantic segmentation on the linear dataset.

Similar to previous experiments, the *Spatial Join* tool in ArcGIS is used to calculate the confusion matrices and the percentage of intersecting polygons in the predicted and ground truth examples. Consistent with the previous results, HRNet performs better than DeepLabV3+ with the ResNet backbone and pretrained models produce higher scores, as evidenced by the confusion matrices in Table 6.22. The area of intersecting polygons in the prediction and ground truth examples line up nicely as well as shown in Table 6.23. Additionally, the results by the semantic segmentation approaches listed in Tables 6.22 and 6.23 are significantly better than those obtained by the instance segmentation approaches listed in Tables 6.15 and 6.16. The semantic segmentation approaches do not confuse categories including ditches, roads and hollow ways with examples of the path category as drastically as the instance segmentation approaches. However, the over-prediction (OP) percentage for semantic segmentation models are higher than the instance segmentation models. Results of qualitative evaluation are shown in Section 6.6.

Actual \ Predicted	Ditch	Path	Road	H-way	NC
Ditch	58.43	11.83	6.17	4.61	18.96
Path	13.30	56.63	11.66	6.35	12.05
Road	23.94	9.50	65.42	1.14	0.00
H-way	18.67	7.29	2.21	49.71	22.11
OP	74.48	80.32	29.16	90.87	

(a) HRNet with random weights

Actual \ Predicted	Ditch	Path	Road	H-way	NC
Ditch	56.09	12.17	12.09	1.48	18.17
Path	14.56	39.31	23.59	2.42	20.13
Road	27.00	4.61	67.93	0.37	0.09
H-way	36.12	10.16	12.53	18.18	23.01
OP	90.11	96.64	88.22	98.53	

(b) DeepLabV3+ with random weights

Actual \ Predicted	Ditch	Path	Road	H-way	NC
Ditch	56.00	13.39	1.39	7.57	21.65
Path	6.65	58.44	10.89	9.55	14.47
Road	23.94	10.10	64.82	1.14	0.00
H-way	9.91	5.00	4.83	61.92	18.35
OP	66.21	64.71	37.61	91.39	

(c) HRNet with RVNet weights

Actual \ Predicted	Ditch	Path	Road	H-way	NC
Ditch	65.13	12.87	5.39	1.22	15.39
Path	12.79	60.22	9.85	4.28	12.87
Road	20.10	11.06	68.16	0.64	0.05
H-way	17.36	6.31	0.57	59.21	16.54
OP	74.14	83.80	43.88	90.56	

(d) HRNet with RVGan weights

Table 6.22.: Confusion matrix for semantic segmentation results on the linear data test set. Values show percentage for number of predicted instances that intersect those of the ground truth. NC denotes percentage for number of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage of predicted instances by the models that do not intersect any ground truth.

Actual \ Predicted	Ditch	Path	Road	H-way	NC
Ditch	27.84	3.34	0.43	5.25	7.98
Path	0.57	53.58	7.48	1.28	4.09
Road	0.29	0.98	89.29	0.05	0
H-way	2.76	1.4	0.22	34.29	12.14
OP	84.14	63.31	24.38	93.22	

(a) HRNet with random weights

Actual \ Predicted	Ditch	Path	Road	H-way	NC
Ditch	15.25	0.99	2.61	1.55	6.58
Path	1.05	20.55	21.66	0.32	6.41
Road	0.34	0.68	86.50	0.02	0
H-way	5.21	1.05	3.44	7.63	11.88
OP	90.13	90.99	69.31	97.19	

(b) DeepLabV3+ with random weights

Actual \ Predicted	Ditch	Path	Road	H-way	NC
Ditch	23.67	1.97	0.27	5.91	8.43
Path	0.37	54.44	3.57	1.55	4.33
Road	0.27	0.91	88.63	0.04	0
H-way	0.93	1.04	0.20	40.71	9.55
OP	79.66	59.68	29.63	94.78	

(c) HRNet with RVNet weights

Actual \ Predicted	Ditch	Path	Road	H-way	NC
Ditch	28.56	3.92	0.73	3.83	4.99
Path	0.59	60.35	4.64	0.64	4.17
Road	0.46	1.37	87.51	0.04	0
H-way	3.43	2.02	0.05	35.34	7.13
OP	81.57	72.42	36.78	94.49	

(d) HRNet with RVGan weights

Table 6.23.: Percentages for the intersection area of predicted instances by semantic segmentation models and the ground truth in the linear dataset. NC denotes percentage for the the area of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage for the area of predicted instances by the models that do not intersect any ground truth.

6.4.3. Stone Quarries Dataset

The stone quarry examples from the dataset explained in Section 4.2.3 are prepared in patches of size 384×384 pixels to train the semantic segmentation model shown in Figure 5.7 and evaluate the SSL pretraining impact. The number of examples and the area for training, validation and testing are listed in Table 6.24. The models are trained for 20 epochs and the F1-scores, precision and recall values on the test data are listed in Table 6.25. Consistent with the previous experiments, HRNet

outperforms DeepLabV3+ with the ResNet backbone, and SSL pretraining improves performance scores compared to random weight initialization.

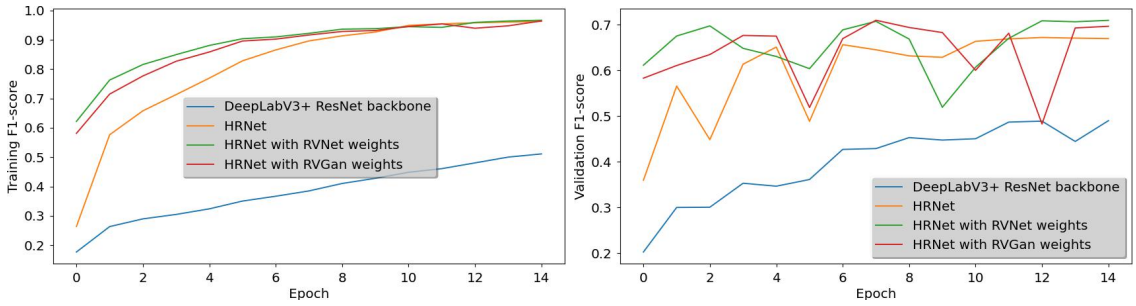
	No. examples	Min. area	Avg. area	Max. area	Total area
Training	2079	25.07 m^2	6415 m^2	3 km^2	13.34 km^2
Validation	414	32.05 m^2	6715 m^2	1.12 km^2	2.78 km^2
Testing	589	20.91 m^2	6616.67 m^2	0.63 km^2	3.90 km^2

Table 6.24.: The number of examples and area for training, validation and test in the stone quarries dataset.

Model	Pretrained weights	F1-score	Precision	Recall
HRNet	-	65.44	76.66	61.69
DeepLabV3+	-	52.39	57.01	56.36
HRNet	RVNet weights	**68.46**	**77.51**	65.67
HRNet	RVGan weights	67.08	72.42	**67.95**

Table 6.25.: Effect of SSL pretraining on semantic segmentation performance on the stone quarries dataset.

The plots for F1-scores by the models on training and validation sets are shown in Figure 6.9. Similar to the models' performances in the previous experiments, using the pretrained weights from the SSL pretext methods, i.e., RVNet and RVGan leads to faster convergence and higher scores.



(a) Training F1-score *(b) Validation F1-score*

Figure 6.9.: Training and validation F1-score plot for semantic segmentation on the quarries dataset.

The confusion matrices and the intersection area for prediction and ground truth examples listed in Tables 6.26 and 6.27 also indicate the superiority of HRNet to DeepLabV3+ with the ResNet backbone and the positive impact of SSL pretext methods.

	Quarries	NC
Quarries	89.30	10.7
OP	56.28	

(a) HRNet with random weights

	Quarries	NC
Quarries	98.3	1.7
OP	92.6	

(b) DeepLabV3+ with random weights

	Quarries	NC
Quarries	93.55	6.45
OP	53.78	

(c) HRNet with RVNet weights

	Quarries	NC
Quarries	94.74	5.26
OP	65.04	

(d) HRNet with RVGan weights

Table 6.26.: Confusion matrix for semantic segmentation on the stone quarries test set. Values show percentage for number of predicted instances that intersect those of the ground truth. NC denotes ground percentage for number of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage of predicted instances by the model that do not intersect any ground truth.

	Quarries	NC
Quarries	16.27	3.75
OP	9.19	

(a) HRNet with random weights

	Quarries	NC
Quarries	15.23	0.03
OP	16.46	

(b) DeepLabV3+ with random weights

	Quarries	NC
Quarries	18.15	2.07
OP	7.43	

(c) HRNet with RVNet weights

	Quarries	NC
Quarries	17.63	2.12
OP	10.76	

(d) HRNet with RVGan weights

Table 6.27.: Percentages for the intersection area of predicted and ground truth stone quarries. NC denotes ground percentage for number of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage of predicted instances by the model that do not intersect any ground truth.

6.5. Evaluation on 4 Test Regions with Distinct Objects

The quantitative evaluations in the previous sections are based on DTM patches cropped from the testing region making sure each patch includes an instance of the objects in each dataset. The predictions are made for each patch prepared in the test set and the results are evaluated. In this experiment, four large test regions are scanned in a sliding window fashion and labeled using Algorithm 2. Each of the 4 regions include examples for one of the categories such as bomb craters, charcoal kilns, burial mounds and mining holes. The predictions are made using the instance segmentation and semantic segmentation approaches and are evaluated quantitatively against the ground truth annotations. As mentioned in Chapter 4, the ground truth for the datasets used in the supervised tasks are not as precise, and are for simplicity approximated with circular polygons for example, even if the original shape of the objects and structures are not perfectly circular. Therefore, this experiment is conducted and three different colleagues are requested to manually annotate the same four test regions very precisely. Their annotations are also compared to the initial ground truth and the predictions by the deep learning approaches. The comparison gives a good impression of how well the predictions by the deep learning models are, even though the initial annotations were not as precise.

Algorithm 2 *The procedure for making predictions on a large test region.*

Require:
model$_{H\times W}$: a deep learning model trained with input size $H \times W$
DTM$_{\text{input}}$: Input DTM of size $P \times Q$

procedure PREDICT(model$_{H\times W}$, DTM$_{\text{input}}$)
  L $\leftarrow$ Zeros($P \times Q$) ▷ a matrix of zeros of shape ($P \times Q$) to store predictions
  for each item i in range(0, $P - H$, step $= H$) **do**
    for each item j in range(0, $Q - W$, step $= W$) **do**
      currentPatch $\leftarrow$ DTM$_{\text{input}}[i : i + H, j : j + W]$
      prediction $\leftarrow$ model$_{H\times W}$(currentPatch) ▷ Labels predicted for the current patch
      L$[n, i : i + H, j : j + W]$ = prediction
    end for
  end for
  return L ▷ Matrix of size P x Q holding predictions for the large region.
end procedure

Predictions for four regions by different deep learning models in the instance and semantic segmentation approaches are loaded into ArcGIS, and converted from raster to vector, i.e., polygons. The *Spatial Join* tool in ArcGIS is used to calculate the confusion matrices and the

percentage of intersection between predicted and ground truth examples. The results are listed in Tables 6.28-6.31. Additionally, the *Spatial Join* tool is used to calculate the same, i.e., confusion matrices and the percentage of intersection between manual annotations by the 3 colleagues and the ground truth examples. The results are listed in Tables 6.32 and 6.33.

The results for instance segmentation approaches shown in Tables 6.28 and 6.29 appear better at the first glance compared to the results by the semantic segmentation approaches shown in Tables 6.30 and 6.31. However, looking at the last rows in each table, it can be observed that the instance segmentation models make high over-predictions (OP) compared to the semantic segmentation models. In other words, higher amounts of background regions containing no objects are falsely categorized as the object instances by the instance segmentation models while very few amounts of background regions are falsely categorized as the objects by the semantic segmentation approaches.

Actual \ Predicted	BC	CK	BM	MH	NC
BC	79.59	4.08	4.08	6.12	6.12
CK	0.00	66.67	2.22	21.11	10.00
BM	5.66	5.66	88.68	0.00	0.00
MH	11.11	9.72	2.78	75.00	1.39
OP	71.10	92.98	93.23	96.94	

(a) HRNet backbone with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	69.39	10.20	0.00	14.29	6.12
CK	0.00	67.78	0.00	26.67	5.56
BM	0.00	11.32	83.02	3.77	1.89
MH	4.17	31.94	0.00	62.50	1.39
OP	70.63	94.60	81.28	98.04	

(b) ResNet backbone with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	87.76	2.04	0.00	4.08	6.12
CK	1.11	80.00	0.00	14.44	4.44
BM	1.89	11.32	81.13	1.89	3.77
MH	12.50	8.33	0.00	75.00	4.17
OP	69.66	94.21	87.54	97.01	

(c) HRNet backbone with RVNet weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	87.76	0.00	0.00	6.12	6.12
CK	1.11	83.33	0.00	6.67	8.89
BM	0.00	9.43	86.79	3.77	0.00
MH	15.28	5.56	0.00	75.00	4.17
OP	63.58	94.71	89.45	96.29	

(d) HRNet backbone with RVGan weights

Table 6.28.: Confusion matrix for instance segmentation results with Mask RCNN with different backbones and weight initialization on the 4 test regions using Algorithm 2. Values show percentage for number of predicted instances that intersect those of the ground truth. NC denotes percentage for number of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage of predicted instances by the models that do not intersect any ground truth.

Actual \ Predicted	BC	CK	BM	MH	NC
BC	71.30	6.06	3.49	78.31	5.08
CK	0	61.17	2.81	17.80	5.95
BM	0.72	2.39	97.04	0	0
MH	5.28	8.43	1.64	90.37	0.68
OP	46.55	87.11	68.33	83.32	

(a) HRNet backbone with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	53.83	34.36	0	76.22	5.08
CK	0	60.04	0	17.84	4.07
BM	0	4.51	94.94	0.60	1.19
MH	3.34	18.77	0	82.77	0.68
OP	45.17	90.27	63.54	85.94	

(b) ResNet backbone with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	78.33	6.99	0	92.54	5.08
CK	0.03	70.30	0	15.89	3.44
BM	0.45	5.81	94.63	0.07	3.84
MH	7.34	9.16	0	88.96	3.16
OP	48.55	89.58	68.98	80.12	

(c) HRNet backbone with RVNet weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	82.35	0	0	79.48	5.08
CK	0.01	65.64	0	8.58	7.37
BM	0	4.63	94.71	0.46	0
MH	9.90	5.86	0	85.16	3.16
OP	28.80	91.48	72.49	57.95	

(d) HRNet backbone with RVGan weights

Table 6.29.: Percentages for the intersection area of predicted and ground truth examples in the 4 test regions by the Mask RCNN for instance segmentation with different backbones and weight initializations using Algorithm 2. NC denotes percentage for the the area of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage for the area of predicted instances by the models that do not intersect any ground truth.

Actual \ Predicted	BC	CK	BM	MH	NC
BC	87.76	2.04	0.00	4.08	6.12
CK	0.00	70.00	0.00	0.00	30.00
BM	0.00	1.89	73.58	0.00	24.53
MH	4.17	1.39	0.00	63.89	30.56
OP	22.03	28.26	23.53	61.60	

(a) HRNet with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	24.49	0.00	6.12	59.18	10.20
CK	3.33	14.44	27.78	11.11	43.33
BM	3.33	14.44	27.78	11.11	43.33
MH	12.50	0.00	2.78	54.17	30.56
OP	77.14	78.69	47.89	88.39	

(b) DeepLabV3+ with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	67.35	0.00	0.00	22.45	10.20
CK	0.00	67.78	0.00	0.00	32.22
BM	0.00	0.00	71.70	0.00	28.30
MH	1.39	0.00	0.00	65.28	33.33
OP	49.25	63.69	38.71	68.48	

(c) HRNet with RVNet weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	87.76	2.04	0.00	4.08	6.12
CK	0.00	71.11	1.11	2.22	25.56
BM	0.00	0.00	73.58	0.00	26.42
MH	6.94	0.00	0.00	59.72	33.33
OP	18.64	41.96	36.51	72.99	

(d) HRNet with RVGan weights

Table 6.30.: Confusion matrix for semantic segmentation results on the 4 test regions using Algorithm 2. Values show percentage for number of predicted instances that intersect those of the ground truth. NC denotes percentage for number of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage of predicted instances by the models that do not intersect any ground truth.

Actual \ Predicted	BC	CK	BM	MH	NC
BC	70.59	0.74	0	6.43	6.34
CK	0	40.25	0	0	27.14
BM	0	0.13	68.09	0	24.23
MH	1.05	0.03	0	52.73	23.02
OP	6.96	15.04	7.98	39.28	

(a) HRNet with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	14.23	0	1.23	43.32	8.66
CK	0.67	5.15	7.36	2.17	40.43
BM	0	0	73.76	0	16.50
MH	8.37	0	0.83	45.49	22.14
OP	49.04	72.71	21.15	64.95	

(b) DeepLabV3+ with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	48.27	0	0	26.66	8.96
CK	0	39.95	0	0	23.38
BM	0	0	59.72	0	23.75
MH	0.65	0	0	53.27	26.50
OP	6.80	35.43	10.27	38.98	

(c) HRNet with RVNet weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	80.14	0.14	0	6.14	4.24
CK	0	41.30	0.02	0.40	20.48
BM	0	0	61.89	0	22.36
MH	2.11	0	0	54.29	28.56
OP	4.51	19.10	6.40	50.00	

(d) HRNet with RVGan weights

Table 6.31.: Percentages for the intersection area of predicted and ground truth examples in the 4 test regions by the semantic segmentation approaches using Algorithm 2. NC denotes percentage for the the area of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage for the area of predicted instances by the models that do not intersect any ground truth.

The results for both: semantic and instance segmentation models, are worse than those of the manual annotations by the three colleagues shown in Tables 6.32 and 6.33. However, the colleagues had the advantage of having a prior knowledge of which of the 4 separate regions contain which of the 4 structures, i.e., bomb craters, charcoal kilns, burial mounds and mining holes. Therefore, the only error in their annotations are either false annotations of background regions as the mentioned categories or missing annotations for some instances. There are no errors of confusing one category with the other. For charcoal kilns, the models even correctly predict and recover higher number of examples compared to the colleagues 2 and 3.

Among the annotations by the colleagues, the highest scores are in general achieved by colleague 1. Among individual categories, the scores are alternating which indicates that even in manual annotations by humans, there are discrepancies even though they had the advantage of prior knowledge about which regions contain which of the structures. The percentage of

Actual \ Predicted	BC	CK	BM	MH	NC
BC	100.0	0.00	0.00	0.0	0.00
CK	0.0	96.67	0.00	0.0	3.33
BM	0.0	0.00	92.45	0.0	7.55
MH	0.0	0.00	0.00	87.5	12.50
OP	2.0	32.03	9.26	70.0	

(a) Colleague 1

Actual \ Predicted	BC	CK	BM	MH	NC
BC	100.00	0.00	0.00	0.00	0.00
CK	0.00	52.22	0.00	0.00	47.78
BM	0.00	0.00	83.02	0.00	16.98
MH	0.00	0.00	0.00	94.44	5.56
OP	5.77	6.0	0	56.69	

(b) Colleague 2

Actual \ Predicted	BC	CK	BM	MH	NC
BC	100.00	0.00	0.00	0.00	0.00
CK	0.00	60.00	0.00	0.00	40.00
BM	0.00	0.00	81.13	0.00	18.87
MH	0.00	0.00	0.00	90.28	9.72
OP	3.92	50.91	14.0	58.86	

(c) Colleague 3

Table 6.32.: Confusion matrix for manual annotations by 3 colleagues compared to the initial ground truth. Values show percentage for the intersection area of predicted instances and the ground truth. NC denotes percentage for the the area of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage for the area of predicted instances by the models that do not intersect any ground truth.

Actual \ Predicted	BC	CK	BM	MH	NC
BC	97.94	0	0	0	0
CK	0	60.24	0	0	1.7
BM	0	0	91.54	0	7.24
MH	0	0	0	73.28	15.38
OP	1.63	27.10	12.82	49.90	

(a) Colleague 1

Actual \ Predicted	BC	CK	BM	MH	NC
BC	100	0	0	0	0
CK	0	25.54	0	0	43.50
BM	0	0	83.99	0	15.67
MH	0	0	0	90.40	8.09
OP	4.55	4.24	0	39.68	

(b) Colleague 2

Actual \ Predicted	BC	CK	BM	MH	NC
BC	100	0	0	0	0
CK	0	63.97	0	0	33.06
BM	0	0	82.39	0	16.59
MH	0	0	0	84.00	11.86
OP	3.11	3.81	15.91	39.31	

(c) Colleague 3

Table 6.33.: Percentages of the intersection area of manual annotations by the 3 colleagues and the initial ground truth. NC denotes percentage for the the area of ground truth examples not captured by the models. OP stands for over-prediction, i.e., percentage for the area of predicted instances by the models that do not intersect any ground truth.

over-predictions for some of the categories, e.g., charcoal kilns and mining holes, by the colleagues are not far behind the over-predictions by the semantic segmentation models.

Since the new manual annotation by colleague 1 matches the original ground truth the best, the predictions by the model are also compared against the annotations by colleague 1 below.

Model Predictions Evaluated Against Annotations by Colleague 1

Manual annotations for the 4 test regions by colleague 1 matched the ground truth annotations the best among other colleagues, as shown previously. This is also confirmed by the visual evaluations in the following section as the annotations by colleague 1 are more detailed and precise. Therefore, the predictions by the deep learning approaches are also compared to the annotations by colleague 1. Tables 6.34 and 6.35 show the confusion matrices and percentages of intersection area for the

Actual \ Predicted	BC	CK	BM	MH	NC
BC	79.25	3.77	3.77	7.55	5.66
CK	0.00	60.94	5.47	19.53	14.06
BM	5.56	3.70	88.89	0.00	1.85
MH	8.76	5.99	3.69	70.51	11.06
OP	63.43	90.71	91.71	92.66	

(a) HRNet backbone with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	67.92	11.32	0.00	15.09	5.66
CK	0.00	64.84	0.00	22.66	12.50
BM	1.85	11.11	79.63	5.56	1.85
MH	2.30	17.97	0.92	66.82	11.98
OP	66.93	92.42	81.17	95.34	

(b) ResNet backbone with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	83.02	3.77	0.00	7.55	5.66
CK	0.78	76.56	0.00	10.94	11.72
BM	0.00	16.67	77.78	1.85	3.70
MH	8.76	9.68	0.00	72.35	9.22
OP	63.84	91.22	87.76	92.45	

(c) HRNet backbone with RVNet weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	84.91	0.00	0.00	9.43	5.66
CK	0.00	72.66	0.00	6.25	21.09
BM	0.00	3.70	88.89	3.70	3.70
MH	11.52	3.69	0.00	73.73	11.06
OP	54.55	93.52	88.91	90.03	

(d) HRNet backbone with RVGan weights

Table 6.34.: Confusion matrix for instance segmentation results with Mask RCNN with different backbones and weight initialization on the 4 test regions using Algorithm 2. Values show percentage for number of predicted instances that intersect those labeled by colleague 1. NC denotes percentage for number of examples by colleague 1 not captured by the models. OP stands for over-prediction, i.e., percentage of predicted instances by the models that do not intersect any annotation by colleague 1.

Actual \ Predicted	BC	CK	BM	MH	NC
BC	62.53	5.42	3.53	70.87	5.84
CK	0	63.46	4.76	16.17	11.79
BM	0.70	3.31	85.43	0	1.74
MH	5.75	3.51	3.17	81.84	4.02
OP	42.88	83.66	77.52	72.00	

(a) HRNet backbone with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	45.78	34.57	0	66.11	5.08
CK	0	64.45	0	18.15	10.89
BM	0.19	5.52	78.14	1.24	2.92
MH	2.69	16.04	0.31	76.57	4.82
OP	43.12	88.15	65.37	76.24	

(b) ResNet backbone with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	67.14	7.40	0	85.60	5.84
CK	0.03	74.50	0	15.01	9.25
BM	0	9.68	77.91	0.46	4.44
MH	5.57	5.34	0	85.30	6.52
OP	46.54	86.76	69.77	75.71	

(c) HRNet backbone with RVNet weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	71.99	0	0	72.74	5.84
CK	0	63.99	0	7.19	15.65
BM	0	2.53	79.38	0.65	4.66
MH	7.82	1.23	0	81.19	7.16
OP	24.86	89.22	72.96	55.05	

(d) HRNet backbone with RVGan weights

Table 6.35.: Percentages for the intersection area of predicted examples in the 4 test regions by Mask RCNN with different backbones and weight initializations using Algorithm 2, and annotations by colleague 1 in the 4 test regions. NC denotes percentage for the the area of examples by colleague 1 not captured by the models. OP stands for over-prediction, i.e., percentage for the area of predicted instances by the models that do not intersect any annotation by colleague 1.

predictions by the instance segmentation models and the annotations by colleague 1. Tables 6.36 and 6.37 show the same for the semantic segmentation models.

As seen in Table 6.28 (instance segmentation vs. initial ground truth) and Table 6.34 (instance segmentation vs. new annotations by colleague 1), there is a stronger match between the predictions by the instance segmentation approaches and the annotations by colleague 1 compared to the match between the said predictions and the initial ground truth. This confirms that even though the models were trained with less precise ground truth annotations, they still learn to properly detect objects. With better and more precise ground truth, they are expected to perform even better. The same is confirmed by the percentages of intersection are between the predictions by instance segmentation models and the initial ground truth (Table 6.29) and the new annotations by colleague 1 (Table 6.35).

Actual \ Predicted	BC	CK	BM	MH	NC
BC	84.91	1.89	0.00	5.66	7.55
CK	0.00	52.34	0.00	0.00	47.66
BM	0.00	1.85	75.93	0.00	22.22
MH	4.61	0.46	0.00	41.94	53.00
OP	8.33	24.73	19.61	31.88	

(a) HRNet with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	22.64	0.00	5.66	62.26	9.43
CK	2.34	12.50	19.53	7.81	57.81
BM	0.00	0.00	81.48	0.00	18.52
MH	9.68	0.00	0.92	42.40	47.00
OP	66.04	74.19	47.52	80.32	

(b) DeepLabV3+ with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	66.04	0.00	0.00	22.64	11.32
CK	0.00	51.56	0.00	0.00	48.44
BM	0.00	0.00	70.37	0.00	29.63
MH	2.30	0.46	0.00	43.32	53.92
OP	38.46	60.82	38.71	43.01	

(c) HRNet with RVNet weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	84.91	1.89	0.00	5.66	7.55
CK	0.00	54.69	0.78	1.56	42.97
BM	0.00	1.85	72.22	0.00	25.93
MH	4.15	0.00	0.00	47.00	48.85
OP	11.48	37.39	37.50	39.89	

(d) HRNet with RVGan weights

Table 6.36.: Confusion matrix for semantic segmentation results on the 4 test regions using Algorithm 2. Values show percentage for number of predicted instances that intersect those annotated by colleague 1. NC denotes percentage for number of examples by colleague 1 not captured by the models. OP stands for over-prediction, i.e., percentage of predicted instances by the models that do not intersect any annotation by colleague 1.

Actual \ Predicted	BC	CK	BM	MH	NC
BC	61.5	0.62	0	5.46	8.11
CK	0	41.92	0	0	42.79
BM	0	0.09	56.28	0	24.72
MH	1.78	0.02	0	44.85	35.97
OP	4.31	9.93	5.59	11.47	

(a) HRNet with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	11.97	0	1.42	37.12	9.00
CK	0.75	4.89	7.10	2.56	54.29
BM	0	0	59.83	0	23.87
MH	6.19	0	0.37	37.88	36.14
OP	42.58	70.25	19.93	56.53	

(b) DeepLabV3+ with random weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	41.40	0	0	22.78	11.03
CK	0	42.57	0	0	42.30
BM	0	0	40.55	0	30.54
MH	1.26	0.12	0	42.02	40.23
OP	4.95	28.6	11.59	21.37	

(c) HRNet with RVNet weights

Actual \ Predicted	BC	CK	BM	MH	NC
BC	68.25	0.12	0	5.33	6.38
CK	0	43.76	0.02	0.35	35.58
BM	0	0.19	43.94	0	25.0
MH	2.03	0	0	44.67	35.28
OP	3.62	12.90	6.21	23.68	

(d) HRNet with RVGan weights

Table 6.37.: Percentages for the intersection area of predicted examples by semantic segmentation using Algorithm 2 and annotations by colleague 1 in the 4 test regions. NC denotes percentage for the the area of examples by colleague 1 not captured by the models. OP stands for over-prediction, i.e., percentage for the area of predicted instances by the models that do not intersect any annotation by colleague 1.

Similarly, the confusion matrix in Table 6.30 compared to Table 6.36 and the table for percentages of intersection area in Table 6.31 compared to Table 6.37 show that the match between predictions by the semantic segmentation models are also matching the new precise annotations by colleague 1 stronger than the initial approximate ground truth annotations. This stronger match is despite the fact the models were trained with the initial imprecise ground truths, further confirming the capacities of the uses approaches. Trained with better, more precise ground truth annotations, the models are expected to produce even better outcomes.

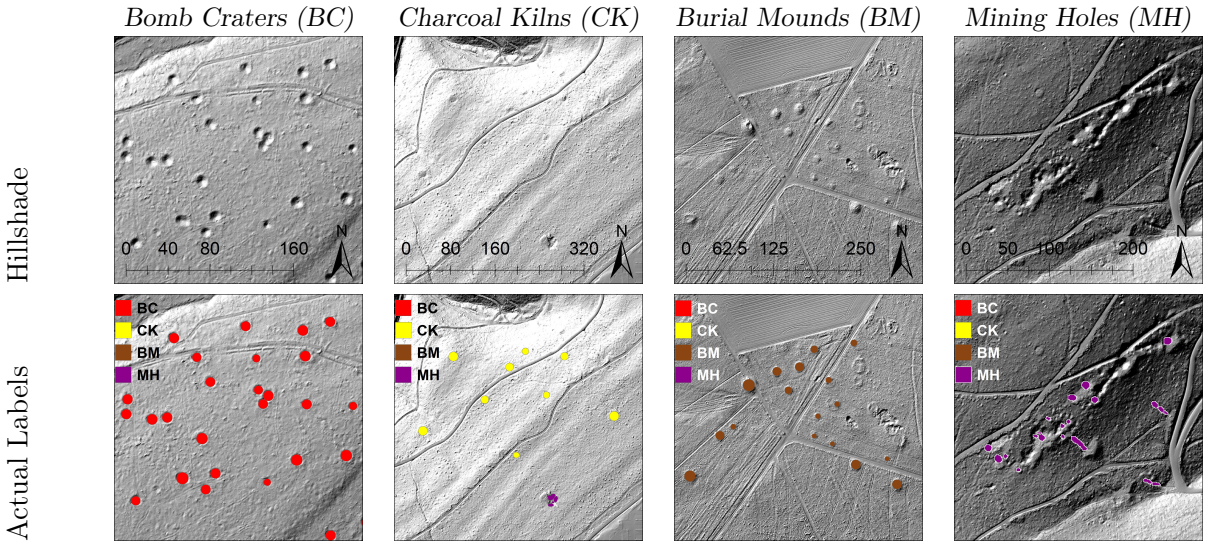
6.6. Qualitative Evaluations

This section contains qualitative evaluation results for the experiments using the three different approaches, i.e., classification, instance segmentation and semantic segmentation on three datasets.

Section 6.6.1 contains example predictions by for the areal dataset, it also compares the predictions by deep learning models to the manual annotations by three separate people. Sections 6.6.2 and 6.6.3 includes the same for the linear and stone quarries datasets.

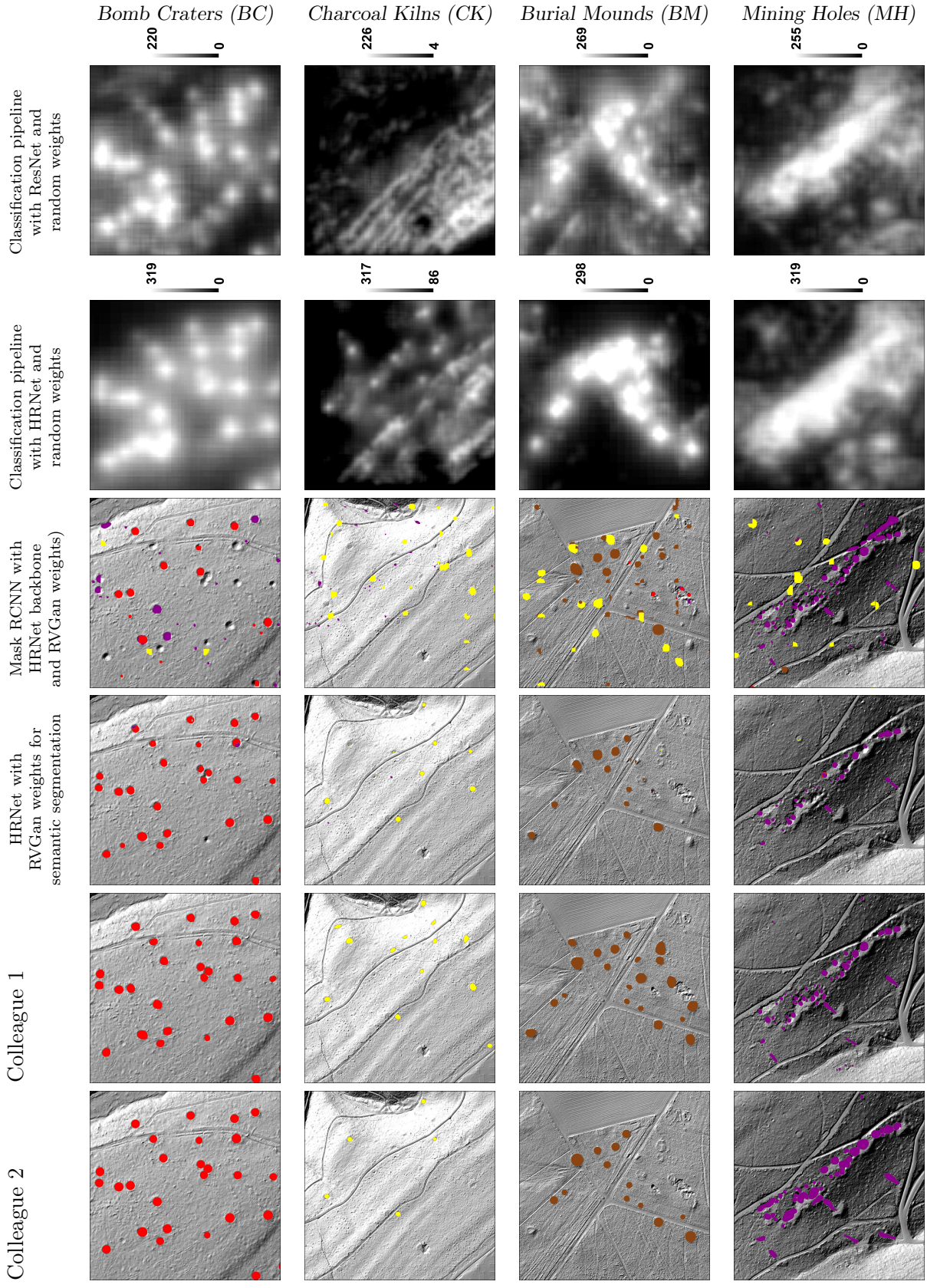
6.6.1. Qualitative Results for Areal Dataset

Three different methodologies, i.e., classification, instance segmentation and semantic segmentation are used to detect archaeological monuments and man-made terrain structures in DTM data. Each method is trained from scratch with random weights and also finetuned with the pretrained weights of RVNet and RVGan from the SSL pretext method. The best results for each method are illustrated side by side for four different regions including bomb craters (BC), charcoal kilns (CK), burial mounds (BM) and mining holes (MH) in Table 6.38. Further qualitative results showing predictions by models trained with random weight initialization and the ones finetuned with pretrained weights for each approach are shown in Appendix A. Additionally, the manual annotations by three different people are also shown for comparison. As shown in Table 6.38, predictions by the classification pipeline are in terms of heatmaps with a rough indication of regions containing structures of interest. For instance segmentation, the outputs are rectangular bounding boxes, segmentation masks and class labels for each instance in the input. However, for clarity, predictions are shown as segmentation maps, and the rectangular bounding box can be inferred using the segmentation maps. For semantic segmentation, the outputs are already multi-class segmentation maps and are visualized as such. For all the regions and objects shown in Table 6.38, the semantic segmentation approach produced more accurate predictions. The visualizations are consistent with the quantitative results in Tables 6.28-6.31 as there are more false predictions by the instance segmentation model, i.e., Mask RCNN with RVGan weights (row 5 in Table 6.38) for bomb craters, burial mounds and mining holes. Additionally, the amount of over-predictions by the instance segmentation model is very high. For semantic segmentation with RVGan weights, the number of false positives are very low, and specially invisible in these regions (row 6). There are also no visible over-predictions by the semantic segmentation model. While the instance segmentation results are clearly worse than the manual annotations by the colleagues, the semantic segmentation predictions are comparable. Bomb craters, charcoal kilns and mining holes are mostly recovered well. There is just an apparent under-prediction for the burial mounds. Overall, the semantic segmentation approach produces better results than the classification and instance segmentation approach.



Continued on next page

Table 6.38 – continued from previous page



Continued on next page

Table 6.38 – continued from previous page

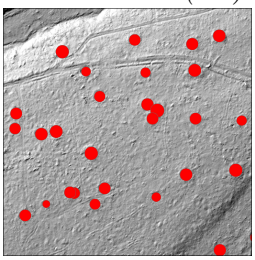
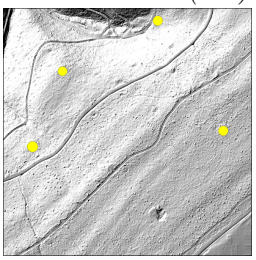
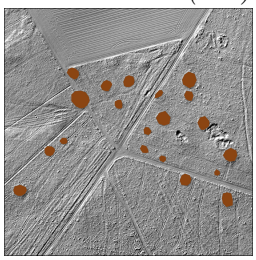
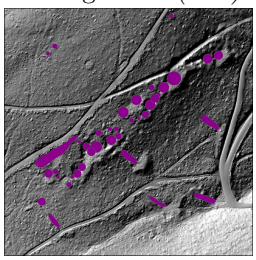
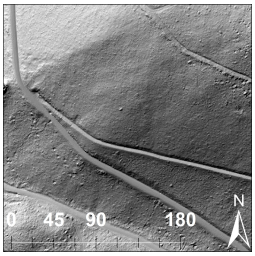
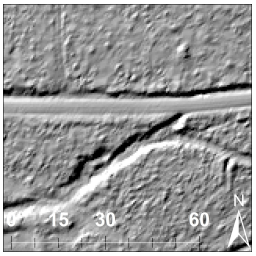
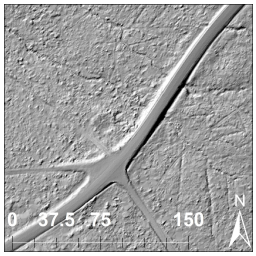
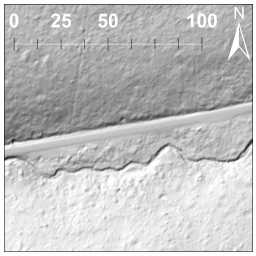
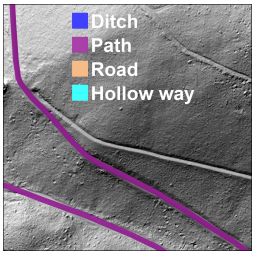
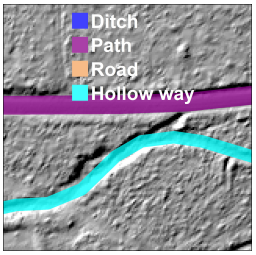
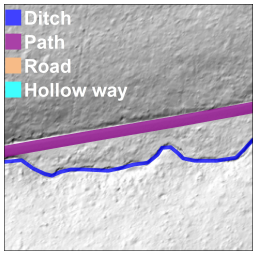
	Bomb Craters (BC)	*Charcoal Kilns (CK)*	*Burial Mounds (BM)*	*Mining Holes (MH)*
Colleague 3				

Table 6.38.: Predictions for 4 *regions by the best trained model in three approaches and manual annotations by* 3 *colleagues. The third and fourth rows contain predicted heatmaps for the corresponding class by HRNet and ResNet classifiers using the inference pipeline shown in Figure 5.5. The fourth and fifth rows are predictions by the instance segmentation and semantic segmentation (SS) approaches using* RVGan weights.

6.6.2. Qualitative Results for the Linear Dataset

Experiments on the linear dataset are conducted using instance and semantic segmentation approaches. For both methods, the model finetuned with the pretrained weights from the RVGan model achieved the top performance scores on the test set. Therefore, qualitative results for these two approaches using the pretrained weights of RVGan on 4 regions are shown in Table 6.39. Similar to the performance on the areal dataset, the semantic segmentation approach produces the better results, specially for thin linear structures such as ditches, as shown in the 4^{th} region in Table 6.39. These results are also consistent with the quantitative results in Tables 6.15-6.23. For most of the categories and examples, the semantic segmentation model makes correct predictions and does not miss. For example, instances of paths roads and ditches are correctly captured, but there are more over-predictions by the semantic segmentation model. The hollow way in Region 2 is better predicted by the instance segmentation model, but the rest of the examples are not captured well. However, the instance segmentation model makes less over-predictions as confirmed by the quantitative results in Tables 6.15 and 6.16. Further qualitative results showing predictions by models trained with random weight initialization and the ones finetuned with pretrained weights for each approach are shown in Appendix A.

	Region 1	Region 2	Region 3	Region 4
Hillshade				
Actual Labels				

Continued on next page

Table 6.39 – continued from previous page

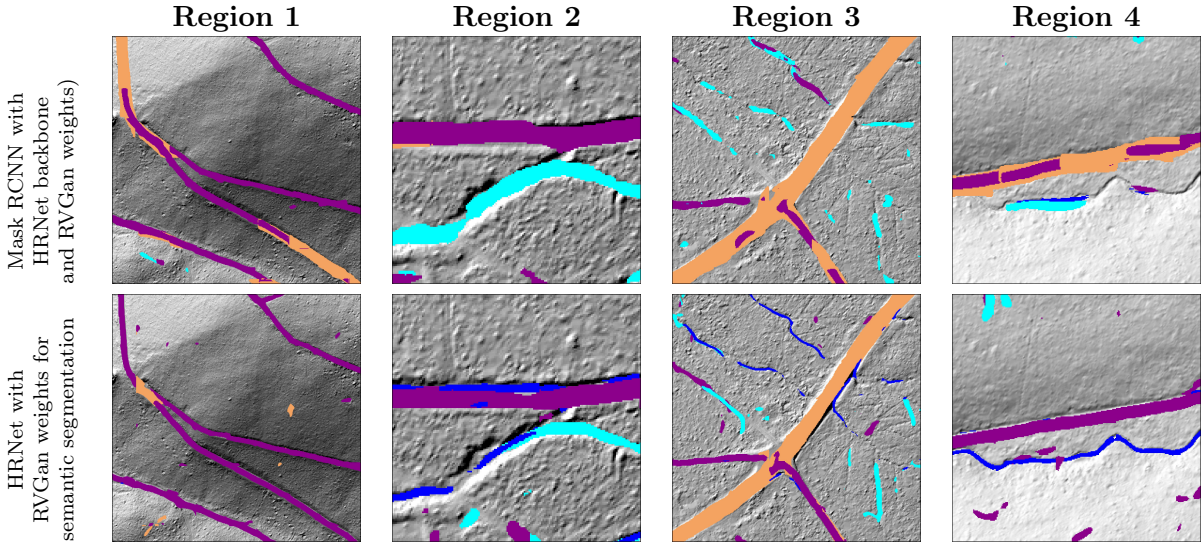
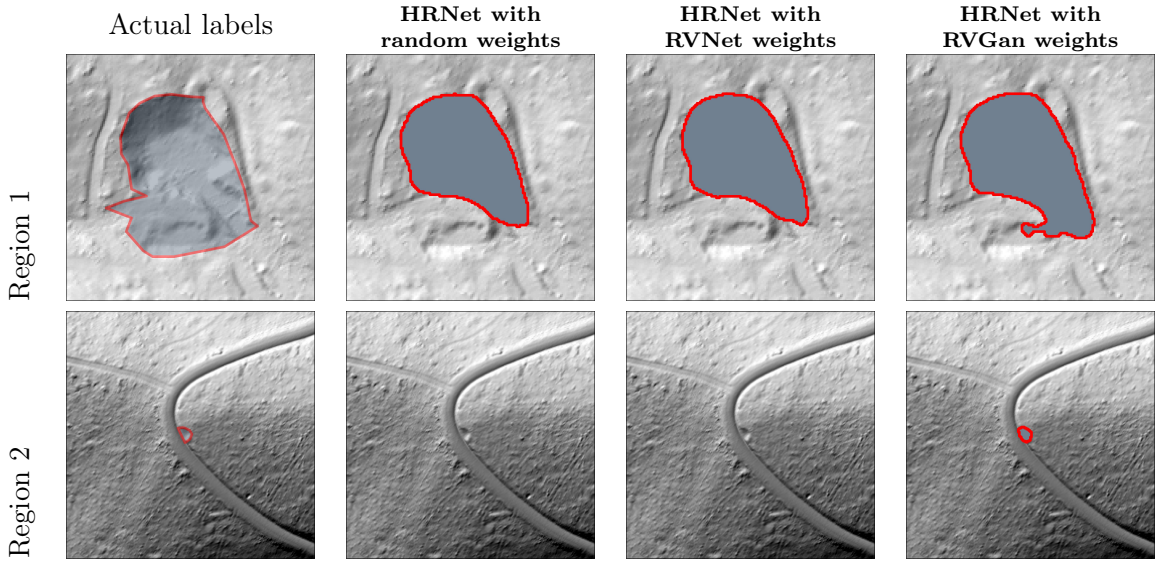


Table 6.39.: Predictions for 4 regions by the best trained model in two approaches, i.e., semantic and instance segmentation, on the test dataset containing linear structures..

6.6.3. Qualitative Results for Stone Quarries Dataset

Experiments on the stone quarries dataset are conducted only using the semantic segmentation approach. Similar to the other two datasets, on this dataset as well, the semantic segmentation approach shown in Figure 5.7 is trained with random weight initialization and also pretrained weights from the RVNet and RVGan models from the SSL pretext phase. Example predictions by the models are shown in Table 6.40. Results by the DeepLabV3+ with ResNet backbone are included in Appendix A. The results reflect the quantitative evaluation in Section 6.4.3 (specifically, Tables 6.26 and 6.27) and show that SSL pretraining improves performance score. Moreover, for smaller examples of stone quarries, the HRNet model with RVGan weights works better than the same with RVNet weights and random weights. For big examples that do not completely fit in the input patch of 384×384 pixels, the results are not as clean as expected and the effect is obvious in the predictions by the 8^{th} region in Table 6.40.



Continued on next page

Table 6.40 – continued from previous page

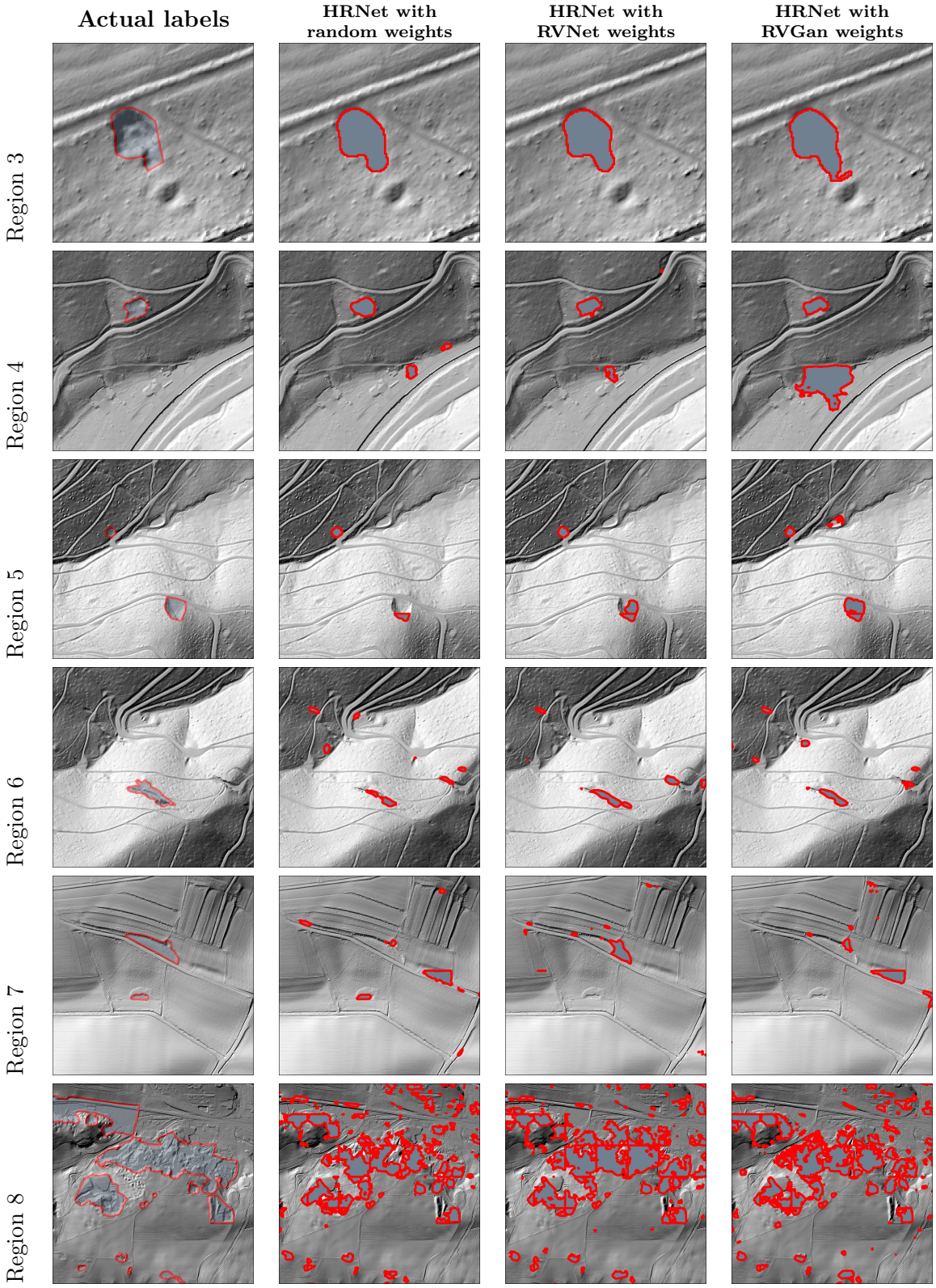


Table 6.40.: Stone quarry predictions for 4 regions by the HRNet model trained with random weights, RVNet weights, and RVGan weights.

6.7. Summary

Two methods, RVNet and RVGan, are trained in the pretext phase in SSL to learn generating relief rasters for a given input DTM patch. The weights of the trained models are used in the second phase, i.e., downstream to finetune deep learning models for supervised tasks such as classification, instance segmentation and semantic segmentation. Experiments on the supervised tasks are conducted on the three datasets explained in Chapter 4. Deep learning models used in the supervised downstream tasks include HRNet, ResNet, Mask RCNN and DeepLabV3+. A short overview reporting which methods are suitable for which tasks is listed in Table 6.41. In general, the most suitable approach is the semantic segmentation approach as it makes better predictions (backed by the qualitative and quantitative evaluations detailed previously). HRNet is the better model compared to ResNet and DeepLabV3+, and SSL pretext has a significantly positive impact on the performance scores as evidenced by the MIOU and MAP scores in the semantic segmentation and instance segmentation approaches, respectively. Additionally, all the models using the pretrained weights from the SSL pretext methods, i.e., RVNet and RVGan converge faster that those initialized with random weights, as observed in the plots for training and validation.

Model	**Backbone**	**Weights**	**Classification**	**Instance Segmentation**	**Semantic Segmentation**
HRNet	-	-	+++	-	+
HRNet	-	RVNet	++	-	++
HRNet	-	RVGan	+	-	+++
ResNet	-	-	o	-	-
DeepLabV3+	ResNet	-	-	-	o
Mask RCNN	ResNet	-	-	+	-
Mask RCNN	HRNet	-	-	+	-
Mask RCNN	HRNet	RVNet	-	++	-
Mask RCNN	HRNet	RVGan	-	+++	-

Table 6.41.: Summary of results showing which methods are suitable for which task. The character '-' indicates that the method/backbone/weights is not used. The suitability starts from worst ('o') to best ('+++').

7. Discussions and Conclusions

The final chapter in this dissertation includes discussions of the methodologies, datasets and evaluation results. It links the conducted experiments and evaluation results to the research hypotheses mentioned in the introduction in Chapter 1. Section 7.1 discusses DL methods used for the pretext tasks and their impact on downstream supervised tasks, i.e., classification, instance segmentation and semantic segmentation. Section 7.2 gives a summary of the dissertation and points out further research directions.

7.1. Discussions

This section gives assessments of approaches used for pretext and downstream tasks in two separate subsections that follow.

7.1.1. Assessment of Pretext Methods

The main idea behind this research is the use of SSL pretraining to improve detection results for archaeological monuments and man-made terrain structures in LiDAR data with the help of supervised DL models. The two pretext methods investigated are encoder-decoder architectures (RVNet) and GANs (RVGan). Both approaches make use of a well-known DL model, i.e., HRNet as their core backbone architecture.

Both pretext methods are designed to train models that learn to generate relief visualization rasters for given DTM inputs. It is hoped that by doing so, the model learns important features in the DTMs (Hypothesis 1) and aids supervised learning models in detection of archaeological monuments and man-made terrain structures (Hypothesis 2).

The experiments and results in Section 6.1 show that relief visualization rasters are good supervision signals that can be used to train deep learning models when there are huge amounts of unlabeled DTM data. They are automatically calculated with tools such as RVT and help tune the parameters of deep learning models. Training deep learning models with such rasters help learn intrinsic characteristics of DTM inputs. The relief rasters generated by the RVNet and RVGan in Table 6.3 confirm that deep learning models learned the properties of DTM inputs well enough to be able to automatically generate the corresponding relief rasters.

While RVNet achieves better scores (Table 6.2) and produces relief visualizations of higher quality (Table 6.3) compared to those by RVGan, the impact of pretraining with RVGan is more prominent on the supervised downstream tasks, specially instance segmentation and semantic segmentation as discussed in Chapter 6. Additionally, pretrained weights of SSL pretext methods help the supervised models in the downstream tasks converge faster, as shown in the training and validation plots in Chapter 6.

In summary, the experiments and results are inline with the assumptions in Hypotheses 1 and 2. In other words, relief visualization rasters are good supervision signals for deep learning with unlabeled DTM data and SSL pretraining with such datasets helps the supervised downstream tasks.

7.1.2. Assessment of Downstream Methods

The first downstream task investigated is a classification approach. Classifiers are trained to categorize input DTMs into 5 classes: bomb craters, charcoal kilns, burial mounds, mining holes and background. Since the label predicted by a classifier is a single integer denoting the class, assigning the predicted label to a squared grid in the region of interest would result into blobs of DTMs with imprecise labels. Morover, objects may appear with different sizes and also at different scales. To account for this, 5 classifiers are trained with inputs of height and width equal to 32, 64, 96, 128 and 224 pixels each. The trained models are then used to scan large DTMs in a sliding window fashion, moving left to right and top to bottom with predefined strides, cropping DTM patches and making predictions. Each classifier ends up with a heatmap of the region for each category. The resulting heatmaps by all the models are summed for each category and used as the final map.

The precision and quality of the final heatmaps depend on the amount of overlap, i.e., the value for the strides during scanning of the region. The smaller the stride is, the more precise and detailed the final heatmaps are. However, the computation time at test time also increases proportionally with the value of the stride. For more precise and crisp predictions, and in order to avoid the overhead in computation time, instance segmentation techniques are applied. This is the main idea behind Hypothesis 3. While the classification approach is good for a rough indication of regions containing objects/structures interest, instance segmentation models provide crisp outlines of objects and structures in the input DTM and assigns class labels and bounding box coordinates for each of them. Moreover, at test time, there is no need to crop overlapping input DTM patches to make predictions. Thus, more accurate predictions are achieved in a smaller period of time. The assumption that instance segmentation is a more suitable approach is confirmed with the qualitative results shown in Table 6.38. The predictions by the classifiers are in terms of heatmaps and show roughly where instances of objects probably exist. Instance segmentation predictions are exact outline of each structure.

There are two issues with instance segmentation frameworks. If an object is elongated linearly throughout the input patch, especially if it lies diagonally, the predicted rectangular bounding box is the rectangle covering the whole input region, which is not a helpful information. Additionally, in region-based instance segmentation frameworks (used in this dissertation), the region proposal network proposes many regions with different sizes where objects of interest could exist. This could lead to multiple proposed regions with different sizes and confidence scores for the same object. It is therefore required to set a threshold for the number of proposed regions and an additional threshold for the confidence scores at prediction time. Setting these threshold may lead to overestimation or underestimation causing false positives or negatives. To alleviate such problems, a third option, semantic segmentation is explored. This is the main idea behind Hypothesis 4.

Semantic segmentation model provides pixel-wise class labels for a given DTM patch. Therefore, each pixel is assigned to one and only one category regardless of the shape and size of objects in the input. Similar to instance segmentation models, semantic segmentation models also scan regions of interest much faster than classifiers and produce more accurate and fine-grained results. The assumption that semantic segmentation is more suitable and specially predicts thin and linearly elongated structures well is confirmed by the results shown in Table 6.39. For example, the ditch in Region 4 is recovered by the semantic segmentation model while the instance segmentation model does not predict it.

In summary, the semantic segmentation approach is found to be the most appropriate method for this task.

7.1.3. Assessment of Selected Core Deep Learning Architectures

A well-known architecture, HRNet (Sun et al., 2019; Wang et al., 2020), is selected for all the methods such as RVNet, RVGan, classification, instance segmentation and semantic segmentation in this research. The choice is based on its superior results compared to another well-known model called ResNet (He et al., 2016). ResNet is a deep learning architecture that can be trained as a classifier, but it can also be incorporated as a feature extractor in instance and semantic segmentation frameworks. In this research, ResNet is used for the classification pipeline, the backbone of Mask RCNN (He et al., 2017) for instance segmentation and the feature extractor in DeepLabV3+ (Chen et al., 2018) for semantic segmentation. The results are compared with the HRNet model and since the HRNet scores better in general, it is used as the main architecture for the SSL pretext frameworks, i.e., RVNet and RVGan.

7.1.4. Assessment of Predictions for each Category

In general, the models predict structures such as charcoal kilns and burial mounds well, e.g., with an IOU score of 65.05 and 68.37 percent with the HRNet model with RVNet weights in semantic segmentation. They have more distinct characteristics and structures, and they are in average bigger in size than bomb craters and mining holes (see Table 4.1). Bomb craters and mining holes are smaller in size compared to burial mounds and charcoal kilns and these two structures are very similar to each other in terms of shape and structure. These properties impact the predictions by the deep learning models. For example, the IOU scores for the HRNet model with RVNet weights in semantic segmentation on bomb craters and mining holes are 42.23 and 41.03 percent, respectively. It is also reflected in the confusion matrices in Table 6.19. Bomb craters are mostly confused for mining holes compared to burial mounds and charcoal kilns, and mining holes are mostly confused for bomb craters than the rest. Additionally, the models seem to miss, i.e., not recover in the predictions, a higher percentage of bomb craters and mining holes while the prediction rate for charcoal kilns and burial mounds are high.

On the linear dataset, for wider, mostly straight structures such as roads and paths, the predictions are better while for thin structures such as ditches or hollow ways, the predictions are worse. For example the IOU scores for paths and roads are 47.78 and 81.91 percent, respectively, while the scores for ditches and hollow ways are 17.52 and 26.57 percent, respectively, by the HRNet model with RVGan weights in semantic segmentation. Roads are mostly confused for paths, which is quite intuitive as the two structures look quite similar specially in the small streets. Examples of ditches are mostly confused with paths, and hollow ways are mostly confused for ditches. Finally the higher percentage of paths are confused for roads, as seen in the confusion matrices in Table 6.22.

The stone quarries dataset is framed to contain binary labels, i.e., quarry vs. background. While the recovery rate for the stone quarries is 94.74 percent by the HRNet model with RVGan weights, and only 5.24 percent of ground truth examples are missed, the over-prediction (percentage of predicted quarries that do not intersect any ground truth) is quite high (65.04 percent) as shown in Table 6.26. The percentage of over-predictions by the HRNet model with RVNet is lower (better) than the rest of the models, which is in turn consistent with the high $F1$ and precision scores shown in Table 6.25.

7.2. Summary and Outlook

This research was aimed at automated detection of structures related to historical mining and archaeology in ALS or LiDAR data. Structures and objects of interest include archaeological

monuments such as charcoal kilns, burial mounds, mining holes, ditches, hollow ways and stone quarries and man-made structures such as bomb craters, paths and roads in the Harz region, Lower Saxony. To this end, deep learning techniques, the emerging methodology in many research areas, are exploited. Due to their intrinsic dependence on a lot of annotated data and the lack of such data in this research, Self Supervised Learning (SSL) methods are explored. As the first step in SSL termed *pretext*, two methods are pretrained on unlabeled DTMs created from ALS data. The first method is an encoder-decoder approach, called RVNet, that learns generating relief visualization rasters for DTM inputs. The second is based on GANs, called RVGan, and is trained to do the same.

The learned parameters from the pretrained models are then used in the second step in SSL termed the *downstream*. Downstream is the task of training a supervised learning model on annotated dataset with the model initialized using parameters of pretrained models from the pretext step. Three different downstream tasks, i.e., classification, instance segmentation and semantic segmentation are experimented with in this research. The results of the experiment confirm the positive impact of pretraining with unlabeled DTM on the performance of supervised learning models with annotated data. It is also shown that initializing the models in the supervised downstream tasks with the weights of SSL pretext methods lead to faster convergence.

Two well-known deep learning architectures: HRNet and ResNet (or DeepLabV3+ and Mask RCNN with the ResNet backbone), are trained with random weight initialization for each supervised downstream task. The model with higher scores in each task, i.e., HRNet, is then selected as the main backbone for the pretext methods, namely RVNet and RVGan. The positive impact of SSL pretraining is confirmed with the increase in MAP scores for instance segmentation experiments. The MAP scores increased from 53.25 to 58.38 in the areal dataset and from 48.77 to 54.95 in the linear dataset. The MIOU scores in semantic segmentation experiments also confirm the positive impact of SSL pretraining. There is an increase from 61.49 to 66.21, 52.24 to 53.46, and from 65.44 to 68.46 in the areal, linear and stone quarries dataset, respectively.

In summary, both the pretext methods: the GAN-based model, i.e, RVGan and the encoder-decoder based model, i.e., RVNet, have positive impacts on the detection scores of supervised tasks. The impact of RVGan, however, is more significant and it is therefore the recommended pretext approach in the future. Compared to the classification and instance segmentation approaches in the supervised downstream tasks, the semantic segmentation approach performs better rendering it as the most suitable approach in the future.

Even though the use of unlabeled data in training the pretext models help achieve better performance in the downstream tasks, the performance is still limited and highly depends on the quality (and also the amount) of annotated data. Furthermore, archaeological structures come in different sizes and shapes, and the input size with which to train deep learning models has a great impact on the performance. While a model trained with a small input size, e.g., 224×224, is able to detect small structures such as mining holes, charcoal kilns and bomb craters, it performs poorly in detecting larger structures such as historical stone quarries, and vice versa. Further research is required to conduct experiments addressing these issues. Combining datasets with objects of different sizes and shapes together, training a single deep learning model with input DTMs of each object at different scales, and using further augmentation techniques can perhaps help.

The predictions by the deep learning models are converted to vector data (shape files containing polygons for each detected structure) which can be loaded into ArcGIS projects (or other GIS tools) and analyzed by archaeologists for further improvement and analysis. The polygons for predicted structures are however sometimes overlapping (depending on the stride size during prediction using

Algorithm 2) or parts belonging to the same structure are split into two or more polygons due to mispreditions for some of the pixels (specially for linear structures). Therefore, further image processing techniques are required to connect separate predictions that belong to the same structure.

The predictions by deep learning approaches can be used by archaeologists to detect and analyze the changes that each structure undergo through time. For example, based on the detected polygons, the 3D shapes of objects can be automatically reconstructed (Elschen, 2018). The reconstructed shape can be compared with a previously stored shape or an expected ideal shape for the said structure in order to analyze the level of degradation that has occurred in the object.

Further research in this direction include deep learning applications for other structures related to historical mining and archaeology. Additional information can also be used as input to the models which can lead to better predictions. Examples include aerial photos corresponding to the regions for the same DTM inputs, tags from Open Street Map (OSM), e.g., building, highway, parks, etc. Finally, in addition to the encoder-decoder and the GAN-based approaches, other methods such as the convolutional variational autoencoder can be explored as the SSL pretext approach. Moreover, this research is focused on detecting archaeological monuments on the preprocessed ALS or LiDAR point cloud in the form of DTMs. Applications of deep learning techniques could be explored to directly work on the LiDAR point cloud directly, without the need for preprocessing.

List of Acronyms

ALS Airborne Laser Scanning
DTM Digital Terrain Model
SVF Sky View Factor
LiDAR Light Detection And Ranging
DCNN Deep Convolutional Neural Network
BCE Binary Cross Entropy
HRNet High Resolution Network
ResNet Residual Network
CE Cross Entropy
GAN Generative Adversarial Network
FPN Feature Pyramid Network
RPN Region Proposal Network
SSL Self Supervised Learning
DL Deep Learning
DNN Deep Neural Network
ReLU Rectified Linear Unit
ELU Exponential Linear Unit
CNN Convolutional Neural Network
LSTM Long Short Term Memory
SLRM Simple Local Relief Model
LD Local Dominance
POS Positive Openness
NEG Negative Openness
MSE Mean Squared Error
MAE Mean Absolute Error
SGD Stochastic Gradient Descent
GPU Graphics Processing Unit
RMSProp Root Mean Square Propagation
Adam Adaptive Moment Estimation
GPS Global Positioning System
IMU Inertial Measurement Unit
RADAR Radio Detection and Range
IFOV Instantaneous Field of View
GIS Geographic Information System
EPSG European Petroleum Survey Group
UTM Universal Transversal Mercator
TIN Triangulated Irregular Network
IDW Inverse Distance Weighting
DSM Digital Surface Model
DEM Digital Elevation Model
ArcGIS ArcGIS
QGIS Quantum GIS
SVM Support Vector Machine
RF Random Forest
ML Machine Learning
CV Computer Vision
ILSVRC ImageNet Large Scale Visual Recognition Challenge
VOC Visual Object Classes

NLP Natural Language Processing
ISPRS International Society for Photogrammetry and Remote Sensing
RVT Relief Visualization Toolbox
RVNet Relief Visualization Network
RVGan Relief Visualization GAN
ESA European Space Agency
NASA National Aeronautics and Space Administration
USGS United States Geological Survey
ILSVRC ImageNet Large Scale Visual Recognition Challenge
ISI Intelligent Systems and Informatics
FPN Feature Pyramid Network
RCNN Region-Based Convolutional Neural Networks
YOLO You Only Look Once
ROI Region of Interest
SPPNet Spatial Pyramid Pooling Network
R-FCN Region-based Fully Convolutional Network
PANet Path Aggregation Network
FCN Fully Convolutional Network
CRF Conditional Random Field
ASPP Atrous Spatial Pyramid Pooling
LULC Land Use and Land Cover Classification
AID Aerial Image Dataset
DIOR Detection in Optical Remote sensing
RICNN Rotation Invariant Convolutional Neural Network
DIRSIG Digital Imaging and Remote Sensing Image Generation
MLP Multi Layer Perceptron
OBIA Object Based Image Analysis
CAD Computer Aided Design
BoW Bag of Words
KNN K-Nearest Neighbor
LBP Local Binary Patterns
MAP Mean Average Precision
MIOU Mean Intersection Over Union
IOU Intersection Over Union

List of Figures

List of Tables

Bibliography

Adams, R. E., 1980. Swamps, canals, and the locations of ancient Maya cities. *Antiquity* 54 (212), pp. 206–214.

Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., Ogden, J. M., 1984. Pyramid methods in image processing. *RCA Engineer* 29 (6), pp. 33–41.

Agapiou, A., Hadjimitsis, D. G., Alexakis, D. D., 2012. Evaluation of broadband and narrowband vegetation indices for the identification of archaeological crop marks. *Remote Sensing* 4 (12), pp. 3892–3919.

Aggarwal, S., 2004. Principles of remote sensing. *Satellite Remote Sensing and GIS Applications in Agricultural Meteorology*, pp. 23–38.

Aha, D. W., Kibler, D., Albert, M. K., 1991. Instance-based learning algorithms. *Machine Learning* 6 (1), pp. 37–66.

Alexakis, D., Sarris, A., Astaras, T., Albanakis, K., 2009. Detection of Neolithic settlements in Thessaly (Greece) through multispectral and hyperspectral satellite imagery. *Sensors* 9 (2), pp. 1167–1187.

Armelagos, G. J., Cohen, M. N., 1984. Paleopathology at the origins of agriculture. Academic Press Orlando, FL.

Armesto-González, J., Riveiro-Rodríguez, B., González-Aguilera, D., Rivas-Brea, M. T., 2010. Terrestrial laser scanning intensity data applied to damage detection for historical buildings. *Journal of Archaeological Science* 37 (12), pp. 3037 – 3047.

Audebert, N., Le Saux, B., Lefèvre, S., 2017. Segment-before-detect: Vehicle detection and classification through semantic segmentation of aerial images. *Remote Sensing* 9 (4), pp. 368.

Awange, J. L., Kyalo Kiema, J. B., 2013. Fundamentals of remote sensing. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 111–118.

Axelsson, P., 2000. DEM generation from laser scanner data using adaptive TIN models. *International Archives of Photogrammetry and Remote Sensing* 33 (4), pp. 110–117.

Badrinarayanan, V., Kendall, A., Cipolla, R., 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (12), pp. 2481–2495.

Barlindhaug, S., Holm-Olsen, I. M., Tømmervik, H., 2007. Monitoring archaeological sites in a changing landscape–using multitemporal satellite remote sensing as an 'early warning'method for detecting regrowth processes. *Archaeological Prospection* 14 (4), pp. 231–244.

Bartels, C., Klappauf, L., 2012. Das Mittelalter. – Geschichte des deutschen Bergbaus, Bd. 1, Aschendorff-Verlag, Münster. , pp. 111–248.

Basu, S., Ganguly, S., Mukhopadhyay, S., DiBiano, R., Karki, M., Nemani, R., 2015. Deepsat: a learning framework for satellite imagery. In: *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. pp. 1–10.

Beck, A., Cowley, D., 2011. Archaeological applications of multi/hyper-spectral data–challenges and potential. *Remote Sensing for Archaeological Heritage Management*, pp. 87–97.

Beraldin, J. ., Blais, F., Cournoyer, L., Rioux, M., El-Hakim, S. H., Rodella, R., Bernier, F., Harrison, N., 1999. Digital 3D imaging system for rapid response on remote sites. In: *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062)*. pp. 34–43.

Bewley, R., 1999. Archiving aerial photography and remote sensing data: a guide to good practice. Vol. 2. Oxbow Books Limited.

Bewley, R., 2002. Aerial survey: Learning from a hundred years of experience? *NATO Science Series Sub Series I Life and Behavioral Sciences* 337, pp. 11–18.

Bewley, R., 2003a. Aerial archaeology. The first century. *Aerial Photography and Archaeology*, pp. 15–30.

Bewley, R. H., 2003b. Aerial survey for archaeology. *The Photogrammetric Record* 18 (104), pp. 273–292.

Bitelli, G., Dellapasqua, M., Girelli, V., Sbaraglia, S., Tinia, M., 2017. Historical Photogrammetry and Terrestrial Laser Scanning for the 3d Virtual Reconstruction of Destroyed Structures: a Case Study in Italy. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 42, pp. 113–119.

Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y. M., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. *ArXiv E-prints*.

Boulch, A., Saux, B. L., Audebert, N., 2017. Unstructured point cloud semantic labeling using deep segmentation networks. In: *Proceedings of the Workshop on 3D Object Retrieval*. pp. 17–24.

Bowens, A., 2011. Underwater archaeology: the NAS guide to principles and practice. John Wiley & Sons.

Brenner, S., Zambanini, S., Sablatnig, R., 2018. Detection of bomb craters in WWII aerial images. In: *Proceedings of the OAGM Workshop*. pp. 94–97.

Brivio, P. A., Pepe, M., Tomasoni, R., 2000. Multispectral and multiscale remote sensing data for archaeological prospecting in an alpine alluvial plain. *Journal of Cultural Heritage* 1 (2), pp. 155–164.

Bundzel, M., Jaščur, M., Kováč, M., Lieskovskỳ, T., Sinčák, P., Tkáčik, T., 2020. Semantic segmentation of airborne lidar data in maya archaeology. *Remote Sensing* 12 (22).

Campana, S., 2017. Remote sensing in archaeology. Springer Netherlands, Dordrecht, pp. 703–725.

Campbell, B. A., 2002. Radar remote sensing of planetary surfaces. Cambridge University Press.

Canuto, M. A., Estrada-Belli, F., Garrison, T. G., Houston, S. D., Acuña, M. J., Kováč, M., Marken, D., Nondédéo, P., Auld-Thomas, L., Castanet, C., et al., 2018. Ancient lowland Maya complexity as revealed by airborne laser scanning of northern Guatemala. *Science* 361 (6409).

Capper, J. E., 1907. XXIII.—Photographs of Stonehenge, as seen from a War Balloon. *Archaeologia* 60 (2), pp. 571–571.

Carrara, A., Bitelli, G., Carla, R., 1997. Comparison of techniques for generating digital terrain models from contour lines. *International Journal of Geographical Information Science* 11 (5), pp. 451–473.

Castagnetti, C., Bertacchini, E., Capra, A., Dubbini, M., 2012. Terrestrial laser scanning for preserving cultural heritage: analysis of geometric anomalies for ancient structures. In: *FIG Working Week 2012–Territory, Environment, and Cultural Heritage*. FIG Federation International des Geometres, pp. 1–13.

Chang, C.-C., Lin, C.-J., 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2 (3), pp. 1–27.

Chase, A. F., Chase, D. Z., Weishampel, J. F., Drake, J. B., Shrestha, R. L., Slatton, K. C., Awe, J. J., Carter, W. E., 2011. Airborne LiDAR, archaeology, and the ancient Maya landscape at Caracol, Belize. *Journal of Archaeological Science* 38 (2), pp. 387–398.

Chen, L., Papandreou, G., Schroff, F., Adam, H., 2017a. Rethinking Atrous Convolution for Semantic Image Segmentation. *CoRR* abs/1706.05587.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A. L., 2017b. Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (4), pp. 834–848.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H., 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. pp. 801–818.

Cheng, G., Han, J., 2016. A survey on object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing* 117, pp. 11–28.

Cheng, G., Han, J., Lu, X., 2017a. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE* 105 (10), pp. 1865–1883.

Cheng, G., Han, J., Zhou, P., Guo, L., 2014. Multi-class geospatial object detection and geographic image classification based on collection of part detectors. *ISPRS Journal of Photogrammetry and Remote Sensing* 98, pp. 119–132.

Cheng, G., Li, Z., Yao, X., Guo, L., Wei, Z., 2017b. Remote sensing image scene classification using bag of convolutional features. *IEEE Geoscience and Remote Sensing Letters* 14 (10), pp. 1735–1739.

Cheng, G., Yang, C., Yao, X., Guo, L., Han, J., 2018. When deep learning meets metric learning: remote sensing image scene classification via learning discriminative CNNs. *IEEE Transactions on Geoscience and Remote Sensing* 56 (5), pp. 2811–2821.

Cheng, G., Zhou, P., Han, J., 2016. Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing* 54 (12), pp. 7405–7415.

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

Chollet, F., et al., 2015. keras.

Chorowicz, J., Kim, J., Manoussis, S., Rudant, J.-P., Foin, P., Veillet, I., 1989. A new technique for recognition of geological and geomorphological patterns in digital terrain models. *Remote Sensing of Environment* 29 (3), pp. 229–239.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., 2016. The cityscapes dataset for semantic urban scene understanding. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 3213–3223.

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

Cox, M., Hunter, J., 2005. Forensic archaeology: advances in theory and practice. Routledge.

Cracknell, A. P., 2007. Introduction to remote sensing. CRC press.

Crawford, O. G. S., Keiller, A., 1928. Wessex from the Air. Clarendon Press.

Dai, A., Ritchie, D., Bokeloh, M., Reed, S., Sturm, J., Nießner, M., 2018. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 4578–4587.

Dai, J., Li, Y., He, K., Sun, J., 2016. R-FCN: object detection via region-based fully convolutional networks. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. pp. 379–387.

David, N., Kramer, C., Nicholas, D., 2001. Ethnoarchaeology in action. Cambridge University Press.

Dayaratne, R., 2012. Landscapes of nation: Constructing national identity in the deserts of Bahrain. *National Identities* 14 (3), pp. 309–327.

Deforce, K., Groenewoudt, B., Haneca, K., 2021. 2500 years of charcoal production in the Low Countries: The chronology and typology of charcoal kilns and their relation with early iron production. *Quaternary International* 593, pp. 295–305.

Deng, Z., Sun, H., Zhou, S., Zhao, J., Lei, L., Zou, H., 2018. Multi-scale object detection in remote sensing imagery with convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing* 145, pp. 3–22.

Deuel, L., 1969. Flights into yesterday: the story of aerial archaeology. St. Martin's Press.

Devereux, B. J., Amable, G. S., Crow, P., Cliff, A. D., 2005. The potential of airborne lidar for detection of archaeological features under woodland canopies. *Antiquity* 79 (305), pp. 648–660.

Di Fazio, S., Laudari, L., Modica, G., 2010. Heritage interpretation and landscape character in the forestry district of Serra San Bruno (Calabria, Italy). In: *XVII World Congress of the International Commission of Agricultural Engineering (CIGR) on Sustainable Biosystems through Engineering, Québec City, Canada.*

Diakogiannis, F. I., Waldner, F., Caccetta, P., Wu, C., 2020. Resunet-a: a deep learning framework for semantic segmentation of remotely sensed data. *ISPRS Journal of Photogrammetry and Remote Sensing* 162, pp. 94–114.

Diao, W., Sun, X., Zheng, X., Dou, F., Wang, H., Fu, K., 2016. Efficient saliency-based object detection in remote sensing images using deep belief networks. *IEEE Geoscience and Remote Sensing Letters* 13 (2), pp. 137–141.

Ding, P., Zhang, Y., Deng, W.-J., Jia, P., Kuijper, A., 2018. A light and faster regional convolutional neural network for object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing* 141, pp. 208–218.

Doersch, C., Gupta, A., Efros, A. A., 2015. Unsupervised visual representation learning by context prediction. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. pp. 1422–1430.

Dominguez, M., Dhamdhere, R., Petkar, A., Jain, S., Sah, S., Ptucha, R., 2018. General-purpose deep point cloud feature extractor. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, pp. 1972–1981.

Doneus, M., 2001. The impact of vertical photographs on analysis of archaeological landscapes. In: *Archaeological Prospection, 4th International Conference on Archaeological Prospection.* pp. 94–96.

Doneus, M., 2013. Openness as visualization technique for interpretative mapping of airborne Lidar derived digital terrain models. *Remote Sensing* 5 (12), pp. 6427–6442.

Dong, P., Chen, Q., 2017. LiDAR remote sensing and applications. CRC Press.

Donoghue, D., 2001. Multispectral remote sensing for archaeology. *Remote Sensing in Archaeology*, pp. 181–192.

Dubbini, M., Curzio, L. I., Campedelli, A., 2016. Digital elevation models from unmanned aerial vehicle surveys for archaeological interpretation of terrain anomalies: Case study of the Roman castrum of Burnum (Croatia). *Journal of Archaeological Science: Reports* 8, pp. 121–134.

Elschen, D., 2018. Automatische Parametrische Beschreibung von Bodendenkmalen. Diplomarbeit, Leibniz University Hannover, Institute of Cartography and Geoinformatics.

Engelmann, F., Kontogianni, T., Schult, J., Leibe, B., 2018. Know what your neighbors do: 3D semantic segmentation of point clouds. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops.*

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., Bengio, S., 2010. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research* 11, pp. 625–660.

Evans, D. H., Fletcher, R. J., Pottier, C., Chevance, J.-B., Soutif, D., Tan, B. S., Im, S., Ea, D., Tin, T., Kim, S., et al., 2013. Uncovering archaeological landscapes at Angkor using lidar. *Proceedings of the National Academy of Sciences* 110 (31), pp. 12595–12600.

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., Zisserman, A., 2015. The Pascal visual object classes challenge: a retrospective. *International Journal of Computer Vision* 111 (1), pp. 98–136.

Feichtinger, H. G., Strohmer, T., 2002. Advances in Gabor analysis. Birkhauser Boston, Inc., USA.

Fischer, W., Hemphill, W., Kover, A., 1976. Progress in remote sensing (1972–1976). *Photogrammetria* 32 (2), pp. 33 – 72.

Florinsky, I. V., 1998. Combined analysis of digital terrain models and remotely sensed data in landscape investigations. *Progress in Physical Geography* 22 (1), pp. 33–60.

Forte, M., Campana, S., 2016. Digital methods and remote sensing in archaeology. *Archaeology in the Age of Sensing*.

Forte, M., Pescarin, S., Pietroni, E., Dell'Unto, N., 2005. The Appia antica project. In: *Archaeological Landscapes through Digital Technologies: Proceedings of the 2nd Italy-United States Workshop. British Archaeological Report (BAR)*.

Fowler, M. J., Fowler, Y. M., 2005. Detection of archaeological crop marks on declassified CORONA KH-4B intelligence satellite photography of Southern England. *Archaeological Prospection* 12 (4), pp. 257–264.

Gal, Y., Ghahramani, Z., 2016. Dropout as a bayesian approximation: representing model uncertainty in deep learning. In: *International Conference on Machine learning*. pp. 1050–1059.

Gallwey, J., Eyre, M., Tonkins, M., Coggan, J., 2019. Bringing lunar LiDAR back down to earth: mapping our industrial heritage through deep transfer learning. *Remote Sensing* 11 (17), pp. 1994.

Garrison, T. G., Chapman, B., Houston, S., Román, E., López, J. L. G., 2011. Discovering ancient Maya settlements using airborne radar elevation data. *Journal of Archaeological Science* 38 (7), pp. 1655–1662.

Gelbman, E., Papo, H., 1984. Digital terrain models for slopes and curvatures. *Photogrammetric Engineering and Remote Sensing* 50 (6), pp. 695–701.

Gennaro, A., Candiano, A., Fargione, G., Mangiameli, M., Mussumeci, G., 2019. Multispectral remote sensing for post-dictive analysis of archaeological remains. A case study from Bronte (Sicily). *Archaeological Prospection* 26 (4), pp. 299–311.

Gheyle, W., Stichelbaut, B., Saey, T., Note, N., Van den Berghe, H., Van Eetvelde, V., Van Meirvenne, M., Bourgeois, J., 2018. Scratching the surface of war. Airborne laser scans of the Great War conflict landscape in Flanders (Belgium). *Applied Geography* 90, pp. 55–68.

Gidaris, S., Singh, P., Komodakis, N., 2018. Unsupervised Representation Learning by Predicting Image Rotations. In: *International Conference on Learning Representations*.

Girshick, R., 2015. Fast r-cnn. In: *Proceedings of the IEEE International Conference on Computer Vision*. pp. 1440–1448.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 580–587.

Going, C., 2002. A neglected asset. German aerial photography of the Second World War period. *NATO Science Series Sub Series I Life and Behavioral Sciences* 337, pp. 23–32.

Gomez-Lahoz, J., Gonzalez-Aguilera, D., 2009. Recovering traditions in the digital era: the use of blimps for modelling the archaeological cultural heritage. *Journal of Archaeological Science* 36 (1), pp. 100 – 109.

Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y., 2016. Deep learning. Vol. 1. MIT press Cambridge.

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

Green, W., Doershuk, J. F., 1998. Cultural resource management and American archaeology. *Journal of Archaeological Research* 6 (2), pp. 121–167.

Grosse, G., Schirrmeister, L., Kunitsky, V. V., Hubberten, H.-W., 2005. The use of CORONA images in remote sensing of periglacial geomorphology: an illustration from the NE Siberian coast. *Permafrost and periglacial processes* 16 (2), pp. 163–172.

Grussenmeyer, P., Alby, E., Landes, T., Koehl, M., Guillemin, S., Hullo, J.-F., Assali, P., Smigiel, E., 2012. Recording approach of heritage sites based on merging point clouds from high resolution photogrammetry and terrestrial laser scanning. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 39, pp. B5.

Guo, X., Liu, X., Zhu, E., Yin, J., 2017. Deep clustering with convolutional autoencoders. In: *International Conference on Neural Information Processing*. Springer, pp. 373–382.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M., 2020. Deep learning for 3D point clouds: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Hadjimitsis, D., Themistocleous, K., Agapiou, A., Clayton, C., 2009. Monitoring archaeological site landscapes in Cyprus using multi-temporal atmospheric corrected image data. *International Journal of Architectural Computing* 7 (1), pp. 121–138.

Hadjimitsis, D. G., Agapiou, A., Themistocleous, K., Alexakis, D. D., Sarris, A., 2013. Remote sensing for archaeological applications: management, documentation and monitoring. In: *Remote Sensing of Environment-Integrated Approaches*. InTech Publisher, pp. 57–95.

Harris, T., Lock, G., 1988. Digital terrain modelling and three-dimensional surface graphics for landscape and site analysis in archaeology and regional planning. *Computer and Quantitative Methods in Archaeology 1987*, pp. 161–170.

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask r-cnn. In: *Proceedings of the IEEE International Conference on Computer Vision*. pp. 2961–2969.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37 (9), pp. 1904–1916.

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

Helber, P., Bischke, B., Dengel, A., Borth, D., 2019. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12 (7), pp. 2217–2226.

Heo, J., Jung, J., Kim, B., Han, S., 2020. Digital elevation model-based convolutional neural network modeling for searching of high solar energy regions. *Applied Energy* 262, pp. 114588.

Hesse, R., 2010. LiDAR-derived local relief models–a new tool for archaeological prospection. *Archaeological Prospection* 17 (2), pp. 67–72.

Hesse, R., 2016. Visualisierung hochauflösender digitaler Geländemodelle mit LiVT. .

Hirt, C., 2014. Digital Terrain Models. Springer International Publishing, Cham, pp. 1–6.

Horne, P., 2011. The English heritage national mapping programme. *Remote Sensing for Archaeological Heritage Management: Proceedings of the 11th EAC Heritage Management Symposium*, pp. 143–151.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv e-prints*, pp. arXiv–1704.

Hu, F., Xia, G.-S., Hu, J., Zhang, L., 2015. Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery. *Remote Sensing* 7 (11), pp. 14680–14707.

Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A., 2020. Randla-net: Efficient semantic segmentation of large-scale point clouds. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 11108–11117.

Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K. Q., 2017. Densely connected convolutional networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 4700–4708.

Hudson, K., 2014. Industrial archaeology: an introduction. Routledge.

Iriarte, J., Robinson, M., de Souza, J., Damasceno, A., da Silva, F., Nakahara, F., Ranzi, A., Aragao, L., 2020. Geometry by design: contribution of lidar to the understanding of settlement patterns of the Mound villages in SW Amazonia. *Journal of Computer Applications in Archaeology* 3 (1), pp. 151–169.

Isola, P., Zhu, J.-Y., Zhou, T., Efros, A. A., 2017. Image-to-image translation with conditional adversarial networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1125–1134.

James, K., Nichol, C. J., Wade, T., Cowley, D., Gibson Poole, S., Gray, A., Gillespie, J., 2020. Thermal and multispectral remote sensing for the detection and analysis of archaeologically induced crop stress at a UK site. *Drones* 4 (4), pp. 61.

Jin, Y., Zhang, J., Li, M., Tian, Y., Zhu, H., 2017. Towards the high-quality anime characters generation with generative adversarial networks. In: *Proceedings of the Machine Learning for Creativity and Design Workshop at Neural Information Processing Systems*.

Kampffmeyer, M., Salberg, A.-B., Jenssen, R., 2016. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. pp. 1–9.

Kazimi, B., Malek, K., Thiemann, F., Sester, M., 2019a. Effectiveness of DTM derivatives for object detection using deep learning. In: *International Conference on Cultural Heritage and New Technologies*.

Kazimi, B., Thiemann, F., Sester, M., 2019b. Semantic segmentation of manmade landscape structures in digital terrain models. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* 42, pp. 87–94.

Kazimi, B., Thiemann, F., Sester, M., 2020. Detection of terrain structures in airborne laser scanning data using deep learning. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* 5, pp. 493–500.

Kemker, R., Salvaggio, C., Kanan, C., 2018. Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning. *ISPRS Journal of Photogrammetry and Remote Sensing* 145, pp. 60–77.

Kersten, T. P., Lindstaedt, M., 2012. Automatic 3D object reconstruction from multiple images for architectural, cultural heritage and archaeological applications using open-source software and web services. *Photogrammetrie - Fernerkundung - Geoinformation* 2012 (6), pp. 727–740.

Kingma, D. P., Ba, J., 2015. Adam: a method for stochastic optimization. In: Bengio, Y., LeCun, Y. (Hrsg.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kirsch, R. A., 1971. Computer determination of the constituent structure of biological images. *Computers and Biomedical Research* 4 (3), pp. 315–328.

Klassen, S., Weed, J., Evans, D., 2018. Semi-supervised machine learning approaches for predicting the chronology of archaeological sites: A case study of temples from medieval Angkor, Cambodia. *PLOS ONE* 13 (11), pp. 1–17.

Klemas, V., 2010. Remote sensing techniques for studying coastal ecosystems: an overview. *Journal of Coastal Research* 27 (1), pp. 2–17.

Koehn, P., 2005. Europarl: A parallel corpus for statistical machine translation. In: *MT summit*. Vol. 5. Citeseer, pp. 79–86.

Kokalj, Ž., Hesse, R., 2017. Airborne laser scanning raster data visualization: A Guide to Good Practice.

Kokalj, Ž., Somrak, M., 2019. Why not a single image? Combining visualizations to facilitate fieldwork and on-screen mapping. *Remote Sensing* 11 (7), pp. 747.

Kokalj, Ž., Zakšek, K., Oštir, K., 2013. Visualizations of lidar derived relief models. *Interpreting Archaeological Topography–Airborne Laser Scanning, Aerial Photographs and Ground Observation*, pp. 100–114.

Kostka, R., 2002. The world mountain Damavand: documentation and monitoring of human activities using remote sensing data. *ISPRS Journal of Photogrammetry and Remote Sensing* 57 (1-2), pp. 5–12.

Krähenbühl, P., Koltun, V., 2011. Efficient inference in fully connected crfs with gaussian edge potentials. *Advances in Neural Information Processing Systems* 24, pp. 109–117.

Krizhevsky, A., Sutskever, I., Hinton, G. E., 2012. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira, F., Burges, C. J. C., Bottou, L., Weinberger, K. Q. (Hrsg.), *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., pp. 1097–1105.

Kumar, A., Sattigeri, P., Fletcher, T., 2017. Semi-supervised learning with gans: Manifold invariance with improved inference. In: *Advances in Neural Information Processing Systems*. pp. 5534–5544.

Kussul, N., Lavreniuk, M., Skakun, S., Shelestov, A., 2017. Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geoscience and Remote Sensing Letters* 14 (5), pp. 778–782.

Lambers, K., Verschoof-van der Vaart, W. B., Bourgeois, Q. P. J., 2019. Integrating remote sensing, machine learning, and citizen science in Dutch archaeological prospection. *Remote Sensing* 11 (7).

Lan, S., Yu, R., Yu, G., Davis, L. S., 2019. Modeling local geometric structure of 3d point clouds using geo-cnn. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 998–1008.

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O., 2019. Pointpillars: Fast encoders for object detection from point clouds. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 12697–12705.

Langley, R. B., 1998. The UTM grid system. *GPS world* 9 (2), pp. 46–50.

Larsson, G., Maire, M., Shakhnarovich, G., 2016. Learning representations for automatic colorization. In: *European Conference on Computer Vision*. Springer, pp. 577–593.

Lasaponara, R., Masini, N., 2006a. Identification of archaeological buried remains based on the normalized difference vegetation index (NDVI) from quickbird satellite data. *IEEE Geoscience and Remote Sensing Letters* 3 (3), pp. 325–328.

Lasaponara, R., Masini, N., 2006b. On the potential of QuickBird data for archaeological prospection. *International Journal of Remote Sensing* 27 (16), pp. 3607–3614.

Lawin, F. J., Danelljan, M., Tosteberg, P., Bhat, G., Khan, F. S., Felsberg, M., 2017. Deep projective 3D semantic segmentation. In: *International Conference on Computer Analysis of Images and Patterns*. Springer, pp. 95–107.

Le, T., Duan, Y., 2018. Pointgrid: A deep network for 3d shape understanding. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 9204–9214.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. *Nature* 521 (7553), pp. 436–444.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86 (11), pp. 2278–2324.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al., 2017. Photo-realistic single image super-resolution using a generative adversarial network. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 4681–4690.

Lee, C., 2019. Automated crater detection on Mars using deep learning. *Planetary and Space Science* 170, pp. 16–28.

Lee, H.-Y., Huang, J.-B., Singh, M., Yang, M.-H., 2017. Unsupervised representation learning by sorting sequences. In: *Proceedings of the IEEE International Conference on Computer Vision*. pp. 667–676.

Leonard, P. B., Baldwin, R. F., Homyack, J. A., Wigley, T., 2012. Remote detection of small wetlands in the Atlantic coastal plain of North America: Local relief models, ground validation, and high-throughput computing. *Forest Ecology and Management* 284, pp. 107 – 115.

Lercari, N., 2019. Monitoring earthen archaeological heritage using multi-temporal terrestrial laser scanning and surface change detection. *Journal of Cultural Heritage* 39, pp. 152 – 165.

Lerma, J. L., Navarro, S., Cabrelles, M., Villaverde, V., 2010. Terrestrial laser scanning and close range photogrammetry for 3D archaeological documentation: the Upper Palaeolithic Cave of Parpalló as a case study. *Journal of Archaeological Science* 37 (3), pp. 499 – 507.

Li, B., 2017. 3d fully convolutional network for vehicle detection in point cloud. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1513–1518.

Li, B., Zhang, T., Xia, T., 2016. Vehicle detection from 3D lidar using fully convolutional network. In: Hsu, D., Amato, N. M., Berman, S., Jacobs, S. A. (Hrsg.), *Robotics: Science and Systems XII, University of Michigan, Ann Arbor, Michigan, USA, June 18 - June 22, 2016*.

Li, E., Xia, J., Du, P., Lin, C., Samat, A., 2017a. Integrating multilayer features of convolutional neural networks for remote sensing scene classification. *IEEE Transactions on Geoscience and Remote Sensing* 55 (10), pp. 5653–5665.

Li, J., Heap, A. D., 2008. A review of spatial interpolation methods for environmental scientists. Geoscience Australia Canberra.

Li, K., Cheng, G., Bu, S., You, X., 2017b. Rotation-insensitive and context-augmented object detection in remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing* 56 (4), pp. 2337–2348.

Li, K., Wan, G., Cheng, G., Meng, L., Han, J., 2020a. Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing* 159, pp. 296–307.

Li, W., Wang, F.-D., Xia, G.-S., 2020b. A geometry-attentional network for ALS point cloud classification. *ISPRS Journal of Photogrammetry and Remote Sensing* 164, pp. 26–40.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B., 2018a. Pointcnn: Convolution on x-transformed points. *Advances in Neural Information Processing Systems* 31, pp. 820–830.

Li, Y., Zhang, Y., Huang, X., Yuille, A. L., 2018b. Deep networks under scene-level supervision for multi-class geospatial object detection from remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing* 146, pp. 182–196.

Lin, G., Milan, A., Shen, C., Reid, I., 2017a. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1925–1934.

Lin, M., Chen, Q., Yan, S., 2014. Network in network. In: Bengio, Y., LeCun, Y. (Hrsg.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017b. Feature pyramid networks for object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 2117–2125.

Lindstaedt, M., Mechelke, K., Schnelle, M., Kersten, T., 2011. Virtual reconstruction of the Almaqah Temple of Yeha in Ethiopia by terrestrial laser scanning. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 38 (5/W16), pp. 381–390.

Liu, K., Mattyus, G., 2015. Fast multiclass vehicle detection on aerial images. *IEEE Geoscience and Remote Sensing Letters* 12 (9), pp. 1938–1942.

Liu, K., Sessions, J., 1993. Preliminary planning of road systems using digital terrain models. *Journal of Forest Engineering* 4 (2), pp. 27–32.

Liu, S., Qi, L., Qin, H., Shi, J., Jia, J., 2018. Path aggregation network for instance segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 8759–8768.

Liu, Y., Fan, B., Xiang, S., Pan, C., 2019. Relation-shape convolutional neural network for point cloud analysis. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 8895–8904.

Lobb, M., Krawiec, K., Howard, A. J., Gearey, B. R., Chapman, H. P., 2010. A new approach to recording and monitoring wet-preserved archaeological wood using three-dimensional laser scanning. *Journal of Archaeological Science* 37 (12), pp. 2995–2999.

Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 3431–3440.

Lu, X., Zheng, X., Yuan, Y., 2017. Remote sensing scene classification by unsupervised representation learning. *IEEE Transactions on Geoscience and Remote Sensing* 55 (9), pp. 5148–5157.

Maguya, A. S., Junttila, V., Kauranne, T., 2014. Algorithm for extracting digital terrain models under forest canopy from airborne LiDAR data. *Remote Sensing* 6 (7), pp. 6524–6548.

Makridis, M., Daras, P., 2013. Automatic classification of archaeological pottery sherds. *Journal on Computing and Cultural Heritage (JOCCH)* 5 (4), pp. 1–21.

Malek, K., 2017. Montanarchäologische Forschungen im Harz. Ein Ausblick. *Montanregion als historisches Erbe. Reflexionen und Ausblicke. Beiträge zum Kolloquium 25 Jahre Welterbe im Harz am 22. und 23.September 2017 im Weltkulturerbe Rammelsberg, Museum und Besucherbergwerk in Goslar. Quensen Druck und Verlag Hildesheim.*

Malek, K., Klappauf, L., 2017. Frühgeschichte des Bergbaus im Harz. *Bergwerk Rammelsberg, Altstadt Goslar, Oberharzer Wasserwirtschaft, Goslar*, pp. 20–35.

Mallat, S., 1999. A wavelet tour of signal processing. Elsevier.

Marcos, D., Volpi, M., Kellenberger, B., Tuia, D., 2018. Land cover mapping at very high resolution with rotation equivariant CNNs: Towards small yet accurate models. *ISPRS Journal of Photogrammetry and Remote Sensing* 145, pp. 96–107.

Marmanis, D., Adam, F., Datcu, M., Esch, T., Stilla, U., 2015. Deep neural networks for above-ground detection in very high spatial resolution digital elevation models. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 2 (3), pp. 103.

Marmanis, D., Wegner, J. D., Galliani, S., Schindler, K., Datcu, M., Stilla, U., 2016. Semantic segmentation of aerial images with an ensemble of CNSS. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2016* 3, pp. 473–480.

Martin, D. L., Harrod, R. P., Pérez, V. R., 2013. Bioarchaeology. Springer.

Masci, J., Meier, U., Cireşan, D., Schmidhuber, J., 2011. Stacked convolutional auto-encoders for hierarchical feature extraction. In: *International Conference on Artificial Neural Networks*. Springer, pp. 52–59.

Masini, N., Coluzzi, R., Lasaponara, R., 2011. On the airborne Lidar contribution in archaeology: from site identification to landscape investigation. In: *Laser Scanning, Theory and Applications*. IntechOpen.

Masini, N., Gizzi, F. T., Biscione, M., Fundone, V., Sedile, M., Sileo, M., Pecci, A., Lacovara, B., Lasaponara, R., 2018. Medieval archaeology under the canopy with lidar. the (re) discovery of a medieval fortified settlement in southern Italy. *Remote Sensing* 10 (10), pp. 1598.

Maturana, D., Scherer, S., 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 922–928.

Maxwell, A. E., Bester, M. S., Guillen, L. A., Ramezan, C. A., Carpinello, D. J., Fan, Y., Hartley, F. M., Maynard, S. M., Pyron, J. L., 2020a. Semantic segmentation deep learning for extracting surface mine extents from historic topographic maps. *Remote Sensing* 12 (24).

Maxwell, A. E., Pourmohammadi, P., Poyner, J. D., 2020b. Mapping the topographic features of mining-related valley fills using mask R-CNN deep learning and digital elevation data. *Remote Sensing* 12 (3), pp. 547.

Meyer, G. P., Laddha, A., Kee, E., Vallespi-Gonzalez, C., Wellington, C. K., 2019a. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 12677–12686.

Meyer, M. F., Pfeffer, I., Jürgens, C., 2019b. Automated detection of field monuments in digital terrain models of Westphalia using OBIA. *Geosciences* 9 (3), pp. 109.

Milz, S., Simon, M., Fischer, K., Pöpperl, M., Gross, H.-M., 2019. Points2Pix: 3D point-cloud to image translation using conditional GANs. In: *German Conference on Pattern Recognition*. Springer, pp. 387–400.

Mirza, M., Osindero, S., 2014. Conditional generative adversarial nets. *arXiv*, pp. arXiv–1411.

Misra, I., Zitnick, C. L., Hebert, M., 2016. Shuffle and learn: unsupervised learning using temporal order verification. In: *European Conference on Computer Vision*. Springer, pp. 527–544.

Moore, E., Freeman, T., Hensley, S., 2006. Spaceborne and airborne radar at Angkor: Introducing new technology to the ancient site. In: *Remote Sensing in Archaeology*. Springer, pp. 185–216.

Moore, I. D., Grayson, R., Ladson, A., 1991. Digital terrain modelling: a review of hydrological, geomorphological, and biological applications. *Hydrological Processes* 5 (1), pp. 3–30.

Moriarty, C., Cowley, D. C., Wade, T., Nichol, C. J., 2019. Deploying multispectral remote sensing for multi-temporal analysis of archaeological crop stress at Ravenshall, Fife, Scotland. *Archaeological Prospection* 26 (1), pp. 33–46.

Myers, J., Piccarreta, F., Ceraudo, G., 2002. Manuale di aerofotografia archeologica: Metodologia, tecniche e applicazioni. *American Journal of Archaeology* 106, pp. 482.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A. Y., 2011. Reading digits in natural images with unsupervised feature learning. In: *Neural Information Processing Systems Workshop on Deep Learning and Unsupervised Feature Learning*.

Neubauer, W., Doneus, M., Studnicka, N., Riegl, J., 2005. Combined high resolution laser scanning and photogrammetrical documentation of the pyramids at Giza. In: *CIPA XX International Symposium*. Citeseer, pp. 470–475.

Nielsen, M. A., 2015. Neural networks and deep learning. Vol. 25. Determination Press San Francisco, CA.

Nogueira, K., Penatti, O. A., Dos Santos, J. A., 2017. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition* 61, pp. 539–556.

Noroozi, M., Favaro, P., 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. In: *European Conference on Computer Vision*. Springer, pp. 69–84.

Obe, R., Hsu, L., 2011. PostGIS in action. *GEOInformatics* 14 (8), pp. 30.

Ojala, T., Pietikainen, M., Harwood, D., 1994. Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. In: *Proceedings of 12th International Conference on Pattern Recognition*. Vol. 2. IEEE Computer Society, Los Alamitos, CA, USA, pp. 582,583,584,585.

Orengo, H., Garcia-Molsosa, A., 2019. A brave new world for archaeological survey: Automated machine learning-based potsherd detection using high-resolution drone imagery. *Journal of Archaeological Science* 112, pp. 105013.

Orengo, H. A., Petrie, C. A., 2018. Multi-scale relief model (MSRM): a new algorithm for the visualization of subtle topographic change of variable size in digital elevation models. *Earth Surface Processes and Landforms* 43 (6), pp. 1361–1369.

Palafox, L. F., Hamilton, C. W., Scheidt, S. P., Alvarez, A. M., 2017. Automated detection of geological landforms on Mars using convolutional neural networks. *Computers & Geosciences* 101, pp. 48–56.

Palmer, R., 2007. Seventy-five years v. ninety minutes: implications of the 1996 Bedfordshire vertical aerial survey on our perceptions of clayland archaeology. *Populating Clay Landscapes*, pp. 88–103.

Pan, B., Shi, Z., Xu, X., Shi, T., Zhang, N., Zhu, X., 2018. CoinNet: Copy initialization network for multispectral imagery semantic segmentation. *IEEE Geoscience and Remote Sensing Letters* 16 (5), pp. 816–820.

Parcak, S. H., 2009. Satellite remote sensing for archaeology. Routledge.

Passmore, D. G., Harrison, S., Tunwell, D. C., 2014. Second World War conflict archaeology in the forests of north-west Europe. *Antiquity* 88 (342), pp. 1275–1290.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A. A., 2016. Context encoders: Feature learning by inpainting. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 2536–2544.

Pavlidis, G., Koutsoudis, A., Arnaoutoglou, F., Tsioukas, V., Chamzas, C., 2007. Methods for 3D digitization of cultural heritage. *Journal of Cultural Heritage* 8, pp. 93–98.

Pearsall, D. M., 2015. Paleoethnobotany: a handbook of procedures. Left Coast Press.

Penatti, O. A., Nogueira, K., Dos Santos, J. A., 2015. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition workshops*. pp. 44–51.

Peters, T., Brenner, C., 2019. Automatic generation of large point cloud training datasets using label transfer. *Wissenschaftlich-Technische Jahrestagung der DGPF* 39.

Peters, T., Brenner, C., 2020. Conditional adversarial networks for multimodal photo-realistic point cloud rendering. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 88 (3), pp. 257–269.

Peters, T., Brenner, C., Song, M., 2020. Improving deep learning based semantic segmentation with multi view outlier correction. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 43 (2), pp. 711–716.

Peucker, T., Fowler, R., Little, J., 2017. Triangulated irregular network. In: *Encyclopedia of GIS*.

Pinheiro, P. O., Lin, T.-Y., Collobert, R., Dollár, P., 2016. Learning to refine object segments. In: *European Conference on Computer Vision*. Springer, pp. 75–91.

Politz, F., Kazimi, B., Sester, M., 2018. Classification of laser scanning data using deep learning. In: *38th Scientific Technical Annual Meeting of the German Society for Photogrammetry, Remote Sensing and Geoinformation*. Vol. 27.

Politz, F., Sester, M., 2018. Exploring ALS and DIM data for semantic segmentation using CNNs. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives 42 (2018), Nr. 1* 42 (1), pp. 347–354.

Politz, F., Sester, M., 2019. Joint classification of ALS and DIM point clouds. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives 42 (2019), Nr. 2/W13* 42 (2/W13), pp. 1113–1120.

Politz, F., Sester, M., Brenner, C., 2020. Geometry-based point cloud classification using height distributions. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* 5, pp. 259–266.

Pope, K. O., Dahlin, B. H., 1989. Ancient Maya wetland agriculture: New insights from ecological and remote sensing research. *Journal of Field Archaeology* 16 (1), pp. 87–106.

Praetzellis, M., Praetzellis, A., 2011. Cultural resource management archaeology and heritage values. *Historical Archaeology* 45 (1), pp. 86–100.

Price, R., Vojinovic, Z., 2008. Urban flood disaster management. *Urban Water Journal* 5 (3), pp. 259–276.

Qi, C. R., Liu, W., Wu, C., Su, H., Guibas, L. J., 2018. Frustum pointnets for 3d object detection from rgb-d data. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 918–927.

Qi, C. R., Su, H., Mo, K., Guibas, L. J., 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 652–660.

Qi, C. R., Yi, L., Su, H., Guibas, L. J., 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. pp. 5105–5114.

Qin, N., Hu, X., Wang, P., Shan, J., Li, Y., 2019. Semantic labeling of ALS point cloud via learning voxel and pixel representations. *IEEE Geoscience and Remote Sensing Letters* 17 (5), pp. 859–863.

Quinn, P., Beven, K., Chevallier, P., Planchon, O., 1991. The prediction of hillslope flow paths for distributed hydrological modelling using digital terrain models. *Hydrological Processes* 5 (1), pp. 59–79.

Rączkowski, W., 2001. Science and/or art: aerial photographs in archaeological discourse. *Archaeologia Polona* 39, pp. 127–146.

Rączkowski, W., 2005. To Overcome Infirmity. In: *Aerial Photography and Archaeology 2003: A Century of Information; Papers Presented During the Conference Held at the Ghent University, December 10th-12th, 2003*. Vol. 4. Academia PressScientific Pub, p. 1121.

Radford, A., Metz, L., Chintala, S., 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. In: Bengio, Y., LeCun, Y. (Hrsg.), *4th International Conference on Learning Representations, 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Rajani, M., Kasturirangan, K., 2014. Multispectral remote sensing data analysis and application for detecting moats around medieval settlements in South India. *Journal of the Indian Society of Remote Sensing* 42 (3), pp. 651–657.

Raper, J., 1989. Three dimensional applications in GIS. CRC Press.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 779–788.

Redmon, J., Farhadi, A., 2017. YOLO9000: better, faster, stronger. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 7263–7271.

Redmon, J., Farhadi, A., 2018. YOLOv3: an incremental improvement. *CoRR* abs/1804.02767.

Reitz, E., Shackley, M., 2012. Environmental archaeology. Springer Science & Business Media.

Reitz, E. J., Reitz, E., Wing, E. S., 1999. Zooarchaeology. Cambridge University Press.

Ren, S., He, K., Girshick, R., Sun, J., 2016. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (6), pp. 1137–1149.

Riegler, G., Osman Ulusoy, A., Geiger, A., 2017. Octnet: Learning deep 3d representations at high resolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 3577–3586.

Risbøl, O., Bollandsås, O. M., Nesbakken, A., Ørka, H. O., Næsset, E., Gobakken, T., 2013. Interpreting cultural remains in airborne laser scanning generated digital terrain models: effects of size and shape on detection success rates. *Journal of Archaeological Science* 40 (12), pp. 4688–4700.

Risbøl, O., Briese, C., Doneus, M., Nesbakken, A., 2015. Monitoring cultural heritage by comparing DEMs derived from historical aerial photographs and airborne laser scanning. *Journal of Cultural Heritage* 16 (2), pp. 202–209.

Roman, A., Ursu, T.-M., Fărcaş, S., Lăzărescu, V.-A., Opreanu, C. H., 2017. An integrated airborne laser scanning approach to forest management and cultural heritage issues: a case study at Porolissum, Romania. *Annals of Forest Research* 60 (1), pp. 127–143.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer, pp. 234–241.

Rottensteiner, F., Sohn, G., Jung, J., Gerke, M., Baillard, C., Benitez, S., Breitkopf, U., 2012. The ISPRS benchmark on urban object classification and 3D building reconstruction. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences I-3 (2012), Nr. 1* 1 (1), pp. 293–298.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., Fei-Fei, L., 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)* 115 (3), pp. 211–252.

Rußwurm, M., Körner, M., 2018. Multi-temporal land cover classification with sequential recurrent encoders. *ISPRS International Journal of Geo-Information* 7 (4), pp. 129.

Rustowicz, R., Cheong, R., Wang, L., Ermon, S., Burke, M., Lobell, D., 2019. Semantic segmentation of crop type in Africa: A novel dataset and analysis of deep learning methods. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. pp. 75–82.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 4510–4520.

Sauder, J., Sievers, B., 2019. Self-supervised deep learning on point clouds by reconstructing space. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (Hrsg.), *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc.

Schiffer, M. B., Gumerman, G. J., 1977. Conservation archaeology. A guide for cultural resource management studies. Academic Press, New York.

Schler, J., Koppel, M., Argamon, S., Pennebaker, J. W., 2006. Effects of age and gender on blogging. In: *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*. Vol. 6. pp. 199–205.

Segers-Glocke, C., Witthöft, H., Balck, F. H., 2000. Aspects of mining and smelting in the upper Harz Mountains (up to the 13th/14th century) in the early times of a developing European culture and economy. Vol. 22. Scripta Mercaturae Verlag.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y., 2014. OverFeat: integrated recognition, localization and detection using convolutional networks. In: Bengio, Y., LeCun, Y. (Hrsg.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S., 2014. CNN features off-the-shelf: an astounding baseline for recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. pp. 806–813.

Shell, C., 2002. Airborne high-resolution digital, visible, infra-red and thermal sensing for archaeology. *NATO Science Series Sub Series I Life and Behavioural Sciences* 337, pp. 181–195.

Shen, Y., Feng, C., Yang, Y., Tian, D., 2018. Mining point cloud local structures by kernel correlation and graph pooling. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 4548–4557.

Shi, S., Wang, X., Li, H., 2019. Pointrcnn: 3d object proposal generation and detection from point cloud. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 770–779.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., Woo, W.-c., 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems* 28, pp. 802–810.

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., Webb, R., 2017. Learning from simulated and unsupervised images through adversarial training. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 2107–2116.

Shu, D. W., Park, S. W., Kwon, J., 2019. 3d point cloud generative adversarial network based on tree structured graph convolutions. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 3859–3868.

Sifre, L., Mallat, S., 2014. Rigid-motion scattering for image classification. *Ph. D. thesis*.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: *International Conference on Learning Representations*.

Snyder, J. P., 1987. Map projections–A working manual. Vol. 1395. US Government Printing Office.

Soilán, M., Lindenbergh, R., Riveiro, B., Sánchez-Rodríguez, A., 2019. Pointnet for the automatic classification Of aerial point clouds. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 4 (2/W5), pp. 445–452.

Somrak, M., Džeroski, S., Kokalj, Ž., 2020. Learning to classify structures in ALS-derived visualizations of ancient Maya Settlements with CNN. *Remote Sensing* 12 (14), pp. 2215.

Soroush, M., Mehrtash, A., Khazraee, E., Ur, J. A., 2020. Deep learning in archaeological remote sensing: Automated qanat detection in the kurdistan region of Iraq. *Remote Sensing* 12 (3), pp. 500.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15 (56), pp. 1929–1958.

Stewart, C., Lasaponara, R., Schiavon, G., 2013. ALOS PALSAR analysis of the archaeological site of Pelusium. *Archaeological Prospection* 20 (2), pp. 109–116.

Stöllner, T. R., 2014. Methods of mining archaeology (Montanarchäologie). Springer New York, New York, NY, pp. 133–159.

Stolze, F., Nöldeke, T., 1882. Persepolis: die Achaemenidischen und Sasanidischen Denkmäler und Inschriften von Persepolis, Istakhr, Pasargadae, Shâhpûr. .

Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E., 2015. Multi-view convolutional neural networks for 3d shape recognition. In: *Proceedings of the IEEE International Conference on Computer Vision*. pp. 945–953.

Sun, K., Xiao, B., Liu, D., Wang, J., 2019. Deep high-resolution representation learning for human pose estimation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 5693–5703.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 2818–2826.

Tan, M., Le, Q., 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: *International Conference on Machine Learning*. pp. 6105–6114.

Tapete, D., Cigna, F., Masini, N., Lasaponara, R., 2013. Prospection and monitoring of the archaeological heritage of Nasca, Peru, with ENVISAT ASAR. *Archaeological Prospection* 20 (2), pp. 133–147.

Tatarchenko, M., Park, J., Koltun, V., Zhou, Q.-Y., 2018. Tangent convolutions for dense prediction in 3d. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 3887–3896.

Tchapmi, L., Choy, C., Armeni, I., Gwak, J., Savarese, S., 2017. Segcloud: Semantic segmentation of 3d point clouds. In: *2017 International Conference on 3D Vision (3DV)*. IEEE, pp. 537–547.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., Guibas, L. J., 2019. Kpconv: Flexible and deformable convolution for point clouds. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 6411–6420.

Torres, R. N., Fraternali, P., Milani, F., Frajberg, D., 2018. A Deep Learning model for identifying mountain summits in Digital Elevation Model data. In: *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE, pp. 212–217.

Tracewski, L., Bastin, L., Fonte, C. C., 2017. Repurposing a deep learning network to filter and classify volunteered photographs for land cover and land use characterization. *Geo-spatial Information Science* 20 (3), pp. 252–268.

Traviglia, A., Torsello, A., 2017. Landscape pattern detection in archaeological remote sensing. *Geosciences* 7 (4).

Trier, Ø. D., Cowley, D. C., Waldeland, A. U., 2019. Using deep neural networks on airborne laser scanning data: Results from a case study of semi-automatic mapping of archaeological topography on Arran, Scotland. *Archaeological Prospection* 26 (2), pp. 165–175.

Trier, Ø. D., Reksten, J. H., Løseth, K., 2021. Automated mapping of cultural heritage in Norway from airborne lidar data using faster R-CNN. *International Journal of Applied Earth Observation and Geoinformation* 95, pp. 102241.

Trier, Ø. D., Salberg, A.-B., Pilø, L. H., 2018. Semi-automatic mapping of charcoal kilns from airborne laser scanning data using deep learning. In: *CAA2016: Oceans of Data. Proceedings of the 44th Conference on Computer Applications and Quantitative Methods in Archaeology*. Archaeopress Oxford, pp. 219–231.

Trier, Ø. D., Zortea, M., 2012. Semi-automatic detection of cultural heritage in lidar data. *Proceedings of the 4th GEOBIA, May 7* 9.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., Smeulders, A. W., 2013. Selective search for object recognition. *International Journal of Computer Vision* 104 (2), pp. 154–171.

Van Damme, T., 2015. Computer vision photogrammetry for underwater archaeological site recording in a low-visibility environment. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XL55, pp. 231–238.

van der Maaten, L., Boon, P., Lange, G., Paijmans, J., Postma, E., 2006. Computer vision and machine learning for archaeology. In: *Proceedings of Computer Applications in Archaeology.*

Van Etten, A., Lindenbaum, D., Bacastow, T. M., 2018. SpaceNet: a remote sensing dataset and challenge series. *arXiv*, pp. arXiv–1807.

Vaughn, S., Crawford, T., 2009. A predictive model of archaeological potential: An example from northwestern Belize. *Applied Geography* 29 (4), pp. 542–555.

Verhoeven, G. J., 2017. The reflection of two fields – Electromagnetic radiation and its role in (aerial) imaging. *AARGnews* 55 (October), pp. 13–18.

Verschoof-van der Vaart, W. B., Lambers, K., 2019. Learning to look at LiDAR: the use of R-CNN in the automated detection of archaeological objects in LiDAR data from the Netherlands. *Journal of Computer Applications in Archaeology* 2 (1).

Villalon-Turrubiates, I. E., Llovera-Torres, M. J., 2011. Archaeological land use characterization using multispectral remote sensing data. In: *2011 IEEE International Geoscience and Remote Sensing Symposium.* IEEE, pp. 86–89.

Volpi, M., Ferrari, V., 2015. Semantic segmentation of urban scenes by learning local class interactions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.* pp. 1–9.

Vondrick, C., Pirsiavash, H., Torralba, A., 2016. Generating videos with scene dynamics. In: *Advances in Neural Information Processing Systems.* pp. 613–621.

Vondrick, C., Shrivastava, A., Fathi, A., Guadarrama, S., Murphy, K., 2018. Tracking emerges by colorizing videos. In: *Proceedings of the European Conference on Computer Vision (ECCV).* pp. 391–408.

Vora, S., Lang, A. H., Helou, B., Beijbom, O., 2020. Pointpainting: sequential fusion for 3d object detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* pp. 4604–4612.

Wallach, H. M., 2006. Topic modeling: beyond bag-of-words. In: *Proceedings of the 23rd international conference on Machine learning.* pp. 977–984.

Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al., 2020. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., Solomon, J. M., 2019. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)* 38 (5), pp. 1–12.

Warden, P., 2018. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv e-prints*, pp. arXiv–1804.

Wei, O. C., Chin, C. S., Majid, Z., Setan, H., 2010. 3D documentation and preservation of historical monument using terrestrial laser scanning. *Geoinformation Science Journal* 10 (1), pp. 73–90.

Wei, X., Yu, R., Sun, J., 2020. View-GCN: View-based graph convolutional network for 3D shape analysis. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* pp. 1850–1859.

Winiwarter, L., Mandlburger, G., Schmohl, S., Pfeifer, N., 2019. Classification of ALS point clouds using end-to-end deep learning. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 87 (3), pp. 75–90.

Wood, D. J., Brown, C. R., Doyle, L., Smith, H. L., Waller, S., Weller, E. F., 2021. Identification of river defences from digital terrain models using deep learning. In: *4th European Conference on Flood Risk Management*. Budapest University of Technology and Economics.

Worboys, M. F., Duckham, M., 2004. GIS: a computing perspective. CRC press.

Wu, H., Zheng, S., Zhang, J., Huang, K., 2019a. Gp-gan: Towards realistic high-resolution image blending. In: *Proceedings of the 27th ACM International Conference on Multimedia*. pp. 2487–2495.

Wu, W., Qi, Z., Fuxin, L., 2019b. Pointconv: Deep convolutional networks on 3d point clouds. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 9621–9630.

Wurm, M., Stark, T., Zhu, X. X., Weigand, M., Taubenböck, H., 2019. Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing* 150, pp. 59–69.

Xia, G.-S., Hu, J., Hu, F., Shi, B., Bai, X., Zhong, Y., Zhang, L., Lu, X., 2017. AID: A benchmark data set for performance evaluation of aerial scene classification. *IEEE Transactions on Geoscience and Remote Sensing* 55 (7), pp. 3965–3981.

Xia, G.-S., Yang, W., Delon, J., Gousseau, Y., Sun, H., Maître, H., 2010. Structural high-resolution satellite image indexing. In: *ISPRS TC VII Symposium-100 Years ISPRS*. Vol. 38. pp. 298–303.

Xinqiao, L., Huadong, G., Yun, S., 1997. Detection of the Great Wall using SIR-C data in north-western China. In: *IEEE International Geoscience and Remote Sensing Symposium Proceedings. Remote Sensing-A Scientific Vision for Sustainable Development*. Vol. 1. IEEE, pp. 50–52.

Xu, D., Anguelov, D., Jain, A., 2018a. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 244–253.

Xu, Y., Wu, L., Xie, Z., Chen, Z., 2018b. Building extraction in very high resolution remote sensing imagery using deep learning and guided filters. *Remote Sensing* 10 (1), pp. 144.

Yaman, T. T., 2019. A model-based statistical classification analysis for Karamattepe arrowheads. *Journal of Computer Applications in Archaeology* 2 (1).

Yan, X., Zheng, C., Li, Z., Wang, S., Cui, S., 2020. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 5589–5598.

Yang, G., Huang, X., Hao, Z., Liu, M.-Y., Belongie, S., Hariharan, B., 2019a. Pointflow: 3d point cloud generation with continuous normalizing flows. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 4541–4550.

Yang, X., He, X., Liang, Y., Yang, Y., Zhang, S., Xie, P., 2020a. Transfer learning or self-supervised learning? a tale of two pretraining paradigms. *arXiv e-prints*, pp. arXiv–2007.

Yang, Y., Newsam, S., 2010. Bag-of-visual-words and spatial extensions for land-use classification. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. pp. 270–279.

Yang, Z., Jiang, W., Xu, B., Zhu, Q., Jiang, S., Huang, W., 2017. A convolutional neural network-based 3D semantic labeling method for ALS point clouds. *Remote Sensing* 9 (9), pp. 936.

Yang, Z., Sun, Y., Liu, S., Jia, J., 2020b. 3dssd: Point-based 3d single stage object detector. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 11040–11048.

Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J., 2019b. Std: Sparse-to-dense 3d object detector for point cloud. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 1951–1960.

Yoeli, P., 1983. Digital terrain models and their cartographic and cartometric utilisation. *The Cartographic Journal* 20 (1), pp. 17–22.

Yokoyama, R., Shirasawa, M., Pike, R. J., 2002. Visualizing topography by openness: a new application of image processing to digital elevation models. *Photogrammetric Engineering and Remote Sensing* 68 (3), pp. 257–266.

Yosinski, J., Clune, J., Bengio, Y., Lipson, H., 2014. How transferable are features in deep neural networks? In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. MIT Press, Cambridge, MA, USA, p. 3320–3328.

Yu, T., Meng, J., Yuan, J., 2018. Multi-view harmonized bilinear network for 3d object recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 186–194.

Yu, X., Hyyppä, H., Kaartinen, H., Hyyppä, J., Ahokas, E., Kaasalainen, S., 2005. Applicability of first pulse derived digital terrain models for boreal forest studies. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 36 (3/W19), pp. 97–102.

Zakšek, K., Oštir, K., Kokalj, Ž., 2011. Sky-View Factor as a relief visualization technique. *Remote Sensing* 3 (2), pp. 398–415.

Zhang, C., Sargent, I., Pan, X., Li, H., Gardiner, A., Hare, J., Atkinson, P. M., 2019a. Joint deep learning for land cover and land use classification. *Remote Sensing of Environment* 221, pp. 173–187.

Zhang, F., Du, B., Zhang, L., Xu, M., 2016a. Weakly supervised learning based on coupled convolutional neural networks for aircraft detection. *IEEE Transactions on Geoscience and Remote Sensing* 54 (9), pp. 5553–5563.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D. N., 2017a. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In: *Proceedings of the IEEE International Conference on Computer Vision*. pp. 5907–5915.

Zhang, J., Hu, X., Dai, H., 2021. Unsupervised learning of ALS point clouds for 3-D terrain scene clustering. *IEEE Geoscience and Remote Sensing Letters*, pp. 1–5.

Zhang, L., Qi, G.-J., Wang, L., Luo, J., 2019b. Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 2547–2555.

Zhang, L., Zhang, L., Du, B., 2016b. Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geoscience and Remote Sensing Magazine* 4 (2), pp. 22–40.

Zhang, M., Hu, X., Zhao, L., Lv, Y., Luo, M., Pang, S., 2017b. Learning dual multi-scale manifold ranking for semantic segmentation of high-resolution images. *Remote Sensing* 9 (5), pp. 500.

Zhang, P., Ke, Y., Zhang, Z., Wang, M., Li, P., Zhang, S., 2018. Urban land use and land cover classification using novel deep learning models based on high spatial resolution satellite imagery. *Sensors* 18 (11), pp. 3717.

Zhang, R., Isola, P., Efros, A. A., 2016c. Colorful image colorization. In: *European Conference on Computer Vision*. Springer, pp. 649–666.

Zhang, S., He, G., Chen, H.-B., Jing, N., Wang, Q., 2019c. Scale adaptive proposal network for object detection in remote sensing images. *IEEE Geoscience and Remote Sensing Letters* 16 (6), pp. 864–868.

Zhang, W., Tang, P., Zhao, L., 2019d. Remote sensing image scene classification using CNN-CapsNet. *Remote Sensing* 11 (5), pp. 494.

Zhang, Y., Rabbat, M., 2018. A graph-cnn for 3d point cloud classification. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 6279–6283.

Zhang, Z., Hua, B.-S., Yeung, S.-K., 2019e. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 1607–1616.

Zhang, Z., Zhang, L., Tan, Y., Zhang, L., Liu, F., Zhong, R., 2018. Joint discriminative dictionary and classifier learning for ALS point cloud classification. *IEEE Transactions on Geoscience and Remote Sensing* 56 (1), pp. 524–538.

Zhao, C., Guo, H., Lu, J., Yu, D., Li, D., Chen, X., 2020. ALS point cloud classification with small training data set based on transfer learning. *IEEE Geoscience and Remote Sensing Letters* 17 (8), pp. 1406–1410.

Zhao, R., Pang, M., Wang, J., 2018. Classifying airborne LiDAR point clouds via deep features learned by a multi-scale convolutional neural network. *International Journal of Geographical Information Science* 32 (5), pp. 960–979.

Zhu, J.-Y., Park, T., Isola, P., Efros, A. A., 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *Proceedings of the IEEE International Conference on Computer Vision*. pp. 2223–2232.

Acknowledgments

This dissertation would not have been possible without the support and encouragement of many wonderful individuals. Firstly, I would like to thank my advisor, Prof. Monika Sester, for giving the opportunity to work with her and for guiding me throughout my research and the dissertation. She gave me the freedom to do research on what excites me, but she motivated me to work harder and gave constant feedback and valuable comments on my work. I could not have asked for a better advisor.

My time at Institute of Cartography and Geoinformatics was amazing thanks to my wonderful colleagues. They all made me feel welcome and at home. We had fruitful discussions at work and made great memories outside of work. I would specially like to thank my friend, colleague and office mate, Torben Peters, for all the great discussions about our research together, and for his comments and reviews of my dissertation. Additionally, I would like to thank Frank Thiemann and Dr. Katharina Malek for helping me with the datasets used in this research.

The support and encouragement of my parents (Mohammad Tahir Khan and Bakhtawar Kazimi), my brothers (Dr. Mohammad Nadir, Abdul Qadir, Mohammad Saber and Mohammad Nazir Kazimi), my sisters (Sakina and Zakia Kazimi), and other family members and friends were overwhelming at every stage of my life. I would specially like to acknowledge the relentless support of my brother, Dr. Mohammad Nadir Kazimi. I would not have made it this far if it were not for his support.

A major part of this work is the result of sacrifices my wife, Saba Jannat, had to make throughout the years. She constantly kept me motivated to write the dissertation while she suffered my rants and meltdowns and her being alone. I would like to show my sincere gratitude and hope I can make up for it.

I would also like to thank Prof. Franz Rottensteiner and Prof. Kourosh Khoshelham for agreeing to review my dissertation. I had the honor of taking the Image Analysis course by Prof. Rottensteiner and the privilege of a research visit under the guidance of Prof. Khoshelham.

Finally, I would like to acknowledge that this research was funded by the Lower Saxony Ministry of Science and Culture through the "Niedersächsisches Vorab" funding initiative and with the collaboration of Lower Saxony State Office for Heritage (Niedersächsisches Landesamt für Denkmalpflege).

Resume

Personal Details

Bashir Kazimi

Born in Ghazni, Afghanistan on 04.07.1991

Education

1998-2002: Korala High School, Ghazni, Afghanistan

2002-2003: Gholam Sakhi Shahid High School, Ghazni, Afghanistan

2003-2004: Bakhtar High School, Balkh, Afghanistan

2004-2007: International Afghan Turk High School, Balkh, Afghanistan

2007-2008: Kenton High School, Ohio, United States of America

2008-2009: International Afghan Turk High School, Balkh, Afghanistan

2009-2010: Ankara University TOMER, Bursa, Turkey

2010-2015: Middle East Technical University, Ankara, Turkey, B.Sc. in Computer Engineering

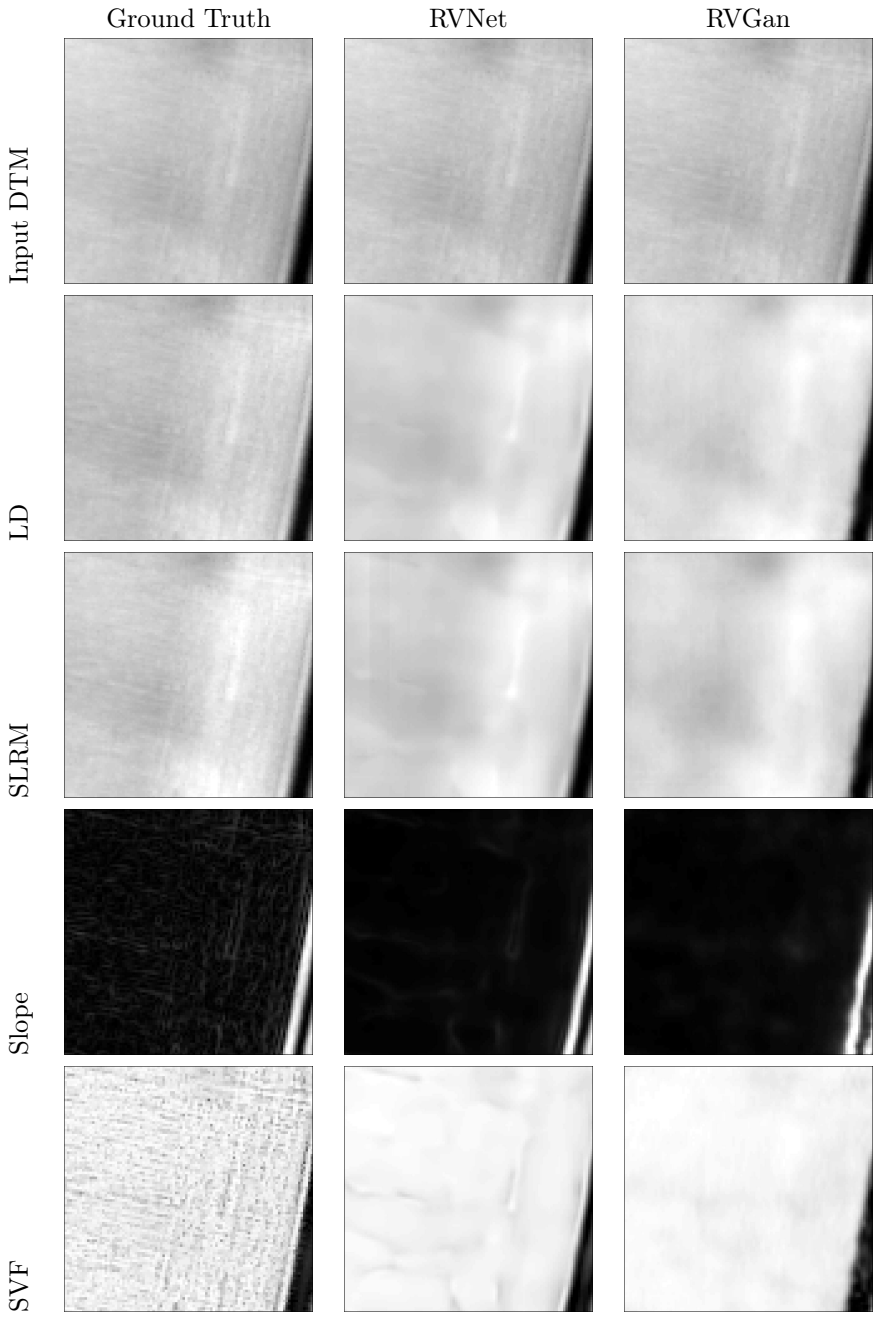
2015-2017: Polytechnic University of Catalonia, Barcelona, Spain, M.Sc. in Artificial Intelligence

Occupation

Since 2017: Researcher at Institute of Cartography and Geoinformatics, Leibniz University Hannover

A. Appendix

A.1. Self Supervised Learning Pretext



Continued on next page

Table A.1 – continued from previous page

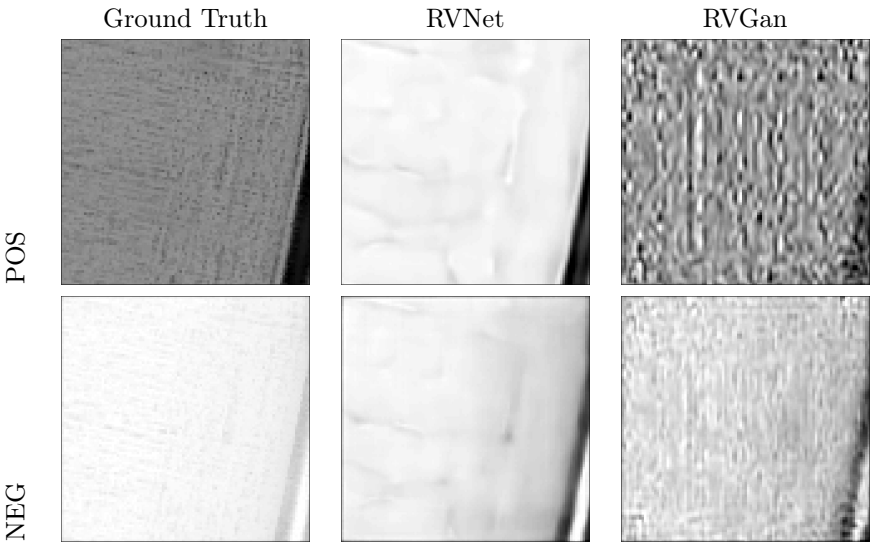
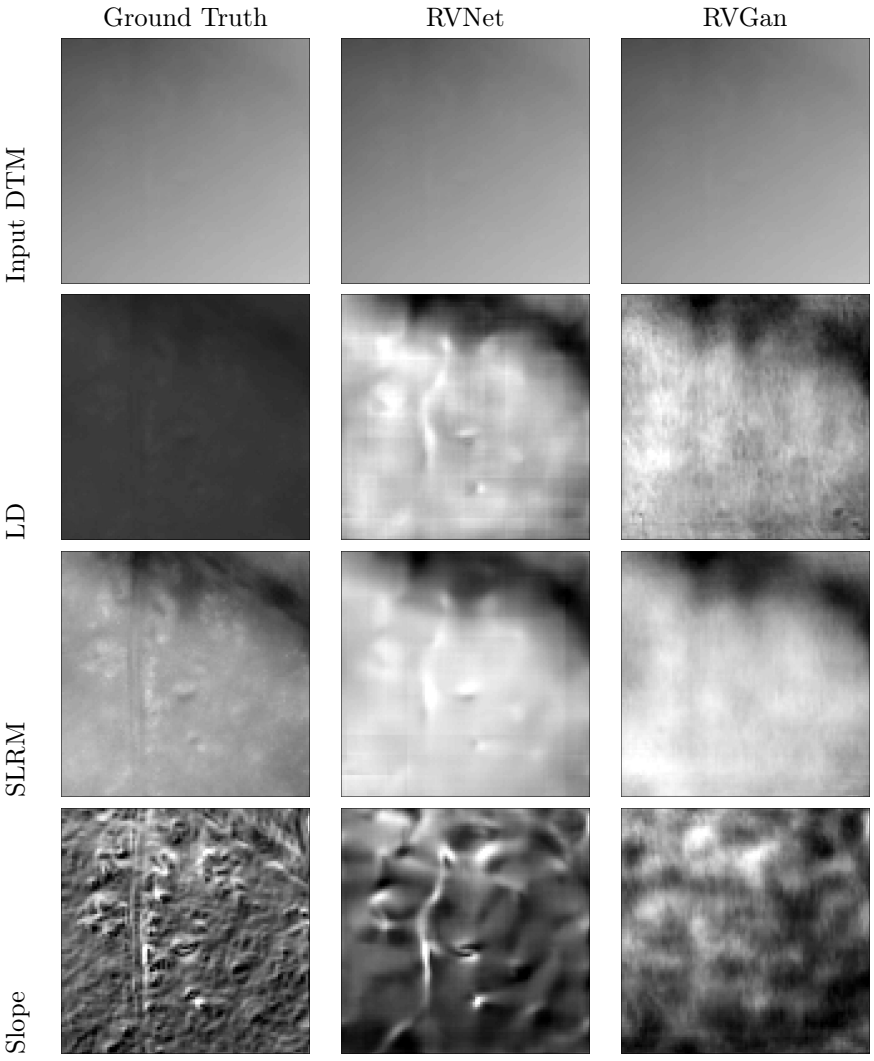


Table A.1.: Examples of relief visualization rasters by RVNet and RVGan



Continued on next page

Table A.2 – continued from previous page

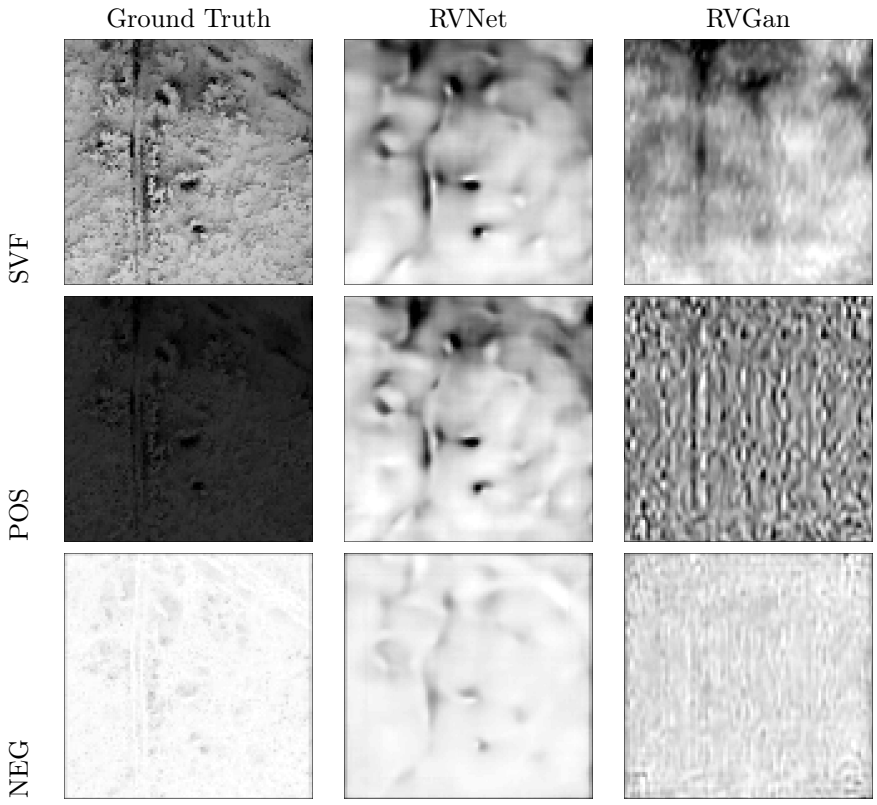
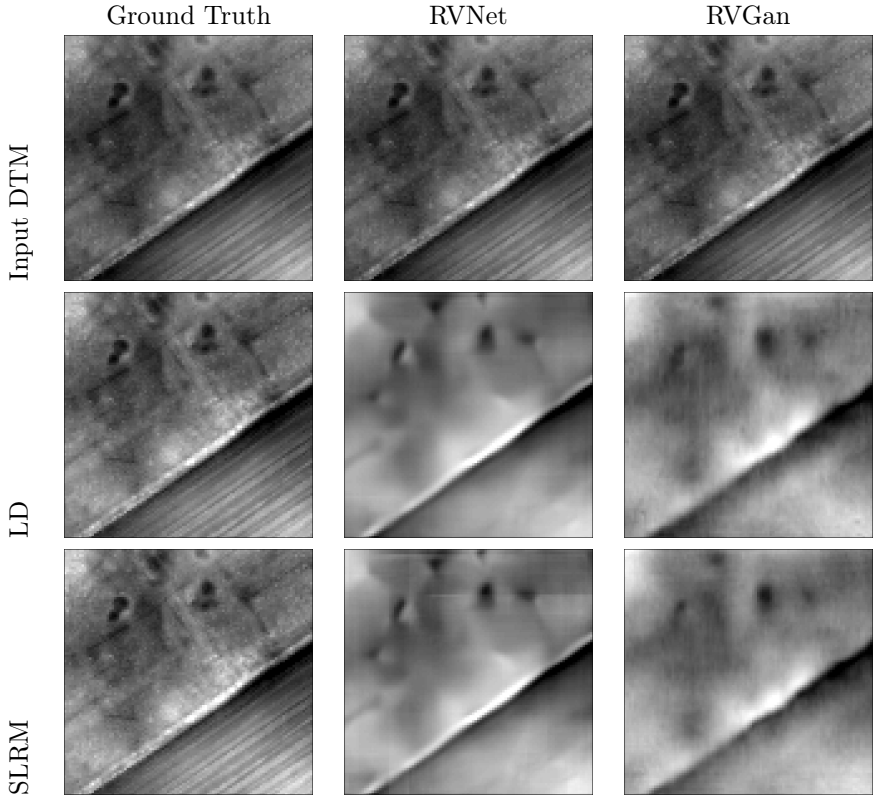


Table A.2.: Examples of relief visualization rasters by RVNet and RVGan



Continued on next page

Table A.3 – continued from previous page

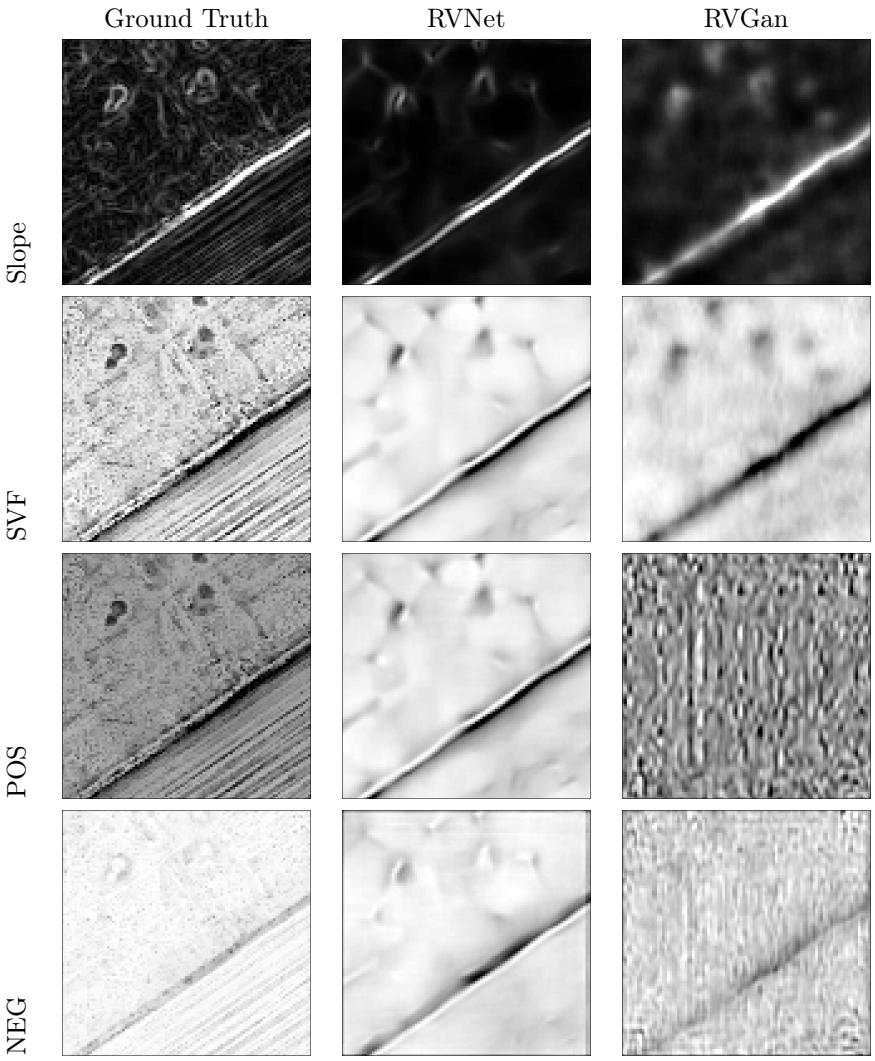
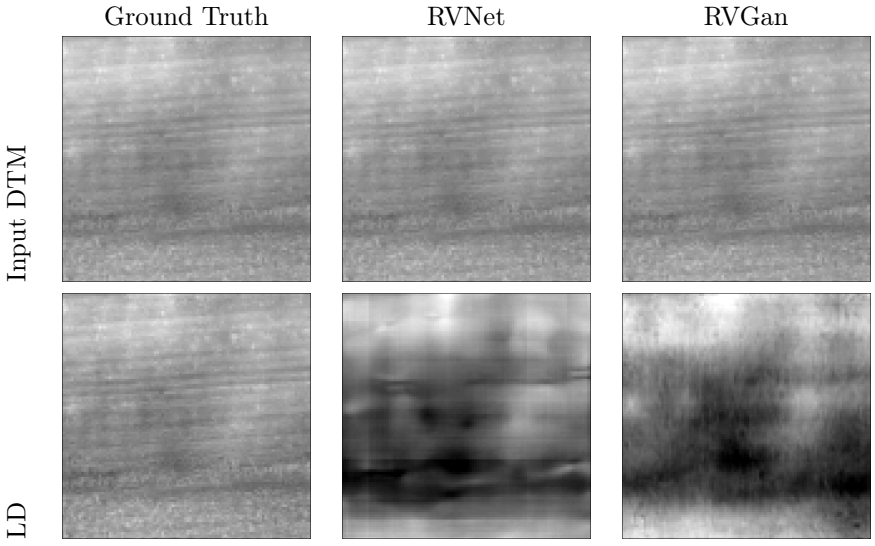


Table A.3.: Examples of relief visualization rasters by RVNet and RVGan



Continued on next page

Table A.4 – continued from previous page

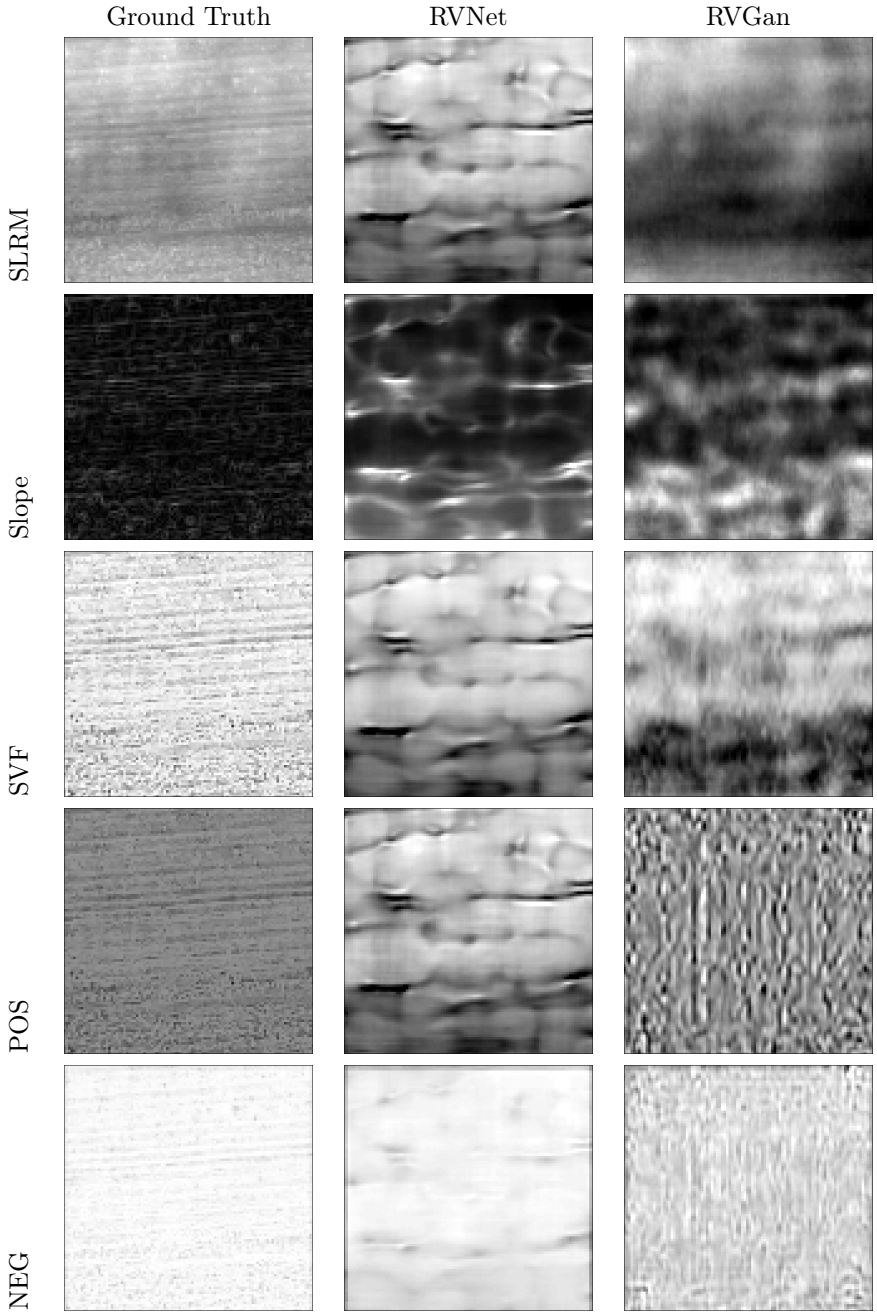
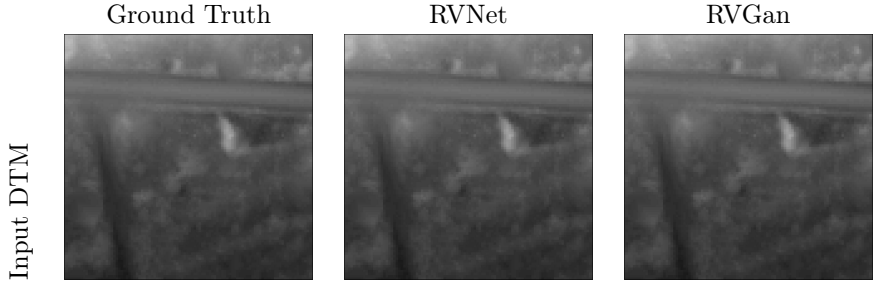


Table A.4.: Examples of relief visualization rasters by RVNet and RVGan



Continued on next page

Table A.5 – continued from previous page

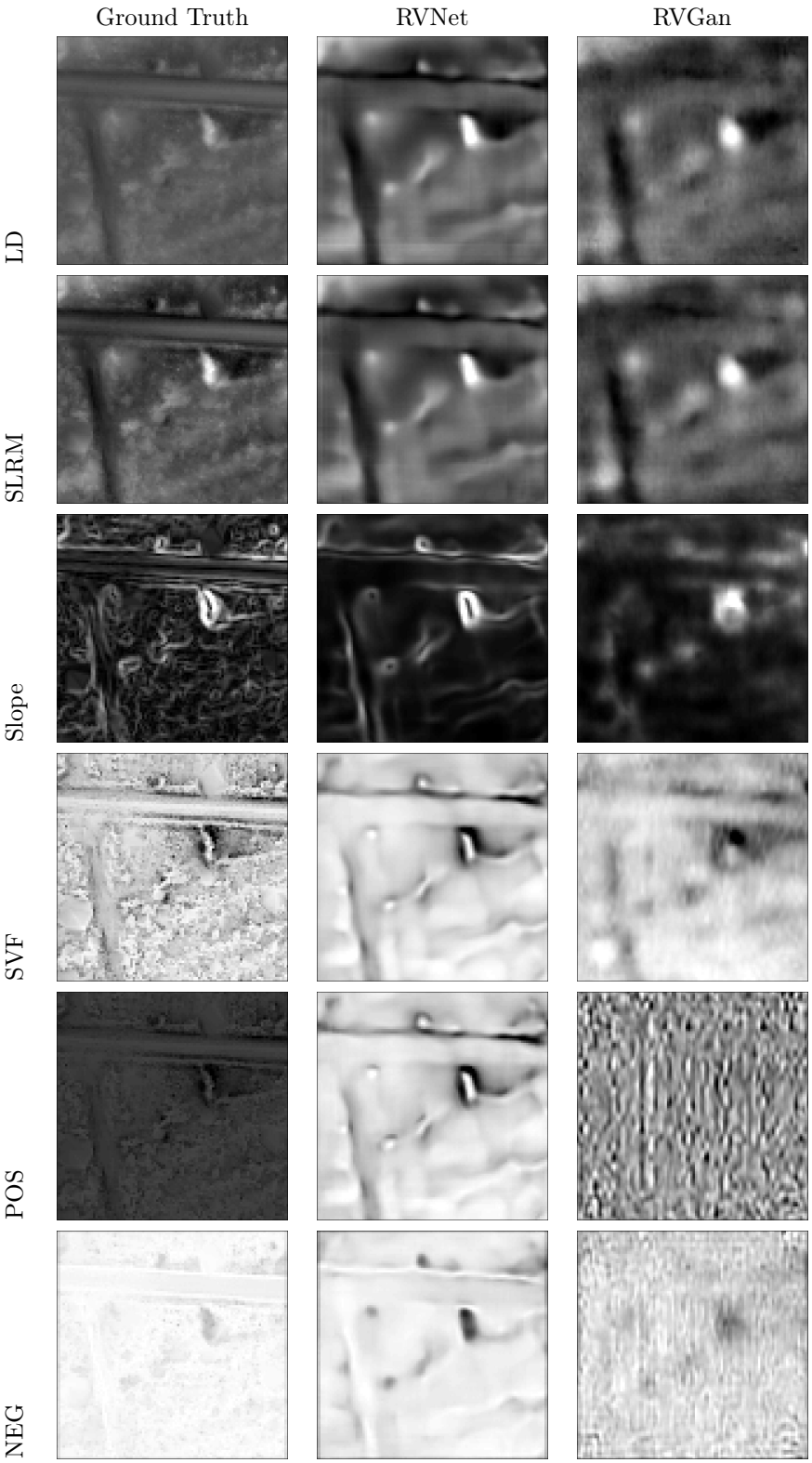
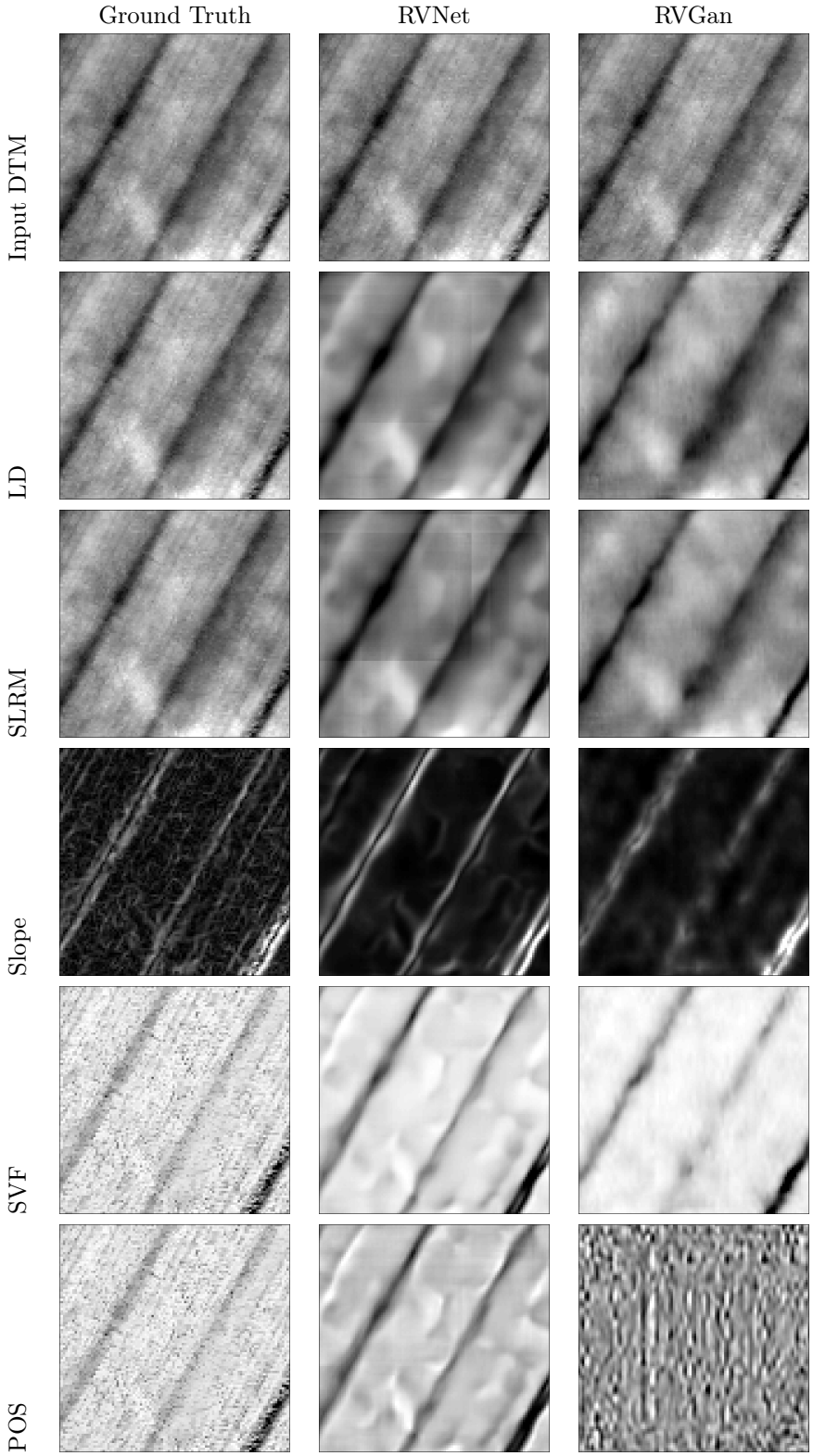


Table A.5.: Examples of relief visualization rasters by RVNet and RVGan



Continued on next page

Table A.6 – continued from previous page

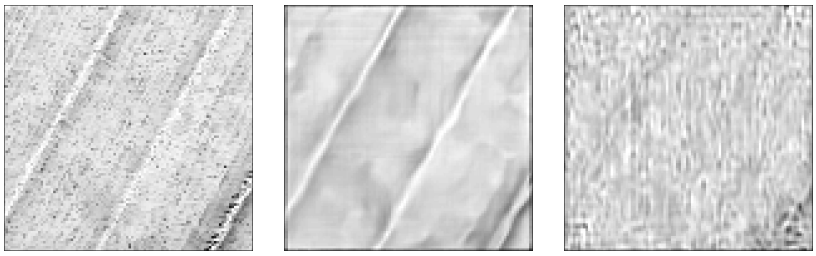
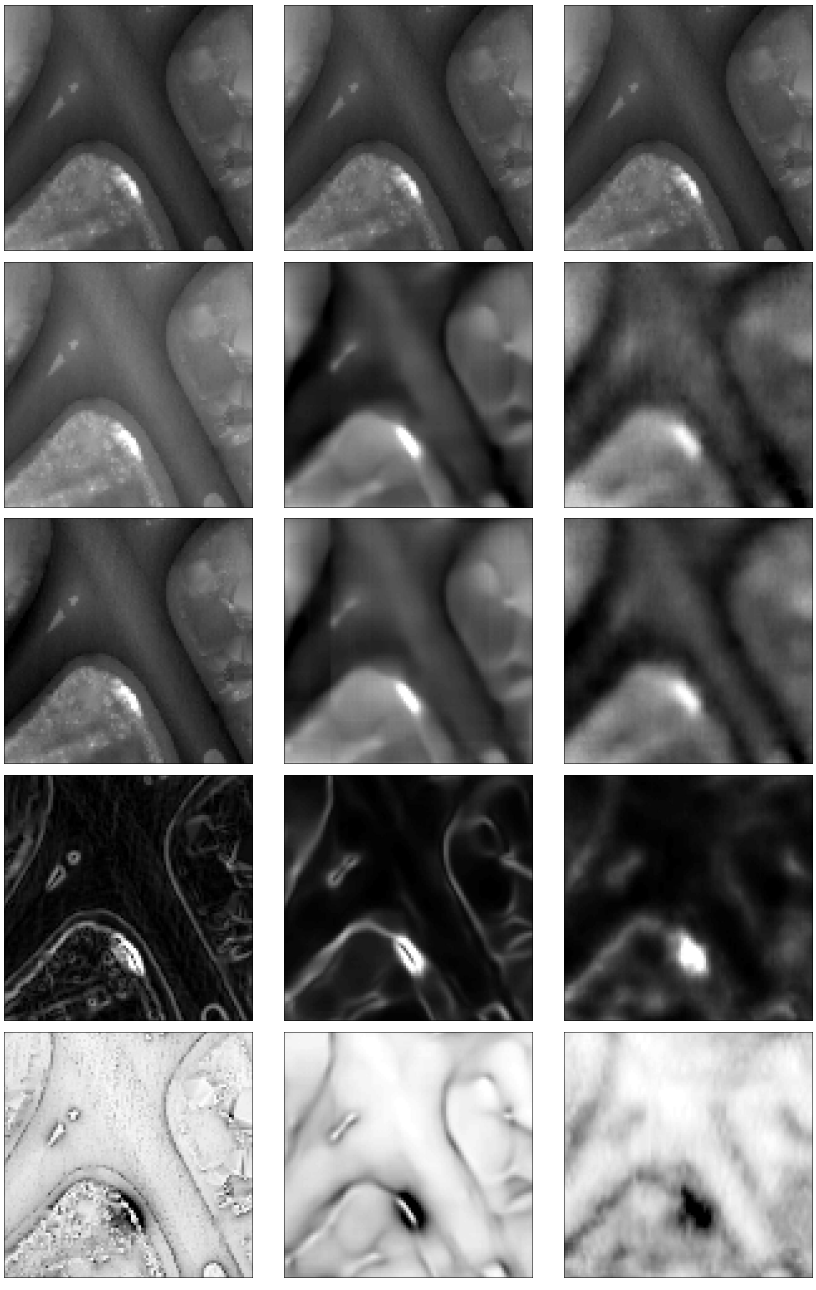
	Ground Truth	RVNet	RVGan
NEG			

Table A.6.: Examples of relief visualization rasters by RVNet and RVGan

	Ground Truth	RVNet	RVGan
Input DTM			
LD			
SLRM			
Slope			
SVF			

Continued on next page

Table A.7 – continued from previous page

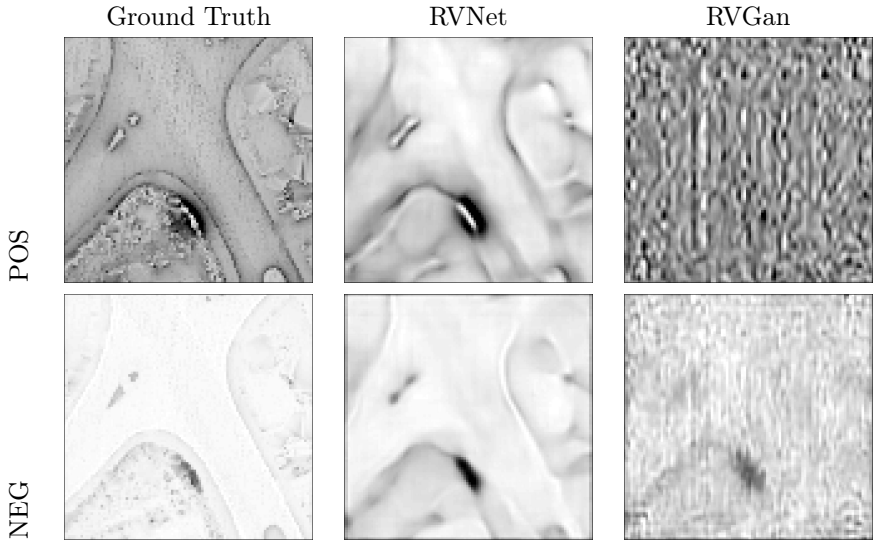
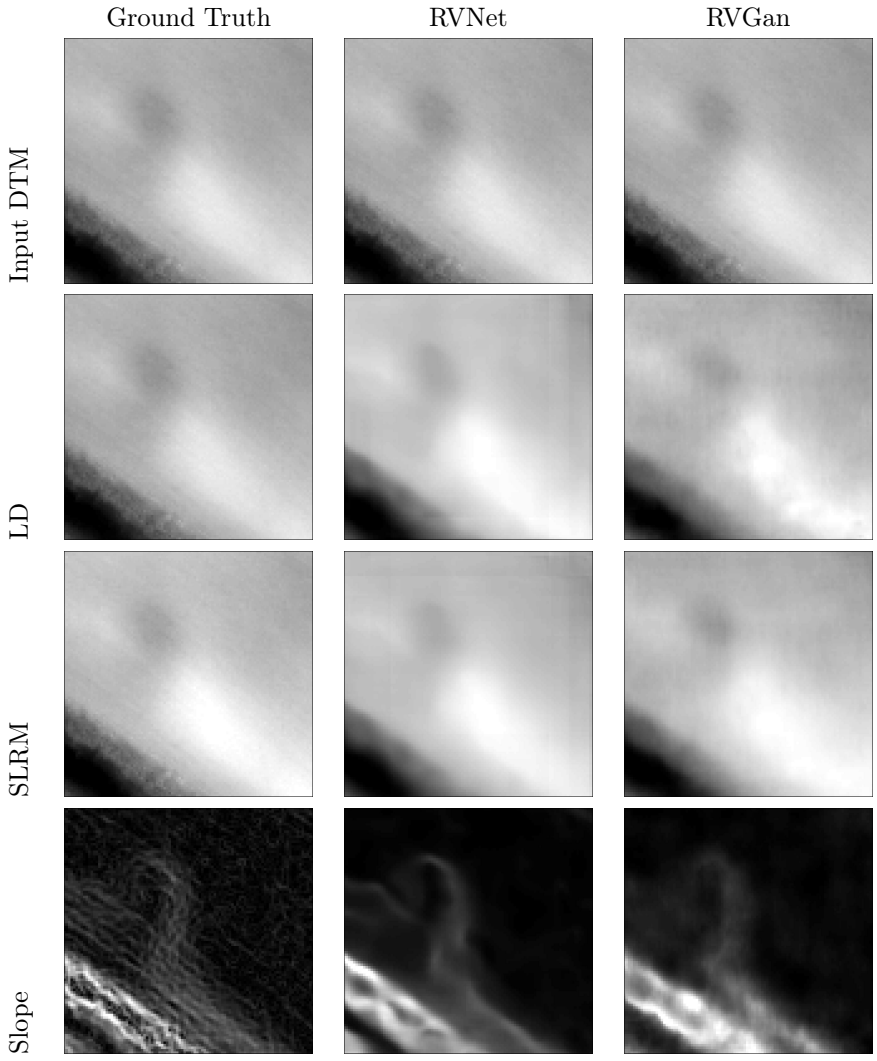


Table A.7.: Examples of relief visualization rasters by RVNet and RVGan



Continued on next page

Table A.8 – continued from previous page

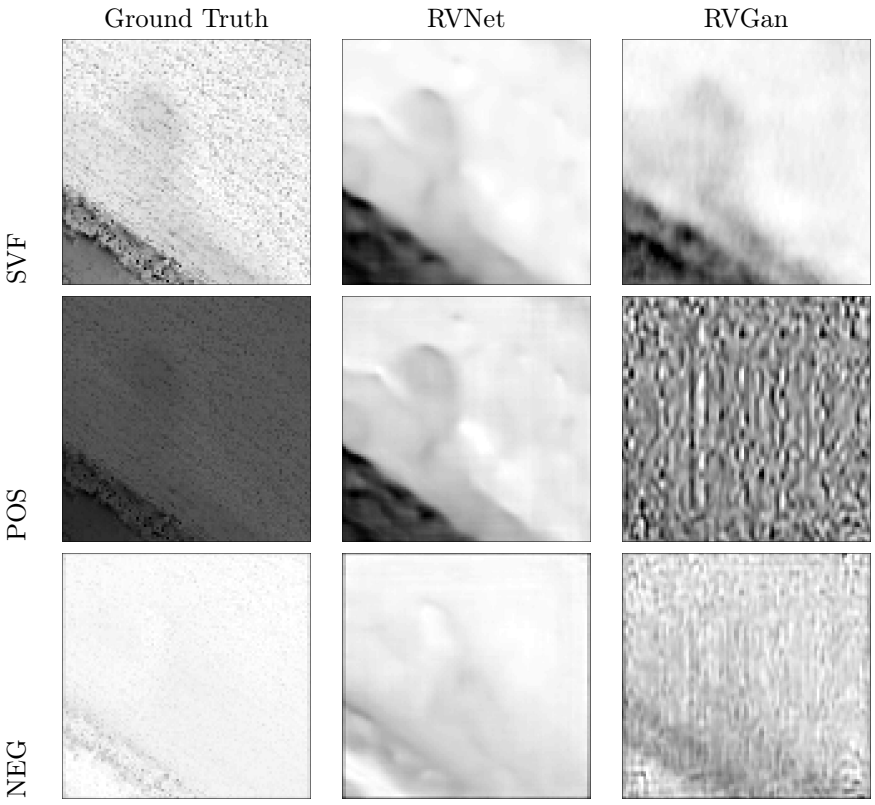
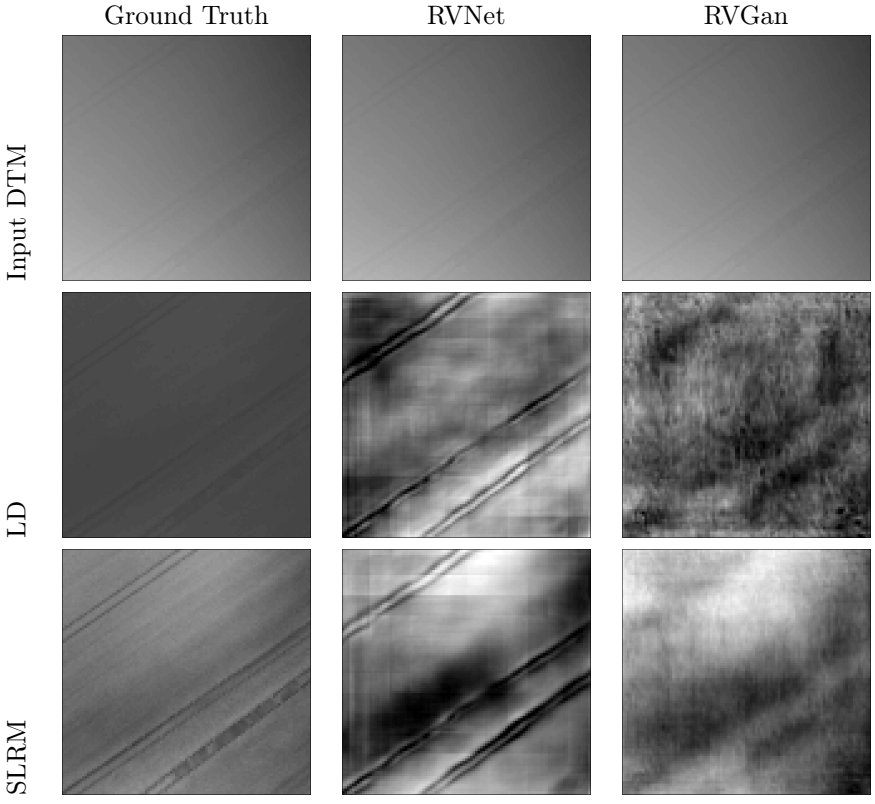


Table A.8.: Examples of relief visualization rasters by RVNet and RVGan



Continued on next page

Table A.9 – continued from previous page

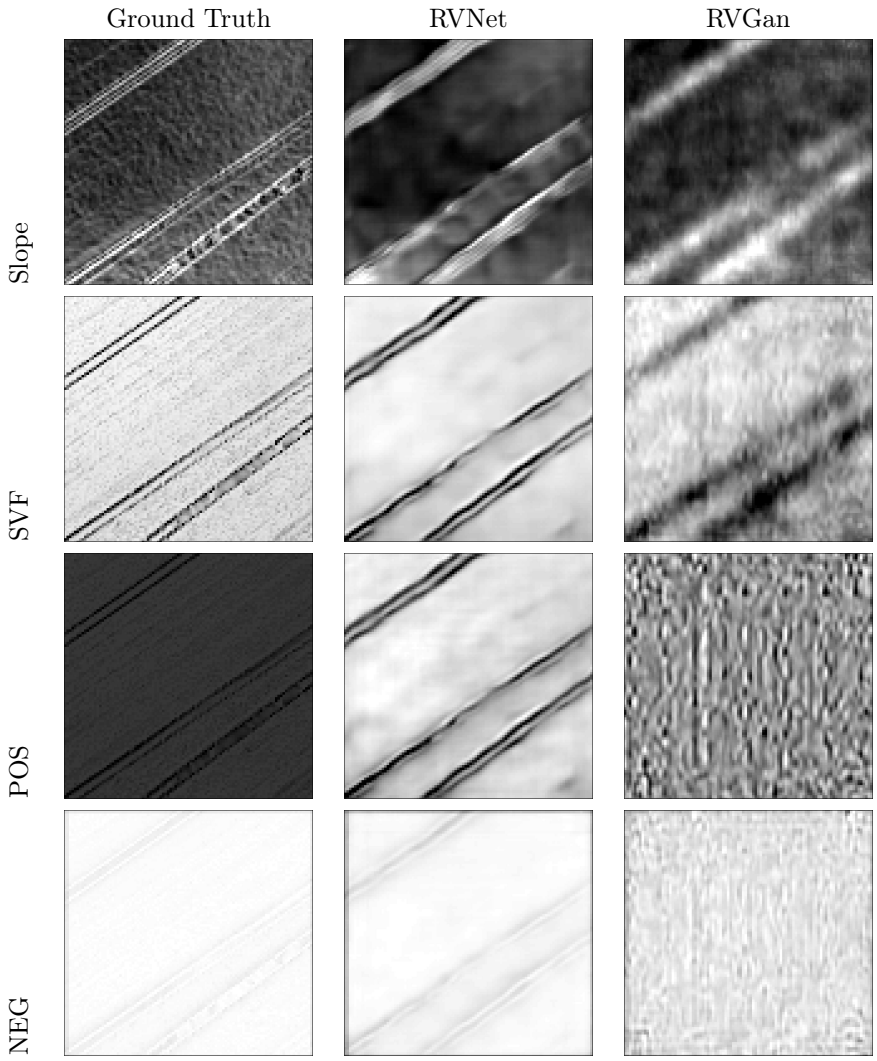


Table A.9.: Examples of relief visualization rasters by RVNet and RVGan

A.2. Classification

Actual \ Predicted	BG	BC	CK	BM	MH
BG	90.00	0.53	1.73	2.40	5.33
BC	20.09	47.03	1.83	0.00	30.59
CK	7.60	0.11	83.54	0.23	8.29
BM	8.47	0.21	1.45	88.84	0.41
MH	4.15	5.68	1.26	0.81	87.74

(a) HRNet with random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	80.40	1.33	5.07	5.33	7.87
BC	12.79	52.97	6.39	0.91	26.48
CK	9.53	0.68	59.82	1.48	28.26
BM	10.12	0.21	1.45	85.54	2.07
MH	4.24	8.48	2.16	0.63	84.13

(b) ResNet with random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	87.73	1.07	0.93	2.93	7.33
BC	25.11	46.12	1.83	0.46	26.03
CK	7.15	0.11	68.33	1.48	22.70
BM	8.06	0.00	0.62	90.29	0.41
MH	3.70	5.32	0.81	1.17	88.64

(c) HRNet with RVNet weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	91.20	0.53	1.20	2.93	4.13
BC	29.22	44.29	3.65	0.46	21.92
CK	7.60	0.00	78.66	0.45	13.05
BM	7.64	0.21	0.21	91.12	0.21
MH	6.67	6.04	0.99	0.81	85.12

(d) HRNet with RVGan weights

Table A.10.: Confusion matrix for the HRNet with input dimensions of 32 and different weight initializations. Values are in percent.

Actual \ Predicted	BG	BC	CK	BM	MH
BG	94.93	0.67	0.40	1.73	2.27
BC	31.51	41.10	2.28	0.46	24.20
CK	2.04	0.00	89.22	0.00	8.51
BM	7.44	0.00	0.41	90.50	1.03
MH	4.87	5.23	0.54	0.90	88.10

(a) HRNet with random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	83.47	6.00	1.60	4.27	4.67
BC	4.11	72.60	1.83	0.91	20.09
CK	2.61	0.68	70.26	0.68	25.54
BM	5.79	2.07	0.83	89.46	1.24
MH	4.15	14.88	0.63	1.44	78.54

(b) ResNet with random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	94.40	0.67	1.20	1.20	2.53
BC	27.85	45.21	1.83	0.91	23.74
CK	2.16	0.00	79.34	0.23	18.05
BM	10.74	0.21	0.21	88.02	0.21
MH	3.70	5.86	0.90	1.53	87.65

(c) HRNet with RVNet weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	94.27	0.53	1.07	1.33	2.80
BC	32.42	36.99	2.28	1.37	26.48
CK	2.38	0.00	79.00	0.11	18.27
BM	8.26	0.21	0.41	89.26	1.24
MH	3.16	4.69	1.44	1.26	89.09

(d) HRNet with RVGan weights

Table A.11.: Confusion matrix for the HRNet with input dimensions of 64 and different weight initializations. Values are in percent.

Actual \ Predicted	BG	BC	CK	BM	MH
BG	95.73	0.53	0.40	1.47	1.87
BC	38.81	36.07	9.13	1.83	13.70
CK	3.86	0.00	82.07	0.34	13.51
BM	8.47	0.41	0.41	89.46	0.62
MH	4.78	4.60	1.26	0.99	88.01

(a) HRNet with random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	85.73	1.60	3.07	5.33	4.27
BC	9.59	56.62	9.13	4.57	19.63
CK	5.11	0.34	41.09	2.95	50.28
BM	3.72	0.41	0.83	93.80	0.62
MH	3.07	6.85	4.33	3.97	81.42

(b) ResNet with random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	96.53	0.40	0.53	0.93	1.60
BC	52.97	25.11	1.83	2.28	17.35
CK	18.96	0.57	52.89	1.25	26.11
BM	9.09	0.00	0.00	90.29	0.00
MH	9.02	5.05	1.71	2.07	81.79

(c) HRNet with RVNet weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	95.33	0.53	1.33	0.67	2.13
BC	31.51	34.25	2.74	1.37	29.68
CK	6.24	0.00	75.37	0.34	17.82
BM	8.88	0.00	0.21	88.64	1.65
MH	3.79	3.70	1.35	0.36	90.44

(d) HRNet with RVGan weights

Table A.12.: Confusion matrix for the HRNet with input dimensions of 96 and different weight initializations. Values are in percent..

Actual \ Predicted	BG	BC	CK	BM	MH
BG	95.87	0.40	0.80	0.93	2.00
BC	47.95	26.03	5.94	0.91	18.72
CK	5.45	0.00	76.16	0.57	17.59
BM	8.06	0.21	0.21	90.50	0.41
MH	4.78	3.52	3.07	0.63	87.65

(a) HRNet with random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	87.60	1.60	2.93	4.40	3.47
BC	9.59	67.12	12.79	3.20	6.85
CK	3.41	0.34	45.52	2.50	48.01
BM	4.13	0.21	1.45	92.98	0.62
MH	3.43	8.84	5.32	5.41	76.65

(b) ResNet with random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	96.80	0.40	0.53	0.40	1.87
BC	54.34	26.03	3.65	0.46	15.07
CK	7.83	0.00	72.08	0.45	19.41
BM	12.60	0.21	0.00	86.57	0.00
MH	2.98	4.42	3.97	0.90	87.38

(c) HRNet with RVNet weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	96.27	0.53	1.33	0.53	1.33
BC	45.21	28.31	5.02	0.46	20.55
CK	1.93	0.00	76.84	0.34	20.66
BM	8.68	0.21	1.03	88.84	0.62
MH	3.61	4.87	7.12	0.90	83.14

(d) HRNet with RVGan weights

Table A.13.: Confusion matrix for the HRNet with input dimensions of 128 and different weight initializations. Values are in percent.

Actual \ Predicted	BG	BC	CK	BM	MH
BG	97.47	0.13	0.53	0.67	1.20
BC	60.27	11.42	2.28	0.00	25.57
CK	14.30	0.00	60.27	0.45	24.74
BM	5.79	0.00	0.00	92.56	1.03
MH	4.51	3.61	0.63	0.18	90.71

(a) HRNet with random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	82.53	1.73	1.33	12.00	2.40
BC	10.50	59.82	5.94	7.76	15.53
CK	2.61	1.25	80.82	1.25	13.85
BM	2.07	0.83	0.41	95.45	0.62
MH	3.25	7.30	5.50	11.54	72.05

(b) ResNet with random weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	98.40	0.13	0.27	0.13	1.07
BC	49.77	26.94	0.46	0.00	22.37
CK	2.38	0.00	91.37	0.00	6.02
BM	14.26	0.00	0.00	84.71	0.41
MH	3.34	4.60	0.27	0.00	91.43

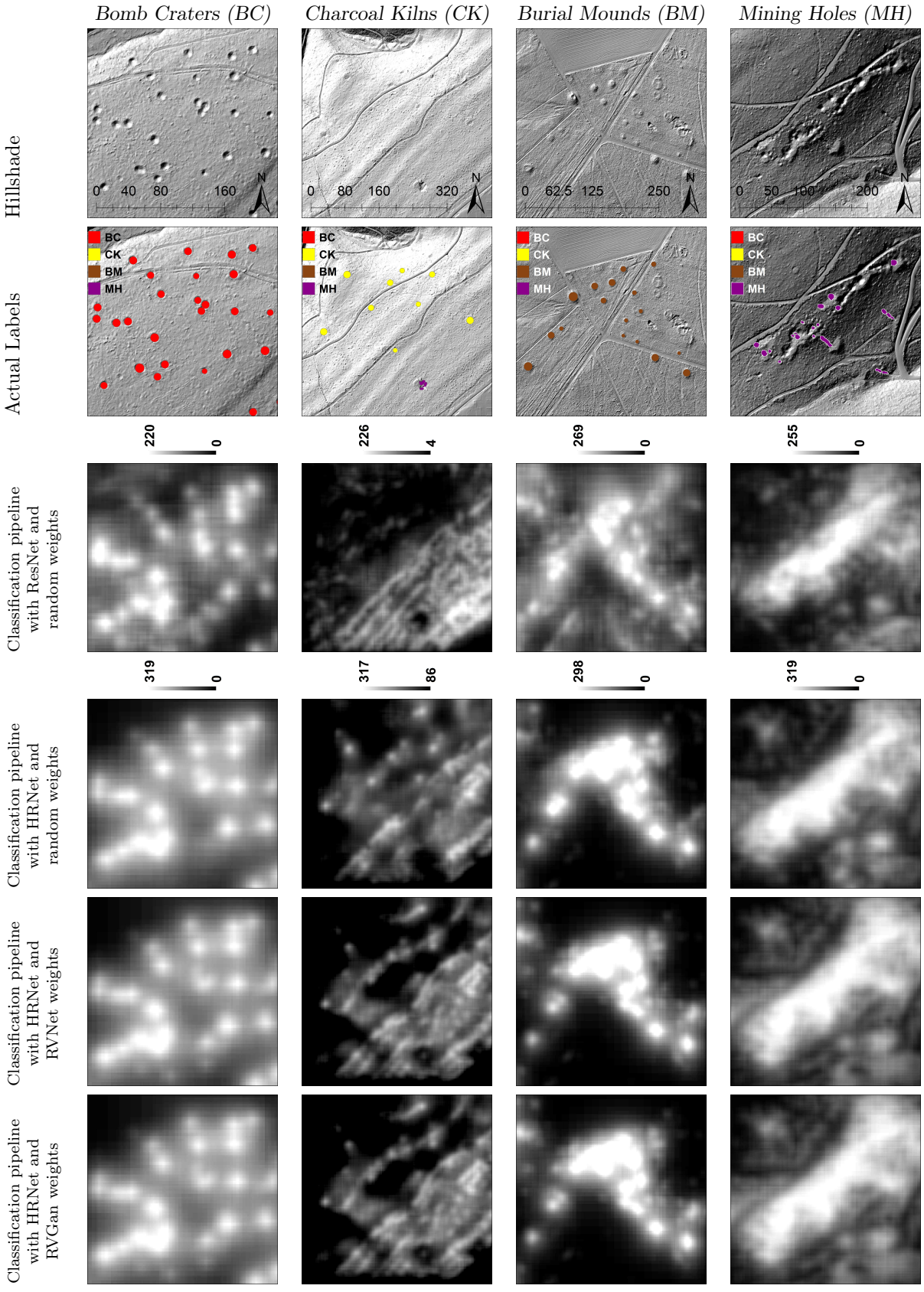
(c) HRNet with RVNet weights

Actual \ Predicted	BG	BC	CK	BM	MH
BG	98.40	0.13	0.00	0.40	1.07
BC	55.71	16.89	0.46	0.00	26.48
CK	6.47	0.00	87.85	0.11	5.33
BM	8.68	0.21	0.21	89.67	0.62
MH	4.87	5.59	0.27	0.09	88.82

(d) HRNet with RVGan weights

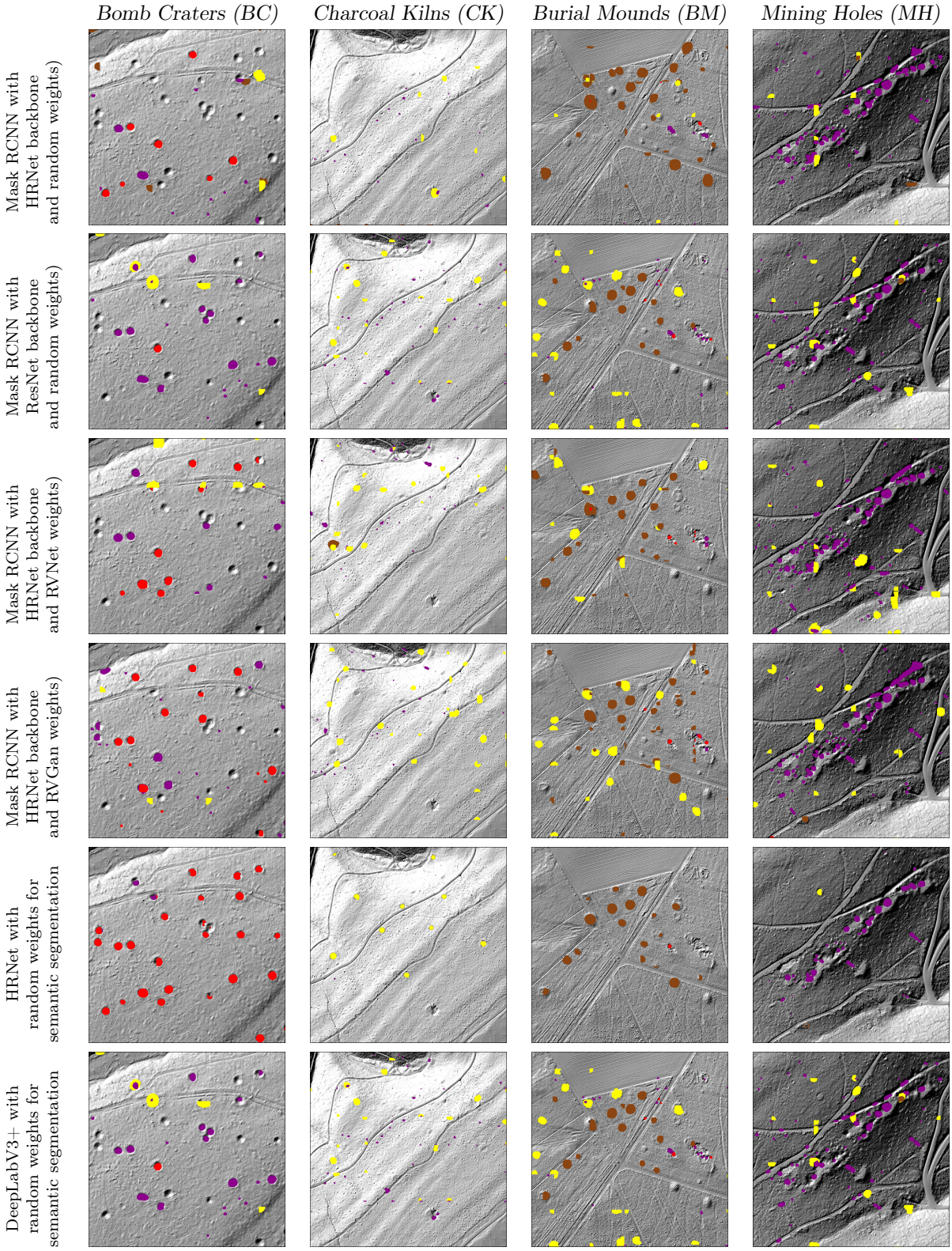
Table A.14.: Confusion matrix for the HRNet with input dimensions of 224 and different weight initializations. Values are in percent.

A.3. Areal Dataset



Continued on next page

Table A.15 – continued from previous page



Continued on next page

Table A.15 – continued from previous page

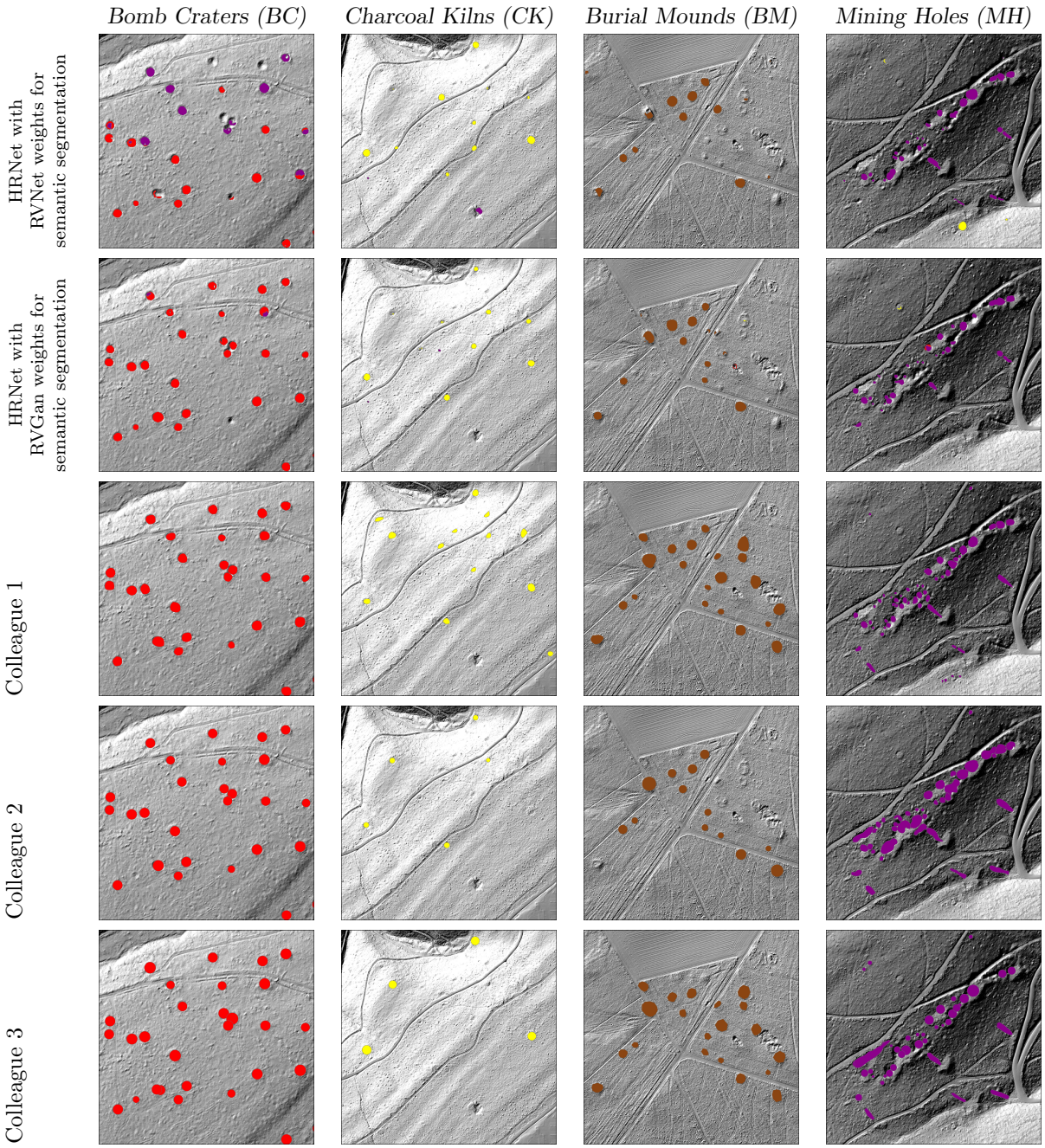
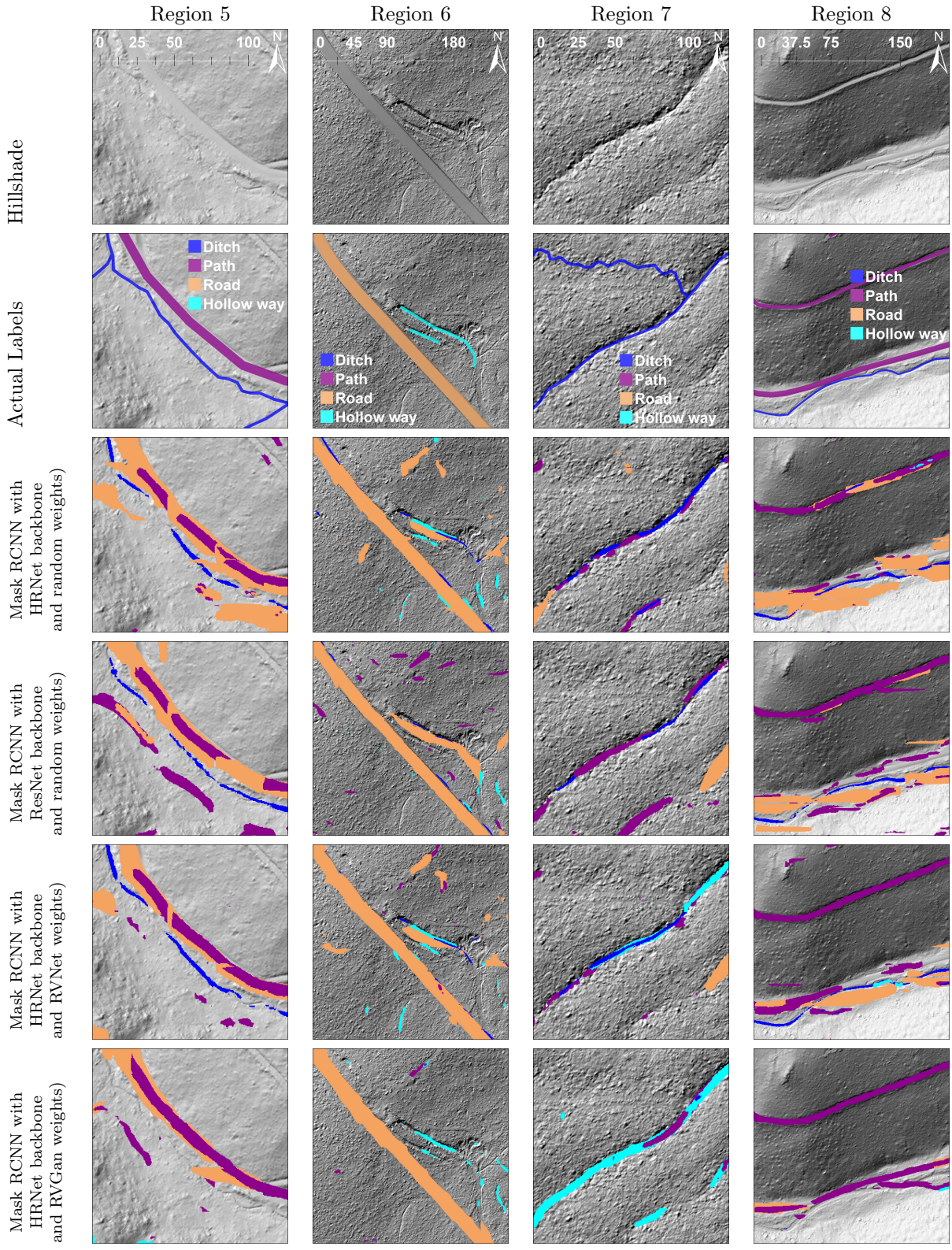


Table A.15.: Predictions for 4 regions in the areal dataset by the trained models in three approaches and manual annotations by 3 colleagues.

A.4. Linear Dataset



Continued on next page

Table A.16 – continued from previous page

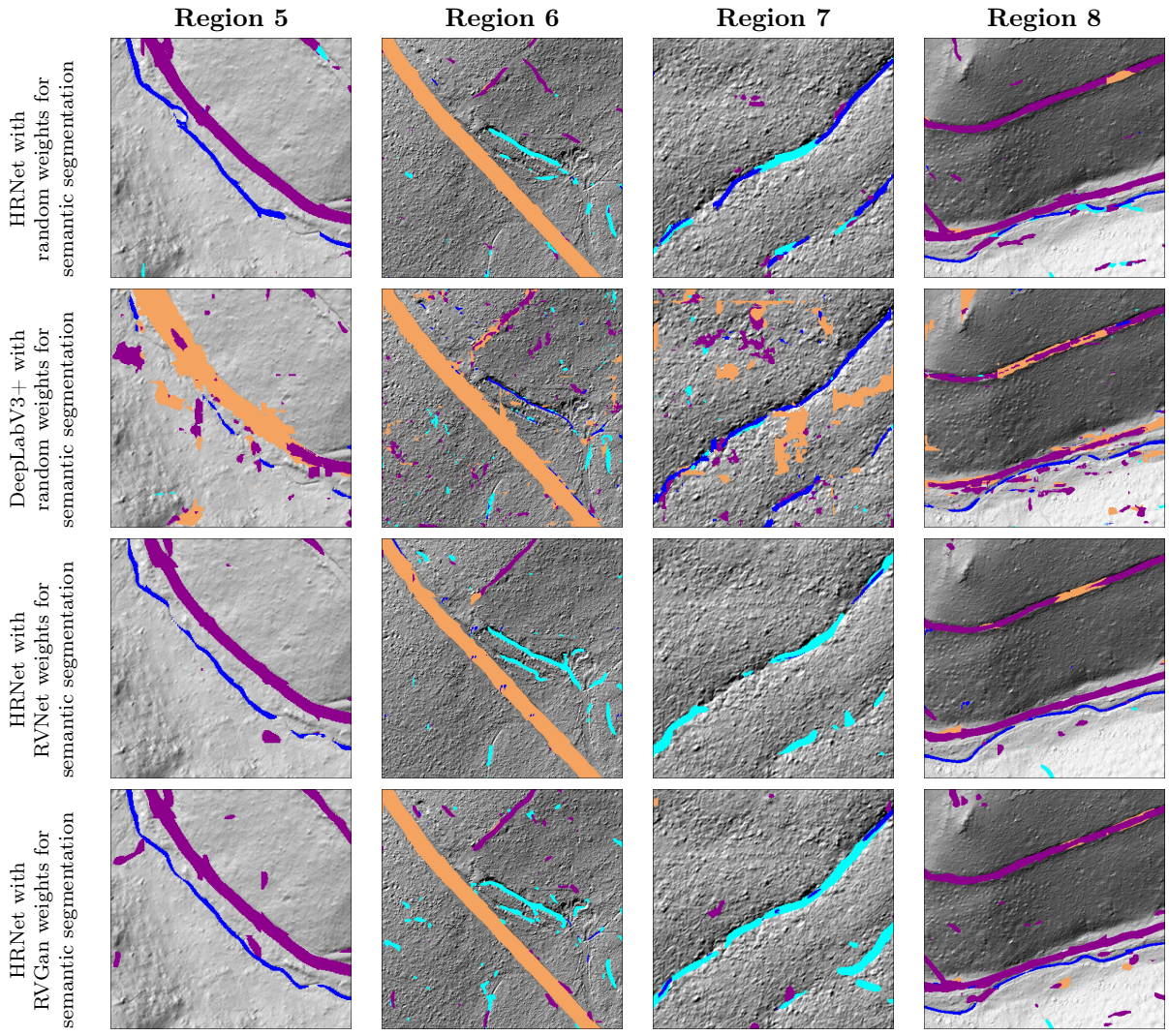
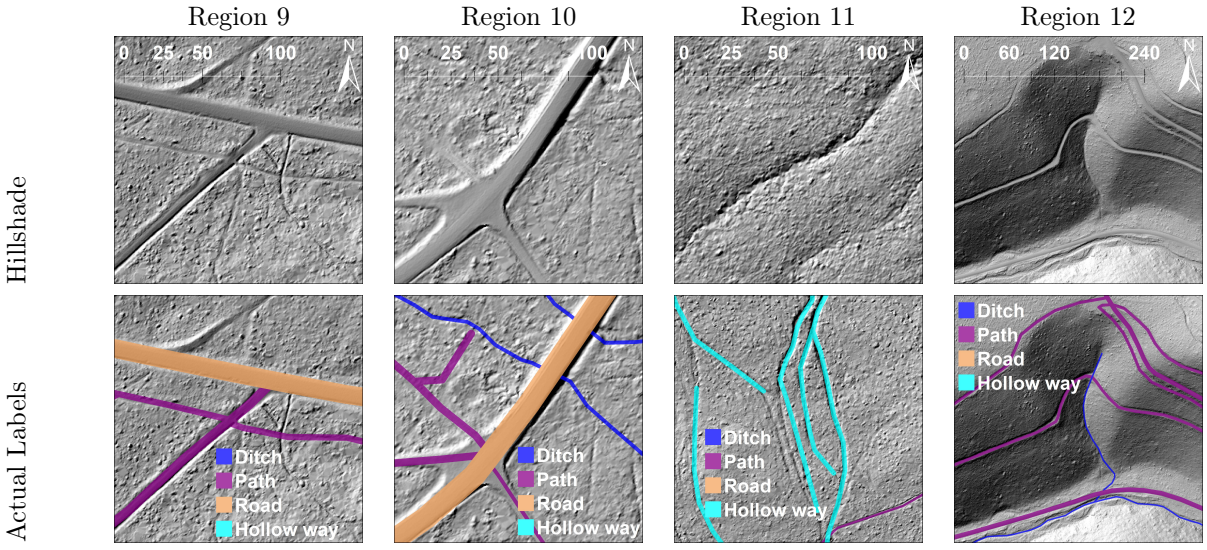
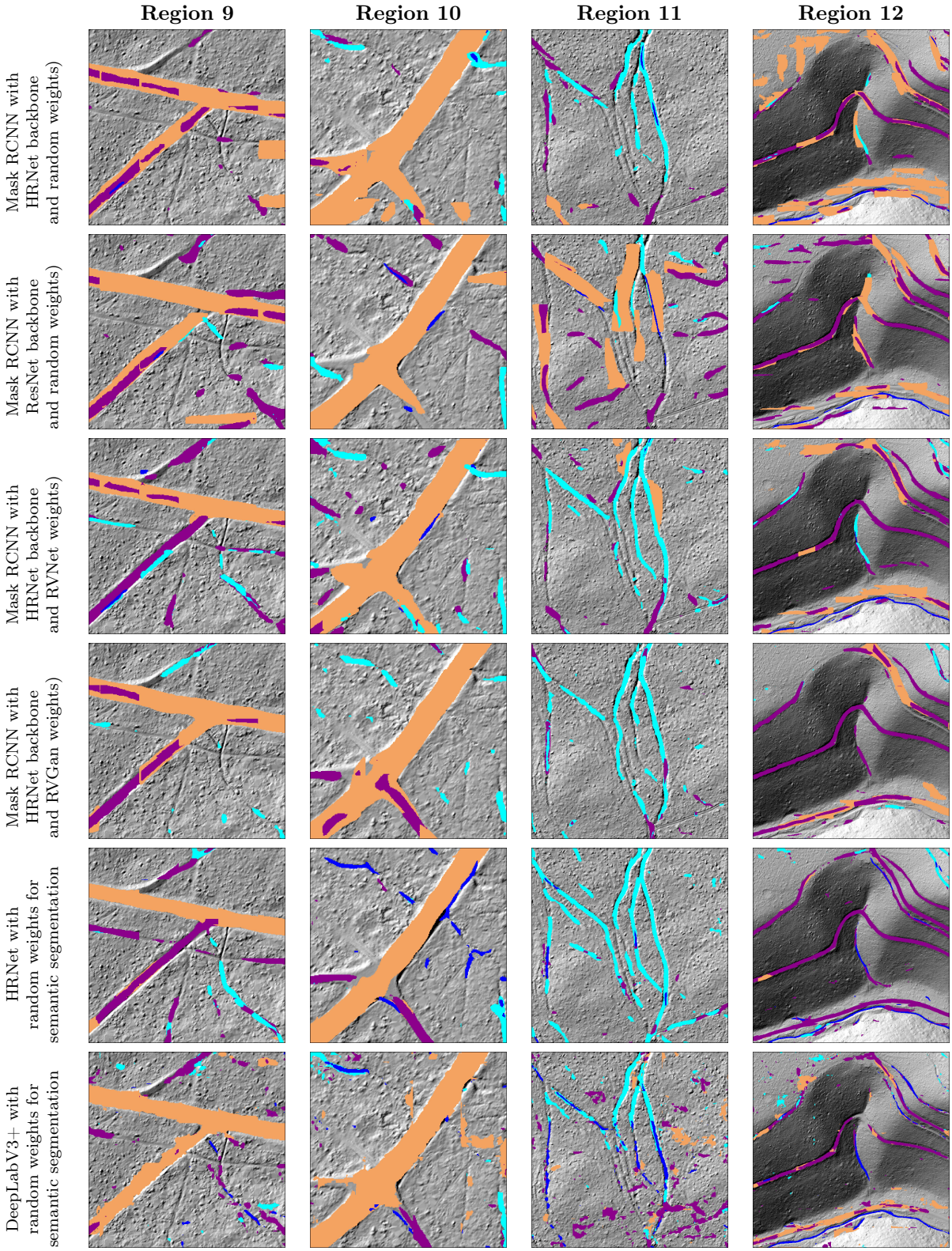


Table A.16.: Predictions for 4 regions by the trained models in the linear dataset.



Continued on next page

Table A.17 – continued from previous page



Continued on next page

Table A.17 – continued from previous page

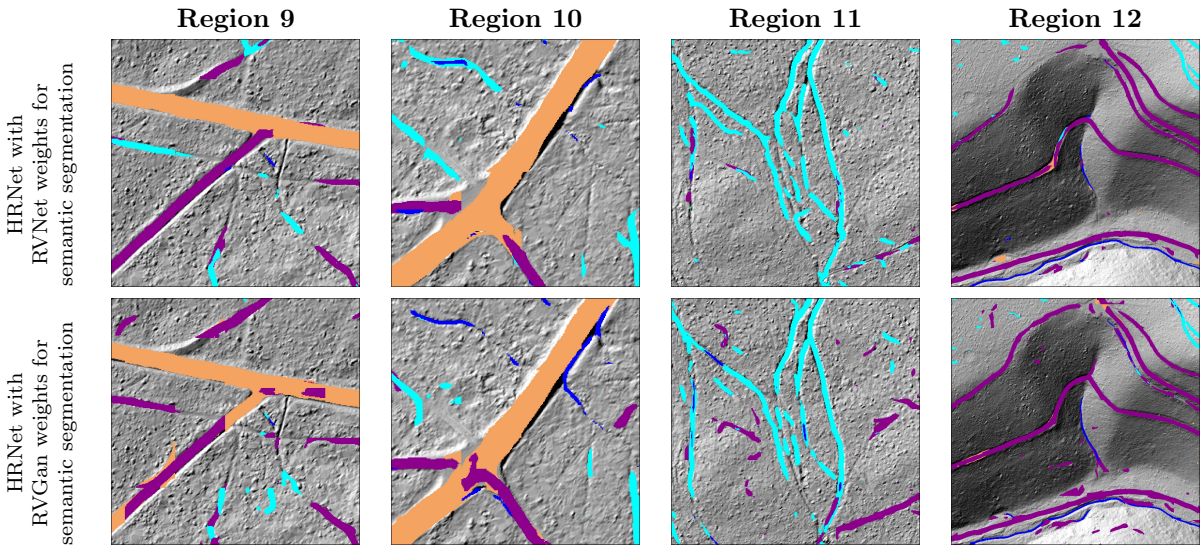


Table A.17.: Predictions for 4 regions by the trained models in the linear dataset.

Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover

(Eine vollständige Liste der Wiss. Arb. ist beim Geodätischen Institut, Nienburger Str. 1, 30167 Hannover erhältlich.)

Nr. 352	ALI, Bashar:	Optimierte Verteilung von Standorten der Schulen unter dem Einfluss des demografischen Wandels am Beispiel Grundschulen (Diss. 2019)
Nr. 353	ZHAO, Xin:	Terrestrial Laser Scanning Data Analysis for Deformation Monitoring (Diss. 2019)
Nr. 354	HAGHIGHI, Mahmud Haghshenas:	Local and Large Scale InSAR Measurement of Ground Surface Deformation (Diss. 2019)
Nr. 355	BUREICK, Johannes:	Robuste Approximation von Laserscan-Profilen mit B-Spline-Kurven (Diss. 2020)
Nr. 356	BLOTT, Gregor:	Multi-View Person Re-Identification (Diss. 2020)
Nr. 357	MAAS, Alina Elisabeth:	Klassifikation multitemporaler Fernerkundungsdaten unter Verwendung fehlerbehafteter topographischer Daten (Diss. 2020)
Nr. 358	NGUYEN, Uyen:	3D Pedestrian Tracking Using Neighbourhood Constraints (Diss. 2020)
Nr. 359	KIELER, Birgit:	Schema-Matching in räumlichen Datensätzen durch Zuordnung von Objektinstanzen (Diss. 2020)
Nr. 360	PAUL, Andreas:	Domänenadaption zur Klassifikation von Luftbildern (Diss. 2020)
Nr. 361	UNGER, Jakob:	Integrated Estimation of UAV Image Orientation with a Generalised Building Model (Diss. 2020)
Nr. 362	COENEN, Max:	Probabilistic Pose Estimation and 3D Reconstruction of Vehicles from Stereo Images (Diss. 2020)
Nr. 363	GARCIA FERNANDEZ, Nicolas:	Simulation Framework for Collaborative Navigation: Development - Analysis – Optimization (Diss. 2020)
Nr. 364	VOGEL, Sören:	Kalman Filtering with State Constraints Applied to Multi-sensor Systems and Georeferencing (Diss. 2020)
Nr. 365	BOSTELMANN, Jonas:	Systematische Bündelausgleichung großer photogrammetrischer Blöcke einer Zeilenkamera am Beispiel der HRSC-Daten (Diss. 2020)
Nr. 366	OMIDALIZARANDI, Mohammad:	Robust Deformation Monitoring of Bridge Structures Using MEMS Accelero-meters and Image-Assisted Total Stations (Diss. 2020)
Nr. 367	ALKHATIB, Hamza:	Fortgeschrittene Methoden und Algorithmen für die computergestützte geodätische Datenanalyse (Habil. 2020)
Nr. 368	DARUGNA, Francesco:	Improving Smartphone-Based GNSS Positioning Using State Space Augmentation Techniques (Diss. 2021)
Nr. 369	CHEN, Lin:	Deep learning for feature based image matching (Diss. 2021)
Nr. 370	DBOUK, Hani:	Alternative Integrity Measures Based on Interval Analysis and Set Theory (Diss. 2021)
Nr. 371	CHENG, Hao:	Deep Learning of User Behavior in Shared Spaces (Diss. 2021)
Nr. 372	MUNDT Reinhard Walter:	Schätzung von Boden- und Gebäudewertanteilen aus Kaufpreisen bebauter Grundstücke (Diss. 2021)
Nr. 373	WANG, Xin:	Robust and Fast Global Image Orientation (Diss. 2021)
Nr. 374	REN, Le:	GPS-based Precise Absolute and Relative Kinematic Orbit Determination of Swarm Satellites under Challenging Ionospheric Conditions (Diss. 2021)
Nr. 375	XU, Wei:	Automatic Calibration of Finite Element Analysis Based on Geometric Boundary Models from Terrestrial Laser Scanning (Diss. 2021)
Nr. 376	FENG, Yu:	Extraction of Flood and Precipitation Observations from opportunistic Volunteered Geographic Information (Diss. 2021)
Nr. 377	YANG, Chun:	A hierarchical deep learning framework for the verification of geospatial databases (Diss. 2021)
Nr. 378	MEHLTRETTER, Max:	Uncertainty Estimation for Dense Stereo Matching using Bayesian Deep Learning (Diss. 2021)
Nr. 379	KAZIMI, Bashir:	Self Supervised Learning for Detection of Archaeological Monuments in LiDAR Data (Diss. 2021)

Die Arbeiten werden im Rahmen des wissenschaftlichen Schriftenaustausches verteilt und sind nicht im Buchhandel erhältlich. Der Erwerb ist zu einem Stückpreis von € 25,00 bei den herausgebenden Instituten möglich.