DGK Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 883

Chun Yang

A hierarchical deep learning framework towards the verification of geospatial databases

München 2022

Bayerische Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5295-6

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 377, Hannover 2022

DGK Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 883

A hierarchical deep learning framework towards the verification of geospatial databases

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. Chun Yang

München 2022

Bayerische Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5295-6

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 377, Hannover 2022

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. Steffen Schön

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

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

Prof. Dr.-Ing. habil. Michael Ying Yang (Twente, The Netherlands)

Tag der mündlichen Prüfung: 22.12.2021

© 2022 Bayerische Akademie der Wissenschaften, München

Alle Rechte vorbehalten. Ohne Genehmigung der Herausgeber ist es auch nicht gestattet,
die Veröffentlichung oder Teile daraus auf photomechanischem Wege (Photokopie, Mikrokopie) zu vervielfältigen

ISSN 0065-5325 ISBN 978-3-7696-5295-6

Abstract

Land use describes the socio-economic function of an area of the earth surface (e.g. *settlement, agricultural, water bodies* etc.). Normally, a land use object consists of many different *land cover* elements. Land cover is related to the physical material of the earth surface, e.g. *building, asphalt, grass, tree*. Commonly, the information of land use is collected and represented in the form of polygons in geospatial databases, often acquired and maintained by national mapping agencies. For applications such as urban management and planning, up-to-date land use information is of high importance. However, the rapid development, e.g. of cities, makes the geospatial databases outdated quickly. Therefore, there is a demand for the automatic verification of geospatial databases. This can be achieved by comparing the database content to current remote sensing data to check whether the information contained in the database is still correct. Errors thus identified can then be corrected. This thesis deals with the automation of the database verification process. Due to the fact that the object catalogue of geospatial databases is frequently constructed in a hierarchical manner, this thesis tries to predict land use in multiple semantic levels *hierarchically* and *simultaneously*.

To achieve that goal, a hierarchical deep learning framework is proposed which applies a two-step strategy. First, given high-resolution aerial images, the land cover information is determined. To achieve this, an encoder-decoder based convolutional neural network (CNN) is proposed. Second, the pixel-wise land cover information and the aerial images serve as input for another CNN to classify land use. To guarantee consistency of the classification results with the class hierarchy, two strategies are proposed. The first one is called *joint optimization* (JO). Using this strategy, predictions are made by selecting the hierarchical tuple over all levels which has the maximum joint class score, providing *consistent* results across the different levels. The second strategy is to use the predictions at the finest level to control the predictions at the coarser ones, which is called *fine-to-coarse* (F2C).

To evaluate the performance and to investigate the strengths and limitations of the proposed methods, extensive experiments are conducted using five datasets: Hameln, Schleswig, Mecklenburg-Vorpommern (MV), Vaihingen and Potsdam. In terms of land cover classification, the proposed CNN achieves overall accuracies between 85% and 90% when dealing with class structures of 6 to 10 land cover classes. Using the Vaihingen and Potsdam datasets, the proposed CNN achieves results on par with state-of-art methods, but only requires 1% of unknown parameters compared to these methods. In terms of hierarchical land use classification, both the JO and F2C strategies achieve very good results. At the coarsest level, in which only four classes are differentiated, the overall accuracy can reach about 95%. As the semantic level increases, i.e. with an increasing number of classes to be discerned, the categories are more and more difficult to be correctly differentiated. In terms of overall accuracy, a drop of 10% - 15% between level I and level II (14 classes), and of about 5% between level II and level III (21 classes) is observed. Furthermore, it is found that object size has an impact on the classification. For large objects, the classification accuracy is higher than the one for small polygons

Keywords: land cover classification, land use classification, CNN, geospatial database, joint optimization

Kurzfassung

Die Landnutzung beschreibt die sozioökonomische Funktion eines Gebietes der Erdoberfläche (z. B. *Siedlung*, *Landwirtschaft*, *Gewässer* usw.). Normalerweise besteht ein Landnutzungsobjekt aus vielen verschiedenen Landbedeckungselementen. Die Landbedeckung bezieht sich auf das physikalische Material der Erdoberfläche, z.B. *Gebäude*, *Asphalt*, *Gras*, *Baum*. Im Allgemeinen werden die Informationen über die Landnutzung gesammelt und in Form von Polygonen in Geodatenbanken dargestellt, die häufig von staatlichen Vermessungssbehörden erfasst und gepflegt werden. Für Anwendungen wie Stadtmanagement und Stadtplanung ist aktuelle Landnutzungsinformation von großer Bedeutung. Die rasante Entwicklung, z.B. von Städten, führt dazu, dass Geodatenbanken schnell veraltet sind. Daher besteht ein Bedarf an der Verifizierung der Geodatenbanken. Dieses Ziel kann erreicht werden, indem der Datenbankinhalt mit aktuellen Fernerkundungsdaten verglichen wird, um zu überprüfen, ob die in der Datenbank enthaltenen Informationen noch korrekt sind. So identifizierte Fehler können dann korrigiert werden. Diese Arbeit beschäftigt sich mit der Automatisierung des Datenbankverifikationsprozesses. Da der Objektartenkatalog von Geodatenbanken häufig hierarchisch aufgebaut ist, versucht diese Arbeit, die Landnutzung in mehreren semantischen Ebenen des hierarchisch strukturietn Objektartenkatalog gleichzeitig zu prädizieren.

Um dieses Ziel zu erreichen, wird ein hierarchisches Deep-Learning-Framework vorgeschlagen, das eine zweistufige Strategie anwendet. Zunächst wird aus hochaufgelösten Luftbildern die Landbedeckungsinformation ermittelt. Um dies zu erreichen, wird ein auf einer Encoder-Decoder Struktur basierendes Convolution Neural Network (CNN) vorgeschlagen. Anschließend dient die pixelweisen Landbedeckungsinformation zusammen mit den Luftbildern als Eingabe für ein weiteres CNN, um die Landnutzung zu klassifizieren. Um die Konsistenz der Ergebnissen mit der Klassenhierarchie zu gewährleisten, werden zwei Strategien vorgeschlagen. Die erste wird als gemeinsame Optimierung (JO) bezeichnet, bei der jenes Tupel aus Klassenlabels über alle Ebenen ausgewählt wird, das einen maximalen gemeinsamen Klassenscore aufweist. Die zweite Methode besteht darin, die Prädoktion auf der feinsten Ebene zu verwenden, um jene auf den gröberen Ebenen zu steuern. Die Strategie wird als fein-zu-grob Methode (F2C) bezeichnet.

Um die Leistungsfähigkeit zu bewerten und die Stärken und Grenzen der vorgeschlagenen Methoden zu untersuchen, werden umfangreiche Experimente mit fünf Datensätzen durchgeführt: Hameln, Schleswig, Mecklenburg-Vorpommern (MV), Vaihingen und Potsdam. In Bezug auf die Landbedeckungsklassifikation erreicht das vorgeschlagene CNN Gesamtgenauigkeiten zwischen 85% und 90%, wenn 6 bis 10 Landbedeckungsklassen unterschieden werden. Mit den Datensätzen von Vaihingen und Potsdam erreicht das vorgeschlagene CNN Ergebnisse, die mit den modernsten Methoden vergleichbar sind, es erfordert jedoch nur ca. 1% der unbekannten Parameter dieser Methoden. Hinsichtlich der hierarchischen Landnutzungsklassifikation erzielen sowohl die JO als auch die F2C Strategie sehr gute Ergebnisse. Auf der gröbsten Stufe, in der nur vier Klassen unterschieden werden, beträgt die Gesamtgenauigkeit etwa 95%. Mit zunehmender semantischer Auflösung, d.h. mit zunehmender Anzahl der zu unterscheidenden Klassen, sind die Kategorien schwerer zu unterscheiden. In Bezug auf die Gesamtgenauigkeit wurde ein Rückgang von 10 - 15% zwischen Level I und Level II (14 Klassen) und von etwa 5% zwischen Level II und Level III (21 Klassen) beobachtet. Weiterhin zeigt

sich, dass die Objektgröße Auswirkungen auf die Klassifizierung hat. Große Objekte lassen sich leichter richtig klassifizieren als kleine Polygone.

Schlagworte: Landbedeckungsklassifikation, Landnutzungsklassifikation, CNN, Geodatenbank, gemeinsame Optimierung

Nomenclature

Abbreviations

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
ALKIS	Amtliches Liegenschaftskatasterinformationssystem
ANN	Artificial Neural Network
BLR	Base Learning Rate
BG	Background
BN	Batch Normalization
C2F	Coarse-to-Fine
CB	Convolution Block
CNN	Convolutional Neural Network
CIR	Color Infra-Red
CRF	Conditional Random Fields
DTM	Digital Terrain Model
DSM	Digital Surface Model
DOP	Digital Orthophoto
F2C	Fine-to-Coarse
FC	Fully-connected
GAP	Global Average Pooling
GIS	Geo-Information System
GLCM	Grey Level Co-Occurrence Matrix
GSD	Ground Sampling Distance
HED	Holistically-Nested Edge Detection
HEX	Hierarchy and Exclusion
HOG	Histogram of Oriented Gradients
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IoU	Intersection over Union

ISPRS	International Society of Photogrammetry and Remote Sensing
IR	Infrared
JO	Joint Optimization
LaiV-MV	Landesamt für innere Verwaltung Mecklenburg-Vorpommern
LC	Land Cover Classification
LI, LII, LIII	Semantic Level I, II, III
LGLN	Landesamt für Geoinformation und Landesvermessung Niedersachsen
LSTM	Long Short Term Memory
LU	Land Use Classification
LVermGeo SH	Landesamt für Vermessung und Geoinformation Schleswig Holstein
nDSM	normalised Digital Sufrace Model
MBS	Mini Batch Size
MLP	Multi Layer Peceptron
MS	Multi-scale
MT	Multi Task
MV	Mecklenburg-Vorpommern
NIR	Near Infarred
ReLU	Rectified Linear Unit
RID	Image consisting of bands Red, Infrared, nDSM
RGB	Red-Green-Blue
RF	Random Forest
RNN	Recurrent Neural Network
ROI	Region Of Interest
SGD	Stochastic mini-batch gradient descent
SVM	Support Vector Machine
UCM	UC Merced Land Use Dataset

List of Symbols

General

$c, d, i, j, k, l, m, n, r, \tau$	indexes
A	area of a patch of 256 x 256 pixels and is GSD-dependent
X, Ω	mini batch of images, and mini batch size
X	2D image
$\mathbf{w}, b, \boldsymbol{\theta}$	weights, bias and all unknown parameters
F_o, F_o'	convolution kernel
Y, Y'	feature maps
f	nonlinear mapping
y, z	output layer of a neural network, output of a linear mapping
$\boldsymbol{h}$	hidden layer
$\boldsymbol{x}, \boldsymbol{x}_n$	feature vector
s, q	size of convolution kernel, number of kernels
N_T	number of training samples
t_i^k	the reference of the *i*-th sample of the *k*-th class
$\mathbb{C}, K$	set of classes, number of classes
$L, \tilde{L}$	loss
$P(C^k \mid \boldsymbol{x_n})$	the class probability of the n^{th} sample of the k^{th} class
H, W	height and width of a 2D image
η, λ, δ	learning rate, weight decay factor and learning decay factor
N_{iter}	the number of iteration in an epoch
C, O	number of input channel and output channel
S	stride for convolution or max-pooling
E_{decay}, E_{total}	decay epoch of learning rate and total training epoch
$\mu_\Omega, \sigma_\Omega^2$	mean and deviation of values of a neuron in a mini batch Ω

Land Cover Classification

M_{LC}	number of land cover classes to be discerned
$\mathbb{C}_{LC}$	land cover class set
$C_{LC}^{1}, \dots, C_{LC}^{M_{LC}}$	elements of $\mathbb{C}_{LC}$
$\boldsymbol{z_{LC}^{i}}$	raw land cover class scores for *i*-th sample before normalization
$z_{LC}^{i,1}, \dots, z_{LC}^{i,M_{LC}}$	elements of $\boldsymbol{z_{LC}^{i}}$
$P_i(C_{LC}^{c}\|X)$	class probability of *i*-th sample belonging to class C_{LC}^{c}
$\boldsymbol{f_{idx}^{j}, v_{idx}^{j}, g^{d}}$	feature maps
$\omega_{j,idx}$, $b_{j,idx}$	parameters of a convolution kernel and bias
L_{LC}	extended focal loss in land cover classification
γ	hyper-parameter in the loss function L_{LC}

Land Use Classification

$N_{input}, N_{shape}, N_{DOP}, N_{3D}, N_{LC}$	number of bands
N_{min}	threshold for determining random sampling during tiling approach
α	scale factor for scaling polygon
$\mathbb{C}_{LU}$	one-level land use class set
$C_{LU^1}, \dots, C_{LU^M}$	elements of $\mathbb{C}_{LU}$
$\boldsymbol{z_{LU}}$	raw one-level land use class scores before normalization
$z_{LU^1}, \dots , z_{LU^M}$	elements of $\boldsymbol{z_{LU}}$
$P(C_{LU^c}\|\mathrm{X})$	class probability of class C_{LU^c}
L_{LU}	extended focal loss in one-level land use classification
ϵ	hyper-parameter in the loss function L_{LU}
B	number of semantic levels
M_l	number of land use classes at semantic level *l*
$C_{LU,c}^{l}$	land use class set at semantic level *l*

$C_{LU,1}^{l}, \dots, C_{LU,M_l}^{l}$	elements of $C_{LU,c}^{l}$
$\boldsymbol{z_{LU}^{l}}$	raw land use class scores at semantic level l
$z_{LU,1}^{l}, \dots, z_{LU,M_l}^{l}$	elements of $\boldsymbol{z_{LU}^{l}}$
$\boldsymbol{z_{LU}^{mid,l}}, \boldsymbol{z_{LU}^{out,l}}$	immediate and final raw land use class scores at semantic level l
$P(C_{LU,c}^{l}\|\mathrm{X})$	class probability of class $C_{LU,c}^{l}$ at level l
θ, ρ	learnable parameters for exchanging information between semantic levels
M_B	number of valid tuples
T_i	the i-th tuple
$P_{joint}^{i,k}(T_i, X_k)$	joint class scores of the i-th tuple T_i
L_{LU}^{JO}	loss for optimizing the JO approach
$L_{LU}^{JO,P1}, L_{LU}^{JO,P2}$	two parts of L_{LU}^{JO}

Evaluation

M	number of classes to be discerned
$\mathbb{C}$	class set
$C^1, \dots, C^M$	elements of $\mathbb{C}$
c_{ij}	element of confusion matrix
OA	overall accuracy
$Precision_m, Recall_m, F1_m$	precision, recall and F1 score of class m
$mF1$	mean F1 scores over all classes
ρ_{im}	degree of class imbalance

Contents

1. Introduction

1.1. Motivation

Land use describes the socio-economic function of an area of the earth surface (e.g. *settlement, agricultural, water bodies* etc.). Normally, a land use object consists of many different *land cover* elements. Land cover is related to the physical material of the earth surface. There are man-made land cover types, e.g. *building, asphalt* and natural ones, e.g. *grass, tree*. Commonly, the information of land use is collected and represented in the form of polygons in geospatial databases, often acquired and maintained by national mapping agencies (NMAs). For applications such as urban management and planning, up-to-date land use information is of high importance. However, the rapid development, e.g. of cities, makes the geospatial databases outdated quickly. Therefore, updating the geospatial databases is an important task.

The first step of this process is the verification of the geospatial databases. It can be achieved by comparing the database content to current remote sensing data to check whether the information contained in the database is still correct. Errors thus identified can then be corrected. This thesis deals with the automation of the database verification process.

To obtain a more efficient update process which guarantees the up-to-dateness of the databases, the NMAs strive for the implementation of an automatic verification process, for which up-to-date high resolution aerial or satellite images, along with derived products such as digital Orthophotos (DOP), digital surface models (DSM) and digital terrain models (DTM) are used. An automatic verification process can be realized based on an automatic classification of land use by using sensor data and comparing the results to the database contents, e.g. (Helmholz et al., 2014). For the purpose of obtaining highly reliable verification results, the quality of the classification of land use is of crucial importance. For practical relevance, many requirements must be fulfilled. First, the classification should provide a fine class structure which shall be identical to the class structure of the databases to be verified. Second, the developed classification method should be directly transferable to other areas so that it can be applied to new areas immediately once a classifier is trained. This means that the application of the method is not limited to a specific area or a specific characteristic (e.g. urban or rural). Last, the classification method should deliver accurate and reliable results. If these requirements are fulfilled, the classification results can be considered to be reliable and applicable for an efficient verification of geospatial databases.

Commonly, the workflow of an automatic verification process consists of three steps: collecting data of a test area, classifying that test area and comparing results to the corresponding database contents. This thesis mainly focuses on the classification step.

Regarding to the classification methods, supervised methods are increasingly applied, because they are more easily transferable to other scenes than model-based techniques. The classifier is trained on training data in a supervised way. The standard workflow starts with extracting features for image primitives which are the classification units for a specific task (e.g. image pixels for pixel-level classification); afterwards, these features along with the reference class labels of the image primitives

are given to a classifier to determine its parameters in a supervised way. Finally, the class labels for new image primitives which were not seen during training are predicted by the classifier. Considering land use classification, existing methods differ by the primitives to be classified, the data sources, the features and the classifiers. Based on the fact that a land use object could contain many land cover elements, many approaches rely on a strategy consisting of two steps (Hermosilla et al., 2012): after determining land cover in the first stage, the results are used to support the land use classification in the second stage. In the context of classification, features defined by a user are referred to as *hand-crafted features*. Possible classifiers that could be trained on the basis of such features are Random Forests (RF; e.g. Albert et al., 2014) or contextual model like Conditional Random Fields (CRF; e.g. Albert et al., 2017).

Since the success of AlexNet (Krizhevsky et al., 2012), convolutional neural networks (CNN) have been shown to outperform such classifiers by a large margin, replacing *hand-crafted features* by a representation learned from training data. CNN consist of building blocks that commonly combine a convolution layer, a non-linearity layer and a pooling layer reducing the spatial resolution of the signal; in the end, a softmax function is applied to obtain probabilistic class scores for classification. Convolution via a filter kernel captures context information; applying a non-linearity (e.g. rectified linear unit, ReLU; Nair and Hinton, 2010) results in a better representation, and pooling is a commonly applied operation to make the learned representation translation-invariant and to increase the size of the receptive field of the captured context. These facts lead to *learned features*, which are more discriminative than hand-crafted features. For instance, AlexNet achieved a top-5 error of 15.3% in the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC 2012), about 10.8% lower than the best methods not based on CNN at that time. Later, networks with more than 100 convolution layers were proposed by (He et al., 2016), bringing the top-5 error down to 3.6%.

Although many supervised methods relying on hand-crafted features have been proposed for land use classification, they suffer from a relatively low accuracy. For instance, the CRF-based method proposed by Albert et al. (2017) achieves about 78% overall accuracy for classifying ten land use types. Due to the powerful classification ability of CNN in vision tasks, it is very interesting to know how it can be used for land use classification to improve the classification performance, and whether more accurate and reliable results can be obtained for the verification of geospatial databases. That question motivates this thesis, which has the goal to develop methods for land use classification based on CNN.

As mentioned earlier, land use objects may contain different land cover elements. For instance, *settlement* is mainly made up of *building, asphalt, grass;* on the other hand, it is not common to have a *forest* area with many man-made objects. Therefore, the land cover information will play an important role for classifying land use objects and thus, a two-step strategy for land use classification is also employed in this work: first, a CNN is used for land cover classification to obtain pixel-level land cover information; second, this information along with the image data (and their derived data such as a DSM) serve as input for the subsequent land use classification.

Another problem of most existing approaches for land use classification is that they solely differentiate a small number of classes, without considering the fact that the object catalogues of land use databases may be much more detailed. For instance, in the land use layer of German Authoritative Real Estate Cadastre Information System (ALKIS; AdV, 2008), about 190 categories are differentiated. Clearly, the catalogue contains object types that cannot be expected to be differentiated based on remote sensing data. However, the usefulness of an automatic verification process grows with an increasing

number of classes. It is an important fact that many topographic databases contain land use information at different semantic levels of abstraction. At the coarsest level, only a few broad classes such as *settlement*, *traffic* or *vegetation* are differentiated. These classes are hierarchically refined, and the full number of different categories is only differentiated at the finest level of the class structure. From the point of view of the application, this hierarchical classification is more practicable for the automatic verification process, so that different levels of the details can be obtained. This fact motivates this thesis, which has the goal to develop methods to classify land use objects to classes at multiple semantic levels, while at the same time keeping the consistency of these classes with the hierarchy of the object catalogue. Performing land use classification in a hierarchical fashion allows to investigate which level of detail of the class structure by the classification can be achieved by the classification.

1.2. Goal and Contributions

The general goal of this work is to develop methods based on CNN for land use classification by using the up-to-date remote sensing data. Aerial images and derived products such as DSM and DTM serve as input for the developed methods. As the land use objects consist of many different land cover elements, the land cover information is classified first to support the land use classification. Therefore, two goals are desired: first, a pixel-wise classification of land cover based on CNN with high quality, and second, the prediction of land use for objects stored in a geospatial database.

In the context of land cover classification, multi-spectral orthophotos and height models are available. To fuse them in one model, a CNN structure similarly to (Audebert et al., 2018) is adopted. In addition, there are two problems to be tackled. First, the class distribution of training data is not balanced. Thus, strategies for addressing this problem in the training process are required. Second, it is known that the pooling operations lead to a loss of spatial information, possibly resulting in a poor geometrical quality of object boundaries (Sherrah, 2016). Therefore, this problem is also addressed in this work. With respect to these two problems, the first contribution of this work is the development of a new CNN-based method for the pixel-wise land cover classification. In this regard there are two main innovations:

1) An extension of the focal loss for binary classification (Lin et al., 2017) to multiclass classification is presented to address the problem of imbalanced class distribution in training data. Many authors use constant class weights to mitigate that problem in a multiclass classification problem, but very few apply adaptive class weights.
2) A new learnable skip-connections is proposed to fuse the low-level features in early layers of the CNN with the high-level features in the corresponding late layers to compensate for the loss of spatial information caused by pooling operations. Compared to the pure elementwise addition of feature maps, the method is able to learn to combine features. To the best knowledge of the author, this is the first work to learn skip-connections.

In the context of land use classification, the most important question is how to use a CNN to classify the land use objects which vary largely in terms of their geometric extent. For instance, a *road* object is thin and long, whereas *settlement* objects may cover very large or quite small areas. However, CNN require regular-sized input. Thus, the first problem to be tackled is how to convert these objects to input patches of a regular size. Subsequently, a newly proposed CNN is used to classify those patches, and the results are again combined to obtain the predictions for land use objects. In this thesis, the hierarchical classification of land use objects is of the highest interest, in particular requiring the results of the classification to be consistent with the hierarchical class structure. In the context of land use classification, there are four main innovations:

1) A workflow of classifying land use objects by CNN is proposed. Regarding to the classification methods, a new CNN is proposed. This CNN is not only limited to the prediction of one land use label, but is also extended for hierarchical land use classification. To the best knowledge of the author, this is the first work to use CNN to hierarchically classify land use objects.
2) To incorporate the hierarchical class structure into the training of the proposed CNN, a novel loss function is proposed to optimize the joint class scores over all multiple semantic levels. In this loss function, penalty terms similar in focal loss are introduced to mitigate the problem of imbalanced class distribution. To the knowledge of the author, this is the first work to consider the class hierarchy in loss functions in a CNN-based method.
3) Two strategies are proposed to obtain predictions guaranteeing consistency with the class hierarchy of the object catalogue. One is to use the predictions at the finest semantic level of the CNN to control the predictions at coarser level by applying a so-called *fine-to-coarse* (F2C) post-processing strategy; the other one is to select the tuple having the maximum joint class score of class labels. Neither of these strategies has been exploited by other works so far.
4) The polygons are represented by binary masks describing their shapes. Relying on these masks, two approaches are proposed to convert the polygons to regular-sized patches. In the *tiling* approach, large polygons are split into patches of identical size (256 x 256 pixels in this thesis), whereas small polygons will correspond to exactly one patch. It has to be noted that in this approach the resolution of the image data is kept unchanged. Using the *scaling* approach, all polygons are scaled to the predefined window size (256 x 256 pixels). For small polygons, multiple scaling factors are used, so that these polygons will be presented to CNN with different context windows. To the best knowledge of the author, this is the first work to convert irregular shaped polygons to regular patches in this way.

The proposed methods for land cover and land use classification will be investigated by extensive experiments using real cartographic datasets. All experiments shall highlight the advantages of the developed methods on the one hand, and on the other hand, show their limitations.

1.3. Structure of this Thesis

This thesis is organized as follows: in Section 2, the theoretical background about neural networks and CNN is introduced, covering the essential building blocks and the training procedure. In Section 3, related work is introduced. Section 4 presents the developed methods for land cover and land use classification. Section 5 describes the datasets and the experimental setups, while the corresponding experiments are presented in Section 6. Finally, Section 7 presents conclusions and an outlook on future work.

2. Fundamentals

In this chapter, the methods on which this thesis mainly relies are introduced. In Section 2.1, the general introduction of classification at image level and at pixel level is presented. Subsequently, the basics of artificial neural networks (ANN) are described in Section 2.2, and in Section 2.3 the fundamentals of CNN are presented.

2.1. Classification

Given an input image, the goal of classification is to predict a discrete class label for the entire image or for each pixel. Classification at image level is referred to as *image classification*, whereas the one at pixel level is referred to as *semantic segmentation* in the computer vision literatures. The input unit is referred to as *primitive*, which can be an image or a pixel, depending on the application. In a classification task, each primitive is represented by a feature vector $\boldsymbol{x}$. The feature vector can be derived from the observed sensor data of the corresponding primitive or in a local neighbourhood. For instance, a feature vector can describe the spectral signature, texture and structure of the objects in the image. In a classification task, feature vectors are required to be determined for all primitives prior to classification. In the context of neural networks, the definition of feature vectors can be learned from input data.

For N input primitives, the membership of a primitive $n \in \{1, \dots, N\}$ to one of the K classes is described by its label $\tilde{C}_n \in \mathbb{C} = \{C^1, \dots, C^K\}$, in which $\mathbb{C}$ is the discrete set of all classes and C^k with $k \in \{1, \dots, K\}$ is a particular class. In classification, the feature vector of the n^{th} primitive, $\boldsymbol{x_n}$, is assigned a class label $\tilde{C}_n$ (Bishop, 2006). In probabilistic methods, probabilities are used for predicting the class labels. The maximum a posterior (MAP) criterion is used to select the class label for which the probability $P(\tilde{C}_n|\boldsymbol{x_n})$ becomes a maximum:

$$\tilde{C}_n = \underset{k}{argmax}\ (P(\tilde{C}_n = C^k|\boldsymbol{x_n})) \tag{2.1}$$

To determine the parameters which are used for the prediction of the class labels, the classifier is trained with the help of the training data. The training data consists of N_T primitives, which are also called *training samples*, and the corresponding training labels for each primitive $\bar{C}_n \in \mathbb{C}$ with $n \in N_T$. It has to be noted that the training data shall be representative, which means that its distribution shall be identical as the data to be classified, because the training is based on the training data to learn the dependencies between the observed data and the labels.

2.2. Artificial Neural Network

An artificial neural network (ANN) is based on a collection of connected units or nodes, which are also called *neurons*. Each connection can transmit a signal to other neurons. A neuron receives a signal then processes it and can emit a signal to neurons connected to it. The simplest ANN is a *perceptron*, which can be used for binary classification. It consists of one neuron only. The input corresponds to the feature vector of one primitive. A more complex structure for classification is offered by *multilayer perceptrons* (MLP), which consist of an input layer, an output layer and at least one hidden layer. In this section, the perceptron and MLP are introduced in Sections 2.2.1 and 2.2.2, respectively. Section 2.2.3 describes the training procedure.

2.2.1. Perceptron

A perceptron consists of one neuron only (cf. Fig. 2.1). Given an input feature vector $\boldsymbol{x}$, the neuron determines a linear combination according to eq. 2.2:

$$z = \mathbf{w} \cdot \boldsymbol{x} + b \tag{2.2}$$

where $\mathbf{w}$ is a vector of weights, $\mathbf{w} \cdot \boldsymbol{x}$ is a dot product of both vectors and b is a bias term. The values of $\mathbf{w}$ and b are learned from the training data. The output z forms the input of a nonlinear function f to obtain the final output y (eq. 2.3). This nonlinear mapping is also called *activation* function.

$$y = f(z) \tag{2.3}$$

One commonly used nonlinear function is the step function, which returns $y = 1$ if $z > 0$, and $y = 0$ otherwise. However, the gradient of this function is 0 except for $z = 0$ when it is undefined, which makes training by gradient descent impossible. An alternative is the *sigmoid* function, defined in eq. 2.4:

$$f(z) = sigmoid(z) = \frac{1}{1+exp^{-z}} \tag{2.4}$$

which can be differentiated everywhere. For a set of two classes $\mathbb{C} = \{C^1, C^2\}$, the output of eq. 2.4 can be interpreted as the posterior probability $\boldsymbol{x}$ belonging to class C^1, denoted as $P(C^1|\boldsymbol{x})$. The probability for class C^2 is $P(C^2|\boldsymbol{x}) = 1 - P(C^1|\boldsymbol{x})$. Thus, the perceptron can be used for binary classification. However, one problem of the perceptron is that it requires the input feature vectors of the two classes to be linearly separable. To tackle that problem, MLP can be a solution.

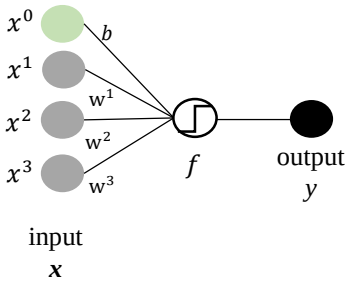


Figure 2.1: An example of a perceptron for binary classification. The input feature vector $\boldsymbol{x}$ has a length of 3. $x^0 = 1$ corresponds to the bias term.

2.2.2. Multilayer Percptrons

A hidden MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Each layer consists of neurons in which the operations presented in Section 2.2.1 are performed. In recent developments of deep learning, the *rectifier linear unit* (ReLU) is more frequently used as activation function instead of the one in eq. 2.4:

$$f(z) = ReLU(z) = \max(0, z) \tag{2.5}$$

The advantage of this function is its constant gradient for $z > 0$, whereas in sigmoid the magnitude of gradient will be very close to 0 for very large z, which will slow down the training process. MLP can not only be used for binary classification, but also for multiclass classification. An example of a MLP consisting of three layers for a three-class classification is shown in Fig. 2.2.

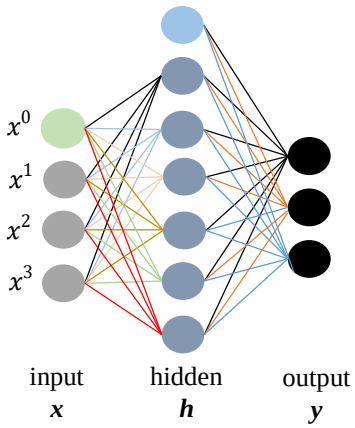


Figure 2.2: An example of MLP with three layers for a three-class classification. For reasons of simplicity, the nonlinear function is not shown. In the input and hidden layers, the uppermost neuron has a value of 1, corresponding to the bias term.

In Fig. 2.2, the outputs of the neurons in the hidden layer are obtained by a linear combination of the input and a subsequent nonlinear activation:

$$\boldsymbol{h} = f(\mathbf{W^h} \cdot \boldsymbol{x} + \boldsymbol{b^h}) \tag{2.6}$$

where $\boldsymbol{h}$ represents the neurons in the hidden layer, $\mathbf{W^h}$ is a weight matrix and $\boldsymbol{b^h}$ is a vector of biases. It has to be noted that each neuron in a hidden layer is connected to all neurons in the previous layer. Such layer is also called a *fully-connected* (FC) layer. In the output layer, a linear combination of the neurons of the previous hidden layer is determined:

$$\boldsymbol{y} = \mathbf{W^o} \cdot \boldsymbol{h} + \boldsymbol{b^o} \tag{2.7}$$

where $\mathbf{W^o}$ is a weight matrix and $\boldsymbol{b^o}$ is a vector of biases. To obtain normalized probabilistic scores, the *softmax* function is commonly applied:

$$P(C^k|\boldsymbol{x}) = softmax(\boldsymbol{y}, C^k) = \frac{\exp(y^k)}{\sum_{j=1}^{K} \exp(y^j)} \tag{2.8}$$

where y^k is the k^{th} element of $\boldsymbol{y}$, corresponding to class $C^k \in \mathbb{C} = \{C^1, \dots, C^K\}$. These probabilistic scores are the argument of the loss function which is used for training the MLP, which will be described in Section 2.2.3.

2.2.3. Training

To determine the parameters of an MLP, a loss function is minimized. In this section, the most frequently used loss function and optimization method are described.

2.2.3.1. Loss Function

For reasons of simplicity, let $\boldsymbol{\theta}$ denote all trainable parameters in a network. For a classification problem with K classes with $\mathbb{C} = \{C^1, \dots, C^K\}$, the most frequently used loss function is the *cross entropy* function, which is defined in eq. 2.9:

$$L(\boldsymbol{\theta}) = -\frac{1}{N_T} \cdot \sum_{i=1}^{N_T} \sum_{k=1}^{K} t_i^k \cdot log(P(C^k|\boldsymbol{x}_i)) \tag{2.9}$$

In eq. 2.9, N_T is the total number of training samples and $\boldsymbol{x}_i$ is the feature vector of the i^{th} sample. t_i^k is 1 if the i^{th} sample belongs to class C^k, otherwise it is 0. $P(C^k|\boldsymbol{x}_i)$ is the probabilistic score for $\boldsymbol{x}_i$ belonging to class C^k, which is determined by eq. 2.8. Commonly, neural networks have a very large number of parameters, which can lead to overfitting to the training data. One option to avoid overfitting is to add a regularization term to the loss function. A simple regularizer is *weight decay*, giving a regularized loss of the form (Bishop, 2006):

$$\widetilde{L}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \frac{\lambda}{2}\boldsymbol{\theta}^{\boldsymbol{T}}\boldsymbol{\theta} \tag{2.10}$$

The coefficient λ is called *weight decay factor*, which is a hyperparameter required to be set before training. In this thesis, this regularizer is applied to train all network variants.

2.2.3.2. Gradient Descent Optimization

The goal of training is to find the parameter vector $\boldsymbol{\theta}$ such that $\widetilde{L}(\boldsymbol{\theta})$ becomes a minimum, which will occur at a point in parameter space where the gradient of the loss vanishes, i.e.:

$$\nabla\widetilde{L}(\boldsymbol{\theta}) = 0 \tag{2.11}$$

However, the loss function typically has a highly nonlinear dependence on the weight and bias parameters, and so there will be many points in the parameter space at which the gradient vanishes. These points are called stationary points. There will typically be multiple inequivalent stationary points and in particular multiple inequivalent minima. A minimum that corresponds to the smallest possible value of the loss for any parameter vector is called a *global minimum*. Any other minima corresponding to higher values of the loss are called *local minima*. Nonetheless, in a real application it may not be feasible to find the global minimum.

To find a good solution to the eq. 2.11, iterative numerical procedures are mostly used, for which some initial values $\boldsymbol{\theta}^{(0)}$ are chosen, usually drawn randomly from a distribution (Glorot et al, 2010), and the parameters are adapted in a succession of update steps of the form:

$$\boldsymbol{\theta}^{(\tau+1)} = \boldsymbol{\theta}^{(\tau)} + \Delta\boldsymbol{\theta}^{(\tau)} \tag{2.12}$$

where τ denotes the iteration step and $\Delta\boldsymbol{\theta}^{(\tau)}$ denotes the parameter vector update at step $\tau + 1$. As one possibility to update the parameter vector, the gradient $\nabla\widetilde{L}(\boldsymbol{\theta})$ of the loss $\widetilde{L}(\boldsymbol{\theta})$ can be evaluated at the parameter vector $\boldsymbol{\theta}^{(\tau)}$ and a vector in the direction of the negative gradient is selected, so that:

$$\boldsymbol{\theta}^{(\tau+1)} = \boldsymbol{\theta}^{(\tau)} - \eta \cdot \nabla\widetilde{L}(\boldsymbol{\theta}^{(\tau)}) \tag{2.13}$$

where $\eta > 0$ is known as the *learning rate*. Commonly, the value of η shall be small. If too large a value is used, the learning will diverge. After each such update, the gradient is re-evaluated for the new parameter vector and the process is repeated. This iterative procedure moving in the negative direction of gradients is called *gradient descent optimization*. To obtain the gradient for a specific parameter, a technique called *error backpropagation* is used. In the forward pass, the loss is computed, and in the backward pass, the gradients of the loss to each neuron and parameter are computed relying on the *chain rule* of differentiation.

It has to be noted that the loss is defined with respect to a training set, and so each step requires the entire training set to be processed in order to evaluate $\nabla\widetilde{L}(\boldsymbol{\theta}^{(\tau)})$. If the training set is very large, this will be a very time-consuming procedure. Thus, it is common only to use a randomly chosen subset of the training data (*minibatch*) in each iteration. This optimization algorithm is called *stochastic minibatch gradient descent* (SGD). In this context, eq. 2.9 is evaluated for a minibatch of Ω training samples in an iteration; the hyperparameter Ω is called minibatch size. The most important property of SGD is that computation time per update does not grow with the number of training samples, which allows fast convergence even when the number of training samples becomes very large. In the context of SGD, the term *epoch* is introduced. An epoch consists of a set of iterations so that in one epoch all training samples are used for training once. The number of iterations N_{iter} per epoch is the number of training samples divided by the minibatch size:

$$N_{iter} = \left\lfloor \frac{N_T}{\Omega} \right\rfloor \tag{2.14}$$

2.2.3.3. Step Learning Policy

Commonly, a large learning rate η is chosen to accelerate the training procedure at the beginning. However, if it is kept large during the whole training procedure, the procedure is likely to diverge. In particular, as the training iteration increases, the learning rate shall adaptively be smaller. In practice, it is necessary to decrease the learning rate over time. Starting from the *base learning rate* $\eta = \eta^{(0)}$, which is required to be set a priori, its value is reduced after each E_{decay} epochs in a total number of E_{total} training epochs by a decay factor $\delta < 1$:

$$\eta^{(\tau+1)} = \begin{cases} \eta^{(\tau)}, & if\,(\tau+1)\ mod\ (\mathrm{E}_{decay} \cdot N_{iter}) \neq 0 \\ \delta \cdot \eta^{(\tau)}, & \text{otherwise} \end{cases} \tag{2.15}$$

where *mod* represents a modulo operation. In each iteration τ, $\eta^{(\tau)}$ is used in eq. 2.13 for updating the parameters. This strategy of adapting learning rate is called *step learning policy* and used for all training processes in this thesis.

2.3. Convolution Neural Networks

In MLP, the neurons of one layer are connected to all neurons in the previous layer and each connection is associated with a weight. These weights correspond to the parameters to be learned. CNN are neural networks that employ a mathematical operation called *convolution*, which is a specialized kind of linear operation. In CNN, the neurons of one layer are connected to a small subset of neurons in the previous layer. CNN commonly consist of building blocks that combine a convolutional layer, a non-linear mapping and a pooling layer (LeCun et al., 1998). In this section, the fundamentals of CNN are described. Section 2.3.1 describes the main components in a CNN. Section 2.3.2 presents an overview of a typical CNN used for image classification, whereas Section 2.3.3 gives a brief overview of CNN used for semantic segmentation. The developments in this thesis rely on these networks. Finally, Section 2.3.4 describes the training procedure for CNN.

2.3.1. Components

The most important components of a CNN are certainly the convolution layers. Beyond that, an operation called *pooling* is frequently used in CNN, and a nonlinear mapping function is also used. Furthermore, FC layers may also be employed; they were already described in Section 2.2. As this thesis deals with images, the introduction of convolution and pooling is based on 2D images with multiple bands.

2.3.1.1. Convolution

Let X denote a 2D image with dimension $H \times W \times q$, corresponding to height, width and number of bands, respectively. An example of X with a dimension of $5 \times 5 \times 3$ is presented in the left part of Fig. 2.3.

Convolution is an operation with two real-valued arguments. One argument is referred to as the *input*, e.g. the image, and the other argument as the *kernel*, which is commonly small (e.g. 3 x 3). In the context of CNN, the output is sometimes referred to as a *feature map*. Given an input image X and O kernels $F_o \in R^{s \times s \times q}$, where s is the kernel size. The convolution operation and the output are illustrated in Fig. 2.3:

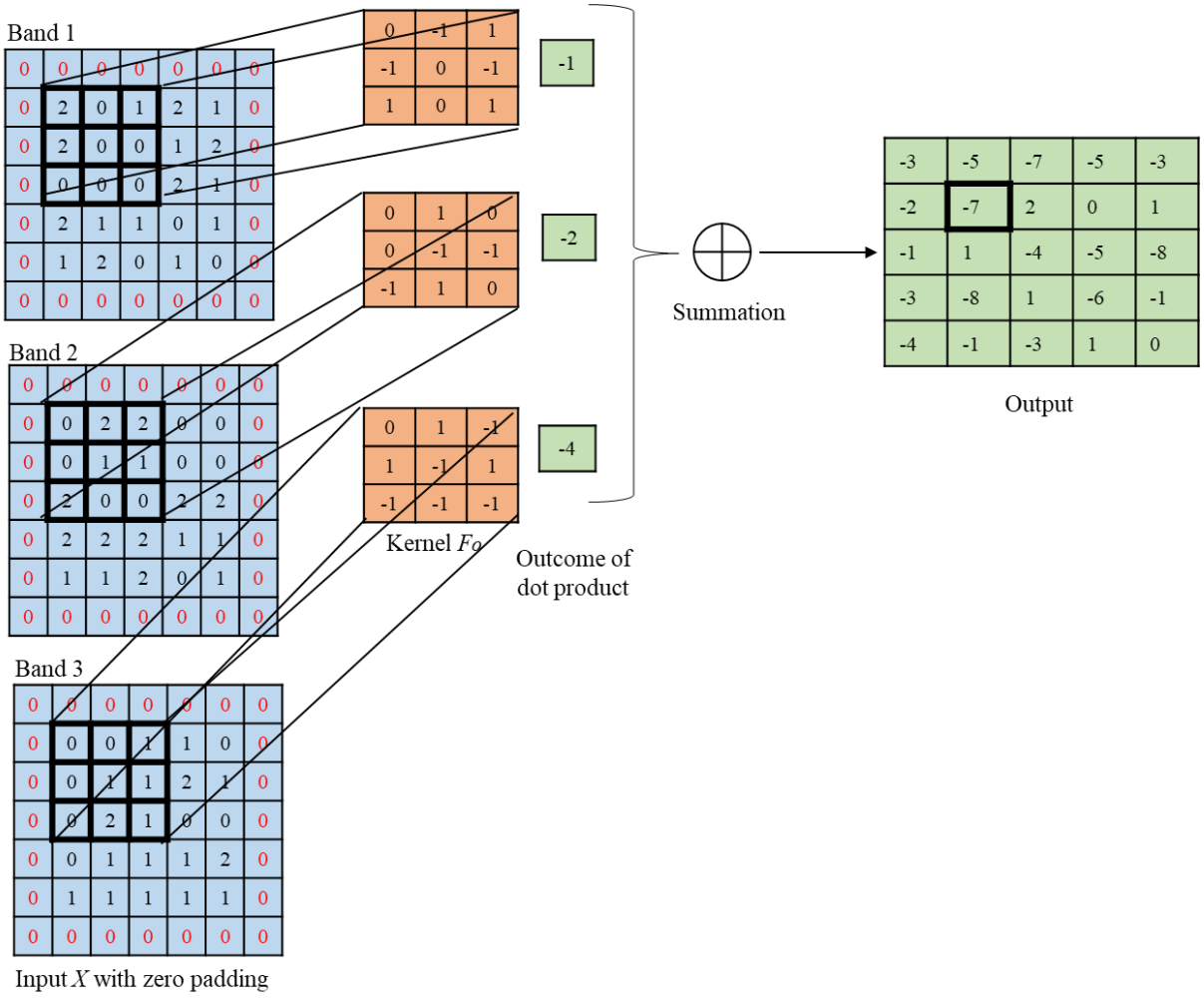


Figure 2.3: Illustration of convolution with an input $X \in R^{5\times5\times3}$ and kernel $F_o \in R^{3\times3\times3}$ with a stride $S = 1$. The red zeros in the input were added for zero padding.

Each band of the input image is convolved with a corresponding kernel having size of $s \times s$. Afterwards, the results of all bands are summed to obtain the final output. In each band, a hyperparameter called *stride*, denoted as S, is used to control the used subset of the input for convolution by controlling the step width in both spatial dimensions. In Fig. 2.3, $S = 1$ is used. Commonly, the size of the output is smaller than the input size. To have an output of the same size as the input, many padding techniques can be used to enlarge the input, of which zero padding is the most frequently used one, as shown in Fig. 2.3. Convolution is formally defined as in eq. 2.16:

$$Y_o(i,j) = (X * F_o)(i,j) = \sum_k \sum_m \sum_n X_k(i \cdot S - m, j \cdot S - n) \cdot F_o^k(m,n) \tag{2.16}$$

where k is the index of input band, i, j are indices of the output Y and m, n are the indices of the kernel F_o in the k^{th} band. In the convolution operation, two important properties need to be emphasized:

- Compared to the input image, the size of the kernel is small. This makes the connectivity between output neurons and input neurons *sparse*, reducing an amount of unknown parameters compared to full connectivity. Despite the small receptive field of a small filter, it is possible to concatenate multiple convolution operations to enlarge the receptive fields.
- Convolution implies that the unknown parameters (i.e. weights in kernel) are *shared*. Sharing parameters also reduces the number of unknown parameters. Because of sharing parameters, the requirements of the model in terms of memory are further reduced.

The convolution mentioned so far is referred to as standard convolution in the remainder of this thesis. A variant called *depth-wise convolution* has been proposed in (Howard et al., 2017), which can reduce the number of unknown parameters further. An illustration of this type of convolution is presented in Fig. 2.4.

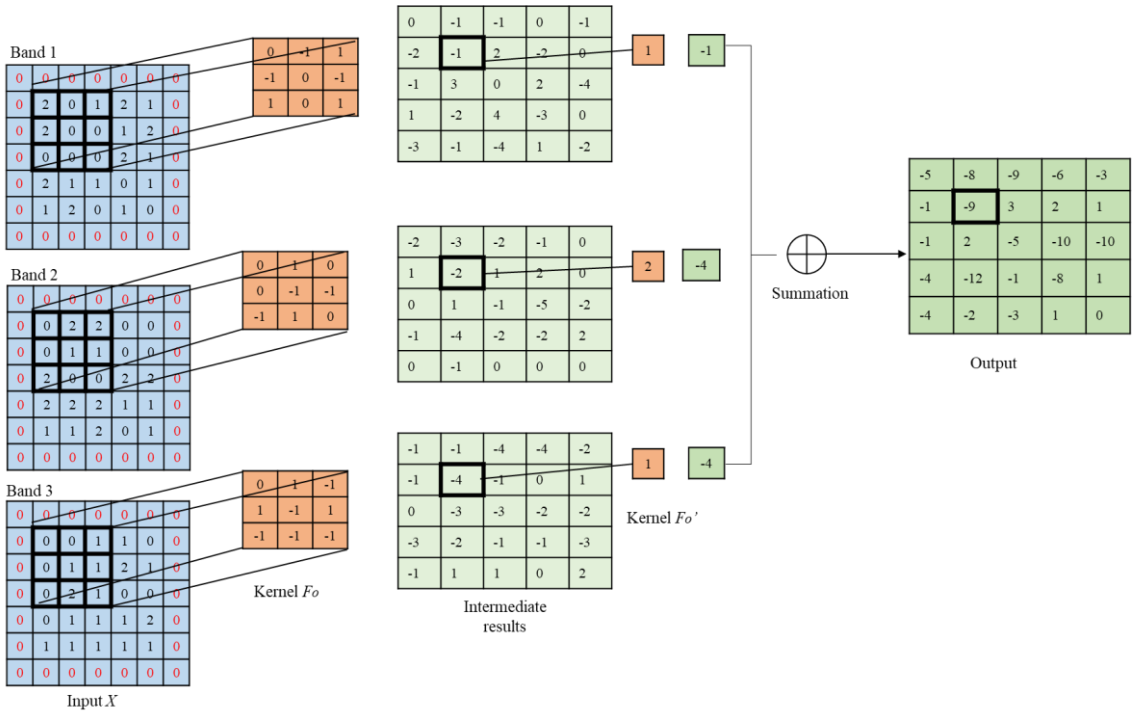


Figure 2.4: Illustration of a depth-wise convolution with an input $X \in R^{5\times5\times3}$, kernel $F_o \in R^{3\times3\times3}$ and kernel $F_o' \in R^{1\times1\times3}$ with a stride $S = 1$. The red zeros in the input were added for zero padding.

In depth-wise convolution, the convolutions using kernel F_o for each input band k are performed independently to obtain intermediate results, which again are convolved with kernel $F_o' \in R^{1\times1\times q}$ to obtain the final output. The procedure is formulized in the following two equations:

$$Y_k'(i,j) = (X * F_o)(i,j) = \sum_m \sum_n X_k(i \cdot S - m, j \cdot S - n) \cdot F_o^k(m,n) \tag{2.17}$$

$$Y_o(i,j) = (Y' * F_o')(i,j) = \sum_k \sum_m \sum_n Y_k'(i \cdot S - m, j \cdot S - n) \cdot F_o'^k(m,n) \tag{2.18}$$

where Y_k' is the intermediate output for the k^{th} band and the remaining symbols are the same as in eq. 2.16. In the standard convolution, the number of unknown parameters is $q \cdot O \cdot s^2$, whereas in the depth-wise convolution the number is $q \cdot s^2 + q \cdot O$. The ratio between the latter and the former is $\frac{1}{q} + \frac{1}{s^2}$. Due to the fact that $O \gg 1$ in a real application, the ratio is usually much smaller than 1, which indicates that the depth-wise convolution reduces the number of unknown parameters.

2.3.1.2. Pooling

A pooling function downsamples an input feature map. The two most frequently used pooling operations are *max-pooling* and *average-pooling*. A max-pooling operation delivers the maximum value within a rectangular neighborhood of the input, whereas an average-pooling operation delivers the average value in that neighborhood. One big advantage of pooling is that it helps to make the feature map invariant to small translations of the input. Invariance to local translations is useful when one is interested in whether some objects are present, rather than the precise location. Another advantage of pooling is that it enlarges the receptive field. Fig. 2.5 presents example for max-pooling and average-pooling. Like in convolution, the hyperparameter stride (S) is also used to control the degree of downsampling. Another hyperparameter is w to specify a neighborhood window of $w \cdot w$. In the example, rectangular neighborhood with $w = 2$ and stride with $S = 2$ is used. It has to be noted that S and w can have different values.

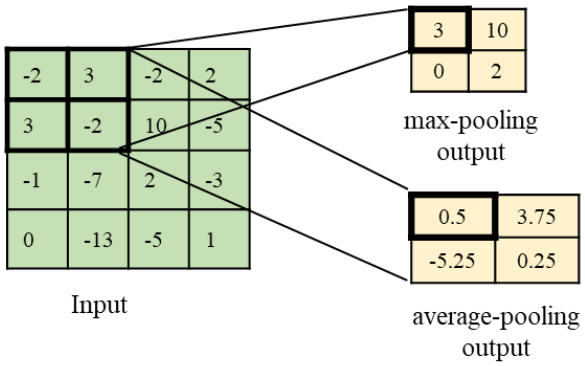


Figure 2.5: Illustration of max-pooling and average-pooling with $w = 2, S = 2$.

2.3.1.3. Batch Normalization

Batch normalization (BN) is a method to accelerate the convergence of training (Ioffe et al., 2015), which is adopted in this thesis. It is commonly applied after convolution layers but before the nonlinearity. Its goal is to standardize the value of each neuron by utilizing the mean and standard deviation of all values of the training samples in a minibatch. Given intermediate feature maps $\boldsymbol{x} \in R^{\Omega \times \mathrm{H} \times \mathrm{W} \times \mathrm{C}}$ in a minibatch with a size of Ω, each feature value in the last dimension of C are normalized. Let x_i^c with $i \in \Omega \times \mathrm{H} \times \mathrm{W}$ and $c \in \mathrm{C}$ denote a feature value to be normalized, the normalization procedure is carried out as follows:

$$u_c = \frac{1}{\Omega \times \mathrm{H} \times \mathrm{W}} \cdot \sum_{i=1}^{\Omega \times \mathrm{H} \times \mathrm{W}} x_i^c \tag{2.19}$$

$$\sigma_c = \sqrt{\frac{1}{\Omega \times \mathrm{H} \times \mathrm{W}} \cdot \sum_{i=1}^{\Omega \times \mathrm{H} \times \mathrm{W}} (x_i^c - u_c)^2} \tag{2.20}$$

$$\bar{x}_i^c = \frac{x_i^c - u_c}{\sigma_c} \tag{2.21}$$

The mean u_c and the standard deviation σ_c are computed by considering all values in the minibatch and both spatial dimensions. $\bar{x}_i^c$ is the normalized output which will have a mean of 0 and a standard deviation of 1.

During inference, to obtain the normalization output of x_i^c, the means and standard deviations based on all training samples need to be used. For N_T training samples, there will be m minibatches of size Ω with $m \cdot \Omega = N_T$. The final mean $u_{c,T}$ and the standard deviation $\sigma_{c,T}$ using all training samples are computed (Ioffe et al., 2015):

$$u_{c,T} = \frac{1}{\mathrm{m}} \cdot \sum_{j=1}^{\mathrm{m}} \mu_c^j, \tag{2.22}$$

$$\sigma_{c,T} = \sqrt{\frac{1}{\mathrm{m}} \cdot \frac{\Omega}{\Omega - 1} \sum_{j=1}^{\mathrm{m}} \sigma_{c,j}^2}, \tag{2.23}$$

where μ_c^j and $\sigma_{c,j}$ are computed using eqs. 2.19 and 2.20, respectively. At inference time, normalization of x_i^c is obtained by eq. 2.21 using $u_{c,T}$ and $\sigma_{c,T}^2$, computed over all training samples.

2.3.2. CNN for Image Classification

In the context of image classification, i.e. classifying one input image into one of pre-defined categories, a frequently used CNN is VGG-16 (Simonyan and Zisserman, 2015), which is shown in Fig. 2.6.

The input of the CNN is a RGB image having a size of 224 x 224 pixels. VGG-16 consists of five blocks, each consisting of either 2 or 3 convolutions, followed by ReLU activation and max-pooling. For all convolutions, a kernel size of 3 x 3, a stride of 1 and zero padding are used. The number of kernels at each layer is shown underneath the corresponding convolution blocks in Fig. 2.6. For all max-pooling layers, a neighborhood of 2 x 2 and stride of 2 are used.

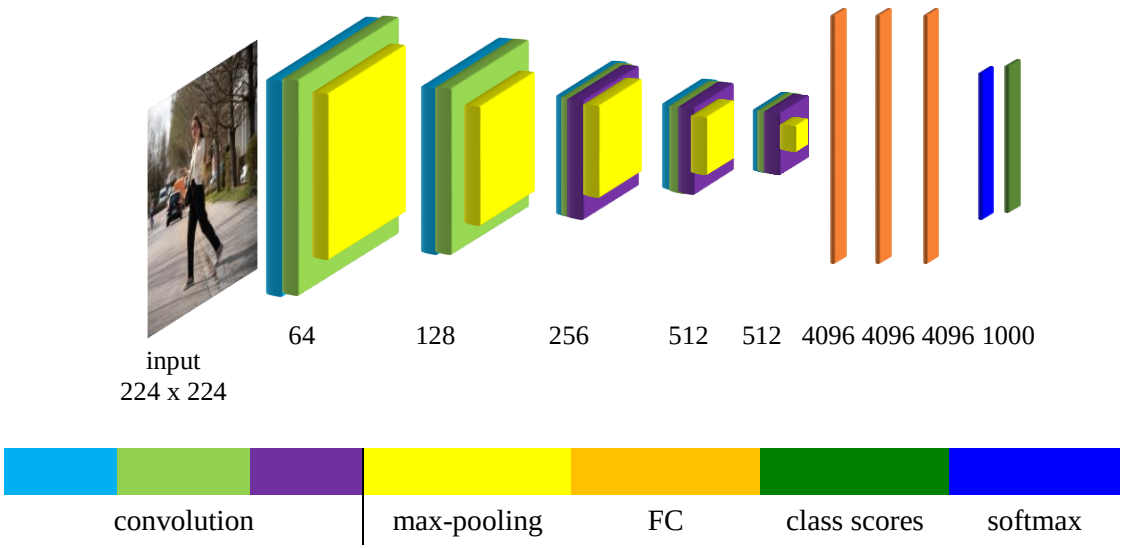


Figure 2.6: Structure of VGG-16. The figure is reproduced based on (Simonyan and Zisserman, 2015).

The first FC layer is connected to all neurons produced by the last max-pooling layer, and two additional FC layers follow. In each FC layer, there are 4096 neurons. Finally, to obtain the class scores, the neurons at the last FC layer are fully connected to an output vector with dimension of 1000 (the number of classes to be discerned in the original work). That output vector is normalized by a softmax function to obtain probabilistic scores (cf. eq. 2.8). To optimize the network, the cross-entropy loss (eq. 2.9) is used. The details of training are presented in Section 2.3.5.

It has to be noted that there is a total of about 53 million trainable parameters in VGG-16, only about 28% of which come from convolutional layers while the rest is related to the FC layers. This huge number of parameters requires a very large dataset for training, otherwise the training procedure might easily overfit. The proposed CNN for land use classification in this thesis is based on VGG-16, but requires only about 1 million trainable parameters.

2.3.3. CNN for Semantic Segmentation

To achieve semantic segmentation, i.e. prediction of class labels at pixel level, the CNN used for image classification can be expanded. One option is to use a fully convolution network (FCN; Long et al., 2015). In this section, FCN and one of its variants called U-Net are presented. The developments for land cover classification presented in this thesis are based on U-Net.

2.3.3.1. Fully Convolution Networks

Published by Long et al. (2015), FCN is the first known end-to-end CNN model applied for semantic segmentation. In FCN, there are no FC layers. The network consists of a series of convolution and pooling layers, acting as an encoder. The output of the encoder is a low-resolution feature map and needs to be upsampled to the resolution of the input image to make predictions at pixel level. For the purpose of upsampling, transposed convolutions or bilinear interpolation can be applied. Fig. 2.7 presents an overview of FCN based on VGG-16.

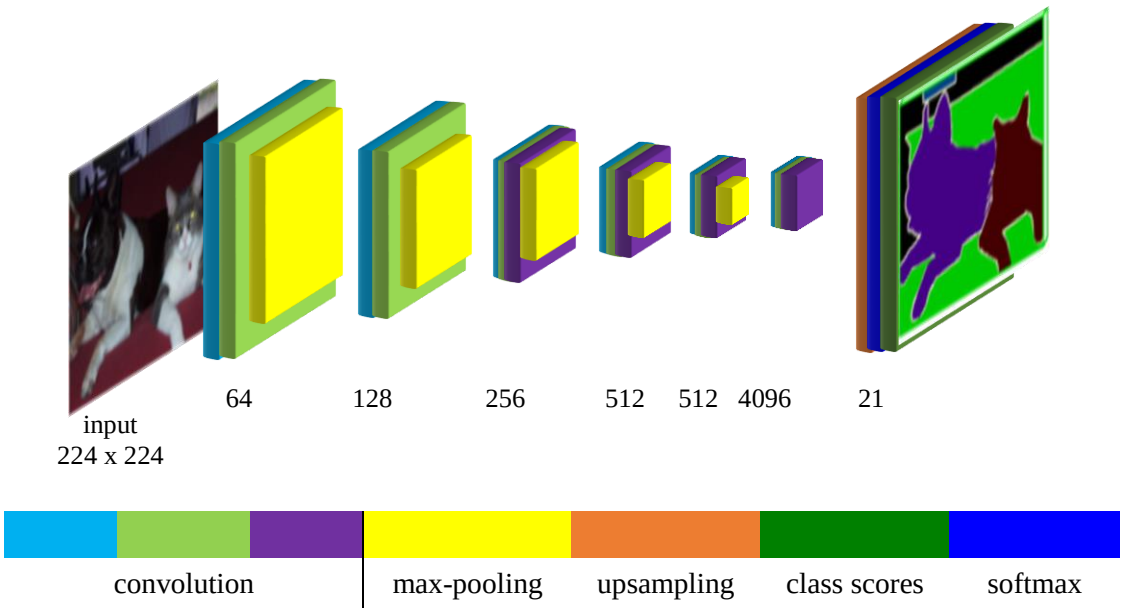


Figure 2.7: Structure of FCN using VGG-16 as encoder. The figure is reproduced based on (Long et al., 2015).

Based on the structure of the FCN in Fig. 2.7, Long et al. (2015) proposed a variant which uses *skip-connections* to connect the feature maps in the lower layers to the one at the final prediction layer, because it is assumed that the lower layers contain rich location information of objects, which is expected to be helpful for the delineation of object boundaries. Training is also based on minimization of the cross entropy loss (eq. 2.9), which, however, has to be determined for every pixel of the label maps. More details are presented in Section 2.3.5.

2.3.3.2. U-Net

A variant of FCN is U-Net (Ronneberger et al., 2015), which was originally designed for biomedical applications. This CNN employs a symmetric encoder-decoder structure: the encoder consists of a successive series of convolution blocks, each block having three convolution layers with ReLU activations and a max-pooling layer for downsampling at the end of that block. After four convolution blocks, the lowest spatial resolution is reached. The resultant feature maps are upsampled in a decoder. Symmetrically to the encoder, the decoder consists of four convolution blocks. In each decoder block, the upsampled feature maps of the previous stage are concatenated with the feature maps of the corresponding encoder block to form the input for the convolution. These connections between encoder and decoder are also called skip-connections. Similarly to FCN, transposed convolution is used for upsampling. At the end of the last convolution block, predictions are made. In the original implementation of U-Net, dense predictions were not made for the entire input image, but only for its central part. Fig. 2.8 presents an overview of the U-Net architecture from (Ronneberger et al., 2015). The sum of the cross entropy loss (eq. 2.9) for each image pixel is again used for training (cf. Section 2.3.5).

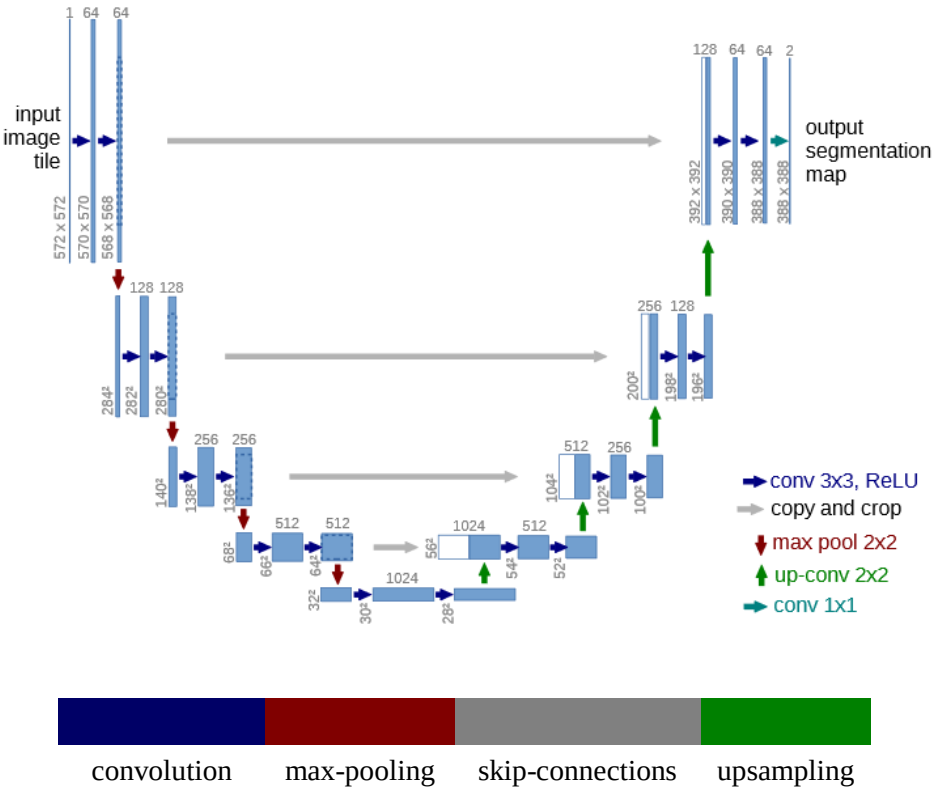


Figure 2.8: Overview of the U-Net architecture (Ronneberger et al., 2015). Each blue box corresponds to a multi-channel feature map. The number of channels is shown on top of each box. White boxes represent the feature maps from the encoder.

2.3.4. Training

To train a CNN, the cross entropy loss can be minimized, and the minimization can also be based on SGD and step learning policy (cf. Sections 2.2.3.2 and 2.2.3.3). Let **X** denote all images in a mini batch with size Ω, and let *H* and *W* denote the height and width of each image, respectively. There are *K* classes from the set $\mathbb{C} = \{C^1, \dots, C^K\}$ to be discerned. In the case of image classification, the cross entropy loss is:

$$L(\boldsymbol{\theta}) = -\frac{1}{\Omega} \cdot \sum_{i=1}^{\Omega} \sum_{k=1}^{K} t_i^k \cdot log(P(C^k|\mathbf{X}_i)) \tag{2.24}$$

where $P(C^k|\mathbf{X}_i)$ is the softmax output for class C^k of the i^{th} image in the mini batch, which is interpreted as a posterior. The variable t_i^k is 1 if the i^{th} sample belongs to class C^k, otherwise it is 0. Weight decay can be used to regularize the unknown parameters. Eq. 2.24 is identical to eq. 2.9 except that the loss is only computed over Ω samples of a minibatch and the input is an image.

In the case of semantic segmentation, the cross entropy loss is computed for all pixels in a mini batch, resulting in:

$$L(\boldsymbol{\theta}) = -\frac{1}{\Omega \cdot H \cdot W} \cdot \sum_{i=1}^{\Omega} \sum_{j=1}^{H \cdot W} \sum_{k=1}^{K} t_{i,j}^k \cdot log(P_j(C^k|\mathbf{X}_i)) \tag{2.25}$$

where $P_j(C^k|\mathbf{X}_i)$ is the softmax output for the j^{th} pixel in the i^{th} image to belong to class C^k. The variable $t_{i,j}^k$ is 1 if that pixel belongs to class C^k, otherwise it is 0. Similarly to image classification, weight decay can be used to regularize the unknown parameters.

Eqs. 2.24 and 2.25 serve as the basis for the loss functions developed in this thesis.

2.3.5. Data Augmentation

Commonly, CNN has large number of trainable parameters. Thus, a large training set is required to avoid overfitting. However, obtaining large training sets is economically expensive and time-consuming. To tackle this problem, one option is to apply data augmentation (Goodfellow et al., 2016) to generate synthetic training samples. In the context of aerial images, one can flip it horizontally and vertically. Other options are to rotate it by a given angle or to crop random windows or to apply scaling. In addition to geometric augmentation, one can also apply radiometric transformation to generate synthetic data. It has to be noted that in the context of geometric transformation, the reference label map has to be transformed using the same operation in semantic segmentation task. For the transformation of images, bilinear interpolation can be used, whereas for label maps nearest neighbor interpolation is used.

3. Related Work

In this chapter, related work of this thesis is presented. Section 3.1 discusses CNN for image classification and semantic segmentation, focusing on the works in Computer Vision. Subsequently, Section 3.2 focuses on land cover classification relying on CNN, whereas Section 3.3 describes recent work on land use classification. Finally, Section 3.4 summarizes the related work to identify the research problem that should be solved in this thesis.

3.1. CNN in general

3.1.1. Image Classification

LeCun et al. (1989) were the first authors to present a CNN trained using backpropagation. The CNN architecture called LeNet has two convolution layers and two FC layers and was designed for recognizing handwritten digits. However, the requirements w.r.t memory and training data prevented CNN from being widely applied for vision tasks at that time. In 2012, AlexNet (Krizhersky et al., 2012) achieved the lowest classification error in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2012; Russakovsky et al., 2015) and outperformed other methods by a large margin, starting a revolution in the field of computer vision. Compared to LeNet, AlexNet increases the network depth (5 convolution layers and 3 FC layers) and implemented several parametrization strategies (e.g. using large convolution kernels) to improve the classification ability of CNN. Simonyan and Zisserman (2015) proposed the VGG network with 19 layers, using smaller kernels of size 3 x 3. Their experiments showed that a series of successive convolution layers with small kernels could produce similar results as larger kernels, but requires fewer trainable parameters. Their findings established a trend for working with small-size kernels in CNN.

GoogLeNet (Szededy et al., 2015) was developed to achieve high classification accuracy while decreasing the computation burden compared to VGG. For that purpose, the concept of inception blocks is proposed, combining multi-scale convolutions using different kernels (5 x 5, 3 x 3 and 1 x 1) and concatenating the outputs to obtain the resultant feature maps. This reduces the number of trainable parameters to about 5 million, which is about 10% of the one in VGG-16 despite having a larger depth (22 layers). ResNet (He et al., 2016) allows to use even deeper networks by using bypass connections so that in each block, only the residual mapping beyond an identity mapping has to be learned. Several variants of ResNet were developed, varying the number of layers between 18 and 1202 layers (Huang et al., 2016). Although the depth is considerably increased, the number of trainable parameters is relatively low in comparison with other CNNs. For instance, ResNet50, which has 49 convolution layers and a single FC layer, has about 25 million parameters.

On the basis of ResNet, many researchers have developed new architectures. For instance, Szededy et al. (2016) proposed Inception-ResNet and Inception-V3/4, combining the ideas of GoogLeNet with the one of ResNet. DenseNet (Huang et al. 2017) employed the idea of residual block and improved it

by connecting one layer to all its subsequent layers in a feed-forward way. However, due to the large degree of connectivity, the number of kernels at each convolution layer is limited to reduce the computation burden. Many other CNNs based on ResNet are proposed for image classification; readers are referred to (Alzubaidi et al. 2021) for a comprehensive review.

So far, all mentioned methods predict one class label for an input image. However, one image showing one object can represent multiple labels which have semantic relations. For instance, an image showing a dog can be classified as *animal* at a first level, which can be refined as *dog*, *husky*, and so on. Deng et al. (2014) proposed the first CNN-based work for classification with semantic relations between different class labels. They defined a HEX (Hierarchy and Exclusion) graph to model different types of semantic relations. Two labels may have a hierarchical relation; they may be exclusive or overlapping. The CNN only has one output layer for all classes, but the HEX graph is considered in both, training and inference, to achieve a consistent classification result, e.g. to ensure that an image cannot be classified as showing a cat and a specific dog breed at the same time. However, this results in a very complex training and inference procedure. Guo et al. (2018) proposed a CNN-RNN (Recurrent Neural Network) strategy to address hierarchical classification. A CNN acts as a feature extractor and is trained to predict class labels at the coarsest semantic level. Then, the CNN features and the output of the coarse level are fed into a RNN structure which is used to propagate the information from the coarse level to finer levels. The consistency of the predictions with the class hierarchy is guaranteed. Their experiments on many datasets have shown that their methods achieve good results. However, information propagation is only considered from the coarse level to the finer labels, so that the semantic levels could not support each level mutually. Hu et al. (2016) proposed a network based on a CNN for hierarchical classification in three semantic levels, using a bidirectional message passing mechanism from the class scores of the coarse category to the class scores of the fine categories and vice versa. Thus, the class scores of each level are enhanced considering information from other levels of the hierarchy. On many datasets such as AWA (Lampert et al., 2014), NUS-WIDE (Chua et al, 2009), the authors achieved state-of-art results. However, their results are not guaranteed to be consistent with the class hierarchy. Despite this fact, the idea of their message passing mechanism is adapted for the hierarchical land use classification in this thesis, but it is expanded to guarantee consistency with the class hierarchy.

3.1.2. Semantic Segmentation

As mentioned in Section 2.3.3.1, the FCN proposed by Long et al. (2015) is the first known CNN for semantic segmentation. Based on FCN, many network variants were proposed in subsequent years. One popular family of CNN for semantic segmentation is based on an encoder-decoder structure, which is now widely used. Encoder-decoder networks apply convolution and pooling operations in the same way as standard CNN in the encoder part. After that, upsampling is carried out in a decoder network that is structured symmetrically to the encoder in order to obtain predictions at the resolution of the input image. As the proposed CNN in this thesis is based on such structure, this review is mainly focused on this family.

Noh et al. (2015) proposed an encoder-decoder architecture based on VGG-16, which is called DeConvNet. In the encoder, the input image is processed on successive convolution and pooling layers to obtain low resolution feature maps, which are subsequently decoded by upsampling and convolution to obtain pixel-wise predictions at the same resolution as input image. For upsampling, the authors utilized the indices which were recorded in the max-pooling layers in the encoder to reconstruct the signal of the corresponding size in the decoder. This work achieved the best results on the PASCAL VOC 2012 dataset (Everingham et al., 2012) at that time. SegNet, proposed by Badrinarayanan et al. (2017) and also using VGG-16 as its encoder, has the same structure as DeConvNet. In the decoder, a similar strategy for upsampling has been implemented. The experiments have shown that SegNet outperformed FCN in terms of overall accuracy by a large margin. In both networks, the pooling indices of the max-pooling layers were used to provide location information of objects, which is supposed to be relevant for classification of the pixels near object boundaries. The geometrical quality in the object boundaries of the SegNet results is still not very good. Other encoder-decoder based CNNs were designed to tackle this problem by connecting feature maps between lower layers to higher layers by elementwise addition, such as in the Stacked Deconvolutional Network (Fu et al., 2019) or in the LinkNet (Chaurasia and Culurciello, 2017). However, their networks suffered from a large number of trainable parameters in the order of 10 million.

So far, the mentioned CNNs were originally designed to segment natural images. U-Net (Ronneberger et al., 2015), also based on an encoder-decoder structure was designed for medical images. They were the first to introduce skip-connections between corresponding feature maps in the encoder and the decoder. At that time, U-Net achieved the best classification results for many datasets. Due to its superior performance, it has been the basis of many network variants. For instance, Cicek et al. (2016) as well as Milletari et al. (2016) proposed a CNN to segment 3D medical images; Zhang et al. (2018) developed a road segmentation and extraction algorithm to process aerial images. Instead of encoder-decoder based CNNs, there are many other CNN using different strategies for semantic segmentation, such as DeepLab (Chen et al. 2017) or feature pyramid network (FPN; Lin et al., 2017b). It is not the goal to review these papers in this thesis. For a comprehensive review, refer to Minaee et al. (2021).

3.2. Land Cover Classification

In the context of land cover classification, each image pixel is assigned a class label, thus it is a variant of what is called *semantic segmentation* in Computer Vision. Before the development of FCN, it was solved by CNN relying on a patch-based strategy. To achieve that goal, an image is divided into many patches, each of them being representative for the label of its central pixel. Each such patch is classified by a CNN. To obtain dense prediction for that image, different strategies have been proposed based on the patch predictions.

Längkvist et al. (2016) applied this procedure in a sliding window approach, making each pixel the center of such a patch to be classified by a CNN. The input data of their approach consists of high-resolution multispectral satellite orthophotos and a DSM of a small city. The proposed CNN requires an input of 25 x 25 pixels and consists of five convolution and pooling layers. The last pooling layer is

connected with a FC layer to obtain a feature vector, which is converted to class scores. To obtain dense predictions of a large image, the authors slide a window from top-left to bottom-right with a stride of 1 and classify every patch. They obtained an overall accuracy of about 95%. However, a stride of 1 makes the prediction a very time-consuming procedure. A similar procedure was applied by Paisitkriangkrai et al. (2016), but with a stride larger than 1. They built a CNN which requires an input of 64 x 64 pixels and has four blocks, each block consisting of a convolution layer, a ReLU and a max-pooling layer. The last pooling layer is connected to two consecutive FC layers with a dimension of 128. Finally, the output of the last FC layer is converted to class scores for classification. To obtain dense predictions for a large input image, they applied the patch-based classification to every n^{th} pixel in the image, using bilinear interpolation of class scores to get prediction for every pixel. Furthermore, they post-processed the dense predictions using a Conditional Random Field (CRF; Kumar and Hebert, 2006). Their approach was tested on the two public datasets of Vaihingen and Potsdam from the ISPRS 2D semantic labeling benchmark (Wegner et al., 2015), which consist of very high resolution aerial images and DSM. Paisitkriangkrai et al. (2016) achieved an overall accuracy of 88% when differentiating five land cover classes. However, despite the usage of a stride of larger than 1, the prediction of class labels for a large image is still time-consuming. Furthermore, a CRF-based post-processing was required to achieve high quality results.

Patch-based CNN classification has also been applied for large-scale remote sensing image classification. Maggiori et al. (2017a) proposed a CNN consisting of three convolutional layers and one FC layer to classify satellite images with labels derived from Open Street Map (OSM); Postdadjian et al. (2017) proposed a CNN consisting of four convolutional layers and one FC layer for pixel-wise land cover classification. They used very high resolution satellite images as input and classified five land cover types to detect changes by comparing the classifications over two time epochs. More papers relying on patch-based classification can be found in the review of (Zhu et al., 2017).

Although patch-based dense classification can deliver good results, the dense prediction of a large image is a very time-consuming process, because it requires multiple repeated convolutions.

This problem is overcome by FCN (Long et al., 2015). It has been increasingly used for land cover classification, often being compared to patch-based methods. Kampffmeyer et al. (2016) proposed a FCN and a patch-based CNN for dense prediction, using the Vaihingen dataset from the ISPRS 2D semantic labeling benchmark for evaluation. Their FCN consists of three blocks, each having two consecutive convolution layers with associated ReLU activations and a final max-pooling layer. Finally, the feature maps at the lowest resolution are upsampled to the resolution of the input image. The upsampled feature maps are the argument of a softmax function, which delivers class probabilities for each pixel. The patch-based CNN was adapted from (Paisitkriangkrai et al., 2016), using an input size of 65 x 65 pixels. The results showed that FCN-based predictions outperform those of patch-based CNN by about 3% in terms of overall accuracy. Volpi et al. (2017) investigated patch-based CNN and FCN as well. Their patch-based CNN consists of four convolution layers and one FC layer, and their FCN is based on an encoder-decoder structure similarly to (Noh et al. 2015). The encoder is identical to the one used in the patch-based CNN and the decoder is symmetric to the encoder. The Vaihingen and Potsdam datasets were used for testing purpose. In both sites, FCN-based predictions outperform those of patch-based CNN by about 4% in terms of overall accuracy. Similar findings of the differences between FCN

and patch-based CNN have been observed in (Sherrah, 2016). Currently, state-of-the-art methods mainly rely on FCN-based approaches.

However, one of the main problems of FCN-based methods is the loss of spatial resolution, which is caused by the downsampling of pooling layers in the CNN. As a consequence, the geometrical accuracy of the object boundaries is affected in a negative way. To solve this problem, Long et al. (2015) retained location information by upsampling feature maps in shallow layers having high spatial resolution feature maps, and fused them with the final feature maps for classification. This design was the first attempt to improve the location information. Many researches have proposed different approaches to mitigate the problem of the loss of spatial accuracy as well.

One approach is to avoid downsampling in a CNN. For instance, Sherrah (2016) proposed a FCN which applies pooling without downsampling by using a stride of 1. As the number of layers increases, regular pooling with a stride of more than 1 can increase the receptive field, which is supposed to support classification by capturing more local context of the input. In the variant without downsampling, capturing a receptive field of the same size requires an exponential increase of the filter size in the convolution and pooling operations, resulting in a large amount of trainable parameters, which might lead to overfitting. To avoid this problem, Sherrah (2016) used the dilated convolution (Chen et al., 2015) to achieve a larger receptive field while keeping a small filter size for convolution. The results showed that the FCN outperforms the regular FCN by up to 2% in terms of overall accuracy for the Vaihingen and Potsdam datasets. However, the method suffers from increased requirements with respect to GPU memory and computation time. The FCN without downsampling is reported to require about 50 times more training time than the regular one.

Another strategies tries to model the detection of object boundaries explicitly in a FCN. Marmanis et al. (2018) extracted edge maps from images by applying a Holistically-Nested Edge Detection (HED) framework (Xie et al., 2017). Edge maps are concatenated with images as input for a FCN, and the outputs are combined for the final class prediction. Their experiments on the Vaihingen and Potsdam datasets showed that explicitly integrating edge maps to the FCN improves the overall accuracy by about 4%. Although they achieved good results, there was a cost of multiple training stages (first HED, subsequently FCN) and a huge number of parameters (their best model has about $8.1 \cdot 10^8$ trainable parameters). Volpi and Tuia (2018) took another step to model object boundaries by building a multi-task CNN: one task is semantic segmentation (i.e. land cover classification), and the other one is learning the object boundaries. They claimed that the prediction of boundaries supports the semantic segmentation tasks, because the boundaries naturally embed shape information and contours of semantic classes. Their methods were tested and validated using the Vaihingen and Zeebrugges datasets (Campos-Taberner et al., 2016). They achieved good results for both datasets. However, their approach required a CRF-based post-processing to combine the predictions of boundary and semantic classes, preventing an end-to-end learning. Cheng et al. (2017) proposed a CNN based on SegNet (Badrinarayanan et al. 2017) for semantic segmentation of remote sensing harbor images. They built two classifiers: a boundary classifier and a multiclass classifier, similar to Volpi and Tuia (2018). Furthermore, an additional loss term is added for regularization. For every pixel, the class scores of its neighbor pixels in an 8-neighborhood are enforced to be close to its own scores if the probability of that neighbor pixel to be an edge is low. However, a two-stage training procedure of their work was required, i.e. while updating the parameters of the boundary classifiers, the unknown parameters of the multiclass classifiers were not

updated, and vice versa. This fact implies that procedure is potentially time-consuming. Zheng et al. (2020) proposed a multi-task CNN based on ResNet for land cover classification, employing an additional boundary classifier as well. They achieve state-of-art results in three public datasets: Cityscapes (Cordts et al., 2016), WHU Aerial Building (Ji et al., 2018) and Vaihingen. In their ablation study, their proposed boundary classifier was shown to improve the semantic segmentation by about 1% in terms of mean IoU (intersection over union) in Cityscapes. However, their ResNet-101-based CNN has a large number of unknown parameters, which can easily lead to an overfitting if no sufficient number of training data is available. In the remote sensing community, there are more publications related to utilizing boundary detection to improve multi-class classification, e.g. (Jiang et al. 2017; Wang et al. 2017; Liu et al. 2018; Diakogiannis et al. 2020). Readers are recommended to read the original works for detail. One common problem of these works is that their number of unknowns is huge, thus, a large number of training data is required to avoid overfitting.

As an alternative to the approaches mentioned so far, another promising strategy is to use *skip-connections* to mitigate the problem of the loss of geometrical accuracy. The rationale of skip-connections is that in the lower layers of a CNN there is precise location information which is lost in the higher layers due to downsampling. Thus, connecting the lower layers with higher layers is expected to help to recover the location information. Skip-connections are indeed a built-in feature of many FCN variants. For instance, in the original U-Net, feature maps of the encoder are connected to the corresponding decoder layers by concatenation, which is followed by standard convolution operations.

Liu et al. (2018) used VGG-16 and ResNet101 as their backbone CNN to encode the input and connected feature maps of the lower layers to those in the upsampled higher layers via elementwise addition. In their ablation study, they found that the difference between the CNN with skip-connections and the one without skip-connections is up to +1.5% in terms of the mean F1 score. Diakogiannis et al. (2020) proposed a network combining U-Net and the residual blocks of ResNet, resulting in their ResUNet-a, in which U-Net-style skip-connections are adopted. Zheng et al. (2020) connect feature maps of the lower layers to those in higher layers via concatenation. Both works achieved results on par with state-of-art methods. Skip-connections seem to be a standard component in most pixel-level classification CNNs, for instance, in (Audebert et al. 2018; Marmanis et al. 2018; Sun et al. 2019; Wittich and Rottensteiner 2019; Liu et al. 2020; Voelsen et al. 2020). However, all methods mentioned so far rely on either elementwise addition or concatenation as skip-connections. The potential of connecting feature maps in a learnable way has not been exploited.

Another problem is related to the imbalanced distribution of classes in training sets. The unknown parameters tend to fit the dominant classes, i.e. those having a large number of training samples. As a consequence, the classifier will have a bad performance for the underrepresented classes. To tackle this problem, many researchers used a *weighted cross entropy* loss function for optimization. For each class, a weight is introduced as a penalty term in the cross entropy loss function. The basic idea is to use the weights to increase the losses of samples from underrepresented classes so that the CNN will pay more attention to them during training. Eigen et al. (2015) used a class weight which is proportional to the inverse of the number of the samples of that class in the training data for semantic segmentation; such strategy was also used in (Badrinarayanan et al., 2017) when classifying street-view images and in (Ren et al., 2020) when classifying satellite images. In both works, pixel-wise classification is desired. Voelsen et al. (2020) firstly computed the ratio between the total number of training samples and the

number of samples belonging to a class, and then applied the natural logarithm to that ratio to obtain the class weight. Later, Voelsen et al. (2021) normalized the class weights by dividing them by the maximum class weight. Their goal is to achieve pixel-wise classification using satellite images. However, all class weights mentioned so far have to be determined from the training data prior to training, and these weights are not updated in the training procedure. Using adaptive weights in the form of the so-called focal loss was firstly proposed in (Lin et al., 2017). Classes having sufficient number of training samples are more likely to be correctly identified, thus, the corresponding class probabilities of the samples are close to one. However, classes having few training samples are harder to be correctly identified, thus, the corresponding class probabilities of the samples tend to be much smaller than one. For each sample, the focal loss utilizes the difference between one and the probability of the reference class as the weight. If a sample of the reference class is classified correctly, i.e. the corresponding class probability is close to one, the weight will be small, otherwise it will be relatively large. These weights are updated in each training iteration. In the binary classification task, their experiments have shown that the proposed focal loss outperforms the standard cross entropy loss and the loss using fixed class weights by a large margin. In this thesis, an extension of the focal loss for multiclass classification is proposed.

Beyond multi-spectral orthophotos (e.g. RGB, IR), height data such as DSM or DTM are often available in remote sensing applications. Thus, the way of fusing these data for land cover classification is also a topic of research in the remote sensing community. One naïve approach is to stack these data to form a multi-channel input, which is presented to a CNN. Examples for such an approach are (Längkvist et al. 2016, Volpi and Tuia 2018, Sun et al. 2019). Compared to the results relying solely on orthophotos, the ones using additional height data are better in all cases. Many researchers claim that a naïve approach might not be the optimal fusion strategy, thus they develop strategies to fuse the intermediate feature maps generated from multi-modal input data. Hazirbas et al. (2016) proposed a SegNet-based CNN having two encoder branches and one decoder, where the two encoders process the image and height data in parallel. They referred to the encoder of the image as the primary branch and to the other one as the secondary branch. In each encoder stage, the feature maps of the secondary branch are added to the primary ones. Audebert et al. (2018) adapted that approach by using a different fusion scheme: the feature maps are concatenated and then convolved with a 3 x 3 kernel to obtain the fused output. Audebert et al. (2018) differentiate two types of fusion schemes: early fusion and late fusion. In the former, the feature maps are fused at the encoder stage, similarly to (Hazirbas et al., 2016). In the latter, there are independent encoders and decoders for each input to obtain the final predictions, and the predictions are fused by computing the average of the class scores. Their results show that early fusion performs slightly better than late fusion. In fact, another advantage of the early fusion scheme is a lower requirement of memory and fewer trainable parameters. In this thesis, an early fusion scheme similar to (Audebert et al., 2018) is adapted to fuse multi-modal data, leading to a two-branch encoder-decoder CNN for land cover classification.

3.3. Land Use Classification

In recent years, the classification of land use was mainly tackled by supervised methods, because they are more easily transferable to other scenes than model-based techniques (e.g. Banzhaf and Hoefer, 2008). In this review of land use classification, only supervised methods are considered. Section 3.3.1 presents an overview of related work using methods not based on CNN. The goal of presenting these methods is to give a brief summary how land use classification was solved before CNN. Section 3.3.2 focuses on methods relying on CNN.

3.3.1. Methods not based on CNN

Existing methods for land use classification differ by the primitives to be classified, the data sources, the features used for classification and the classifiers used to predict the class labels. Many approaches rely on a strategy consisting of two steps: the land cover is determined from remote sensing data in the first stage, and the results are used to support the classification of land use in the second stage, e.g. (Gerke et al. 2008; Hermosilla et al. 2012; Helmholz et al. 2014). Random Forests (RF; e.g. Walde et al. 2014; Albert et al. 2014) and Support Vector Machine (SVM; e.g. Montanges et al. 2015) are frequently used for classification in this context. *Hand-crafted features* of the primitives to be classified have to be extracted on the basis of the given data.

Here, two types of features are discriminated: *Primitive-specific features* describe characteristics of a primitive and are directly derived from sensor data. Examples are the spectral values and the statistic characteristics, e.g. minimum, maximum and standard deviation in a local neighborhood or inside a segment. Texture features, e.g. based on Grey Level Co-occurrence Matrix (GLCM; Haralick et al., 1973), structure features such as Histogram of Oriented Gradients (HOG; Freeman and Roth, 1995), and 3D features derived from DSM are widely used. *Context feature*s describe the relationship of a primitive with its neighbors. Two types of metrics have been proposed: *spatial metrics* and *graph-based metrics*. Spatial metrics quantify the spatial configuration of land cover elements within a land use object, e.g. in the form of the ratio of the area of all building pixels to the total number of pixels (Hermosilla et al., 2012) or in the form of the relative position of a building segment with respect to the land use boundaries or other building segments (Novack and Stilla, 2015). Graph-based metrics, derived based on graph theory, describe the frequency of the local spatial arrangement of land cover elements within a land use object. Barnsley & Barr (1996) firstly introduced these features for the classification of land use. Inside a land use object, a graph is built by connecting a land cover element to its neighbors via edges. Later, the *adjacent-event matrix* is applied to describe the neighborhood relationships of the land cover elements at pixel level. Features can be derived from that matrix such as the normalized number of edges between two land cover classes. Walde et al. (2014) adapted the adjacent-event matrix by building a segment-based graph instead of a pixel-based one.

Context features incorporate context information in an implicit way in a classification process. To explicitly incorporate context information, graphical models can be applied e.g. CRF. For instance,

Albert et al. (2014) presented a two-step land use classification approach using CRF, which was extended to an integrated approach in (Albert et al., 2017). In Albert et al. (2015), two separate CRFs are applied for land cover and land use classification. Both CRFs model spatial dependencies between neighboring sites, i.e. pixels (Albert et al., 2014) or super-pixels (Albert et al., 2015) in the case of land cover and polygons from the database in the case of land use classification. Novack and Stilla (2015) classified urban land use using high-resolution space borne synthetic aperture radar (SAR) data. The initial class labels are obtained by computing the arithmetic mean of the predictions delivered by three different classifiers (RF, logistic regression and nearest neighbors). A CRF is applied for post-processing to obtain the final result. Montanges et al. (2015) proposed an approach for the classification of urban land use using satellite images and a DSM. Their CRF models the combination of the output of a SVM classifier and structural information describing the spatial relations of neighboring land use cells. However, the references mentioned so far do not model the complex semantic relationship between land cover and land use explicitly in a graphical model. Albert et al. (2017) proposed a two-layer CRF (a land cover layer and a land use layer) for the simultaneous classification of land cover and land use. In the land cover layer, the nodes correspond to super-pixels, whereas in the land use layer they are objects from a geospatial database. Beyond the intra-layer edges which model the spatial dependencies between neighboring image sites, inter-layer edges are additionally introduced to model the semantic relation between land cover and land use sites, leading to higher order cliques. With respect to land use, the method achieved an overall accuracy in the order of 80% using two datasets when differentiating 10 classes.

However, all methods mentioned so far suffer from a relatively low overall accuracy (in the order of 70% - 90%; the larger the number of classes to be discerned, the smaller the overall accuracy could be achieved) for land use classification. Thus, the classification results might not be sufficiently good for practical application. Another problem of these works is that they only differentiate a small number of land use classes (between 2 and 11), and these classes do not have hierarchical relations, which may also hamper practical relevance.

3.3.2. CNN-based Methods

All methods described in Section 3.3.1 rely on hand-crafted features. In the context of CNN, features are learned from training data, and a simple classifier (e.g. softmax) can be used for the classification purposes. Compared to the methods described in Section 3.3.1, CNN methods improve classification accuracy by a considerable margin, indicating that the learned features are more discriminative than the hand-crafted ones.

In this thesis, the goal of the land use classification is to predict the land use of the objects stored in geospatial databases. Commonly, these objects are stored in the form of polygons. However, the polygons are typically irregular-shaped and vary considerably in terms of their geometric extent. As a CNN requires input patches of regular size, the problem of converting polygons to patches needs to be tackled. Zhang et al. (2018) proposed a segment-based approach to determine land use from remote sensing data. The authors started with an initial non-semantic image segmentation using mean-shift

(Comaniciu and Meer, 2002). The resultant segments are considered to correspond to objects for which land use is to be predicted. These segments are split into rectangular patches using the moment bounding box method of (Zhang and Atkinson, 2016). These patches, which consist of either 48 x 48 or 128 x 128 pixels, are classified independently from each other using a CNN. The final class label of each segment is determined by combining the predictions for all patches by a simple majority vote. Later, Zhang et al. (2019) adapted their methods by proposing a joint deep learning framework for classifying land cover and land use simultaneously in an iterative approach. However, they required a pre-processing step to obtain polygons (i.e. the segments), which differs from the case presented in this thesis, where the polygons are already available. The classification task in this thesis is more similar to (Huang et al., 2018), where land use is predicted relying on the availability of polygons representing urban blocks. In that work, each polygon is represented by a series of rectangular processing units of 227 x 227 pixels, which are positioned inside the polygon on the basis of a skeleton. These processing units are classified independently from each other using a CNN. The final prediction for a polygon is obtained by computing the arithmetic mean of the class scores of all corresponding processing units. Nonetheless, Huang et al. (2018) only considered urban areas and have not tested their methods using rural areas. Although the mentioned methods proposed ways to obtain regular patches, there are other possibilities to achieve this goal, which will be investigated in this thesis.

Many works only differentiate a small number of classes. For instance, in (Zhang et al., 2018) and (Zhang et al., 2019), only 10 land use classes were discerned, whereas in (Huang et al., 2018), 13 land use classes were differentiated, and these classes have no hierarchical relations. In the publicly available datasets for land use, the number of classes is also limited. For instance, in the UC Merced Land Use dataset (UCM; Yang and Newsam, 2010), 21 land use classes are considered and for each class, 100 images are available; in the Banja-Luka dataset (Othman et al., 2016), 6 classes are of interest and for each class about 90 images are available. Using these datasets, CNN-based methods achieved high accuracies (>90% in overall accuracy), such as in (Hu et al., 2015), (Luus et al., 2015) and (Nogueira et al., 2017). However, even though the results are very good, they are based on a very small number of samples only, which means that their significance for real application still needs to be shown. Another problem is that the land use information is often more detailed. Often, the land use is encoded on the basis of hierarchical object catalogue in a database. For instance, in the land use layer of the German cadastre, about 190 categories are differentiated (AdV, 2008). It is an important fact that many topographic databases contain land use information in different semantic levels of abstraction. At the coarsest level, only a few broad classes such as *settlement*, *traffic* or *vegetation* are differentiated. At the finer levels, these classes are hierarchically refined, and the full number of different categories is only differentiated at the finest level of the class structure. Albert et al. (2016) investigated the maximum semantic resolution that their CRF-based land use classification method could achieve. They divided the land use categories into two levels, both corresponding to mixtures of the three coarsest semantic levels according to (AdV, 2008). Starting from a classification at the coarse level, they refined one coarse category after the other: in a greedy iterative procedure, one category is split into the maximum set of sub-categories and then sub-categories are merged if the results indicate they cannot be separated. As a result, Albert et al. (2016) obtained a class structure consisting of 10 categories from different semantic levels of the object catalogue and concluded that this is the largest set of classes that can be separated using their approach. However, their method did not rely on CNN and on the other hand, they

did not predict land use classes at multiple semantic levels. In addition, the greedy iterative procedure of the refinement results in multiple classification models and is very time-consuming. To the best knowledge of the author, the potential of classifying land use objects at multiple semantic levels while keeping consistency with the class hierarchy has not been exploited yet.

3.4. Discussion

The previous sections have given an overview about existing and related work of land cover and land use classification. This section briefly summaries open questions and unsolved problems in both classification tasks. Based on these open points, the methods proposed in the following chapter will tackle the identified research gap.

3.4.1. Land Cover Classification

In a multiclass classification task, one prominent issue is the imbalanced class distribution of training data. For instance, in the Vaihingen dataset, more than 25% of the pixels belong to the class *building*, whereas less than 3% of the pixels belong to the class *car*. Similar issues have also been found in the Potsdam dataset. This poses a problem for a CNN: the trained CNN is more likely to predict the dominant classes. To tackle this issue, many existing works use weighted cross entropy loss. The idea is to add an additional penalty term in the standard cross entropy loss (eq. 2.19). The weights of the penalty term for underrepresented class are larger than the ones for well-represented classes. There are many ways to obtain the weights for each class, e.g. the inverse of the frequency of a class in the training data (Eigen et al., 2015). In remote sensing applications, works using weighted cross entropy loss are (Voelsen et al. 2020; Ren et al., 2020). However, the weights have to be determined before training based on all training data and are not updated during training. Inspired by the focal loss proposed in (Lin et al., 2017), which is originally applied for binary classification, an adaptive weight for each class during training is proposed in this thesis by extending the focal loss to the multiclass classification problem. As samples of underrepresented classes are supposed to be harder to be classified, their weights will be relatively large in comparison with the ones of samples of well-represented classes, so that their contributions to the updates of the unknown parameters are increased.

To tackle the problem of the loss of geometric accuracy caused by pooling, this thesis focuses on the development of skip-connections instead of explicitly learning object boundaries, because the latter approach commonly requires an additional CNN structure. In most existing works, skip-connections are implemented via elementwise addition of feature maps, e.g. in (Liu et al. 2018; Audebert et al. 2018; Marmanis et al. 2018; Zheng et al. 2020). However, the combinations of feature maps can be learned, which has not been exploited so far. It is expected that the classification will profit from learning. To implement learnable skip-connections, the feature maps at different layers are convolved via depth-wise

convolution and then concatenated. Subsequently, a 1 x 1 convolution is applied to combine the resultant feature maps.

3.4.2. Land Use Classification

One big problem of existing methods for land use classification is that they only differentiate a small number of classes, for instance, in (Helmholz et al. 2014; Novack and Stilla, 2015; Albert et al., 2017), two, eleven and ten land use classes are differentiated, respectively. However, the object catalogues of land use databases may be much more detailed. For instance, in the land use layer of the German cadaster, about 190 categories are differentiated (AdV, 2008). One important fact is that many topographic databases contain land use information in different semantic levels of abstraction. Albert et al. (2016) attempted to classify objects into multiple semantic levels using their CRF-based method in a greedy iterative procedure. They obtained a class structure which is a mixture of 10 categories from different semantic levels of the object catalogue according to (AdV, 2008), because their method cannot separate many sub-categories. In the context of CNN for land use classification, many methods differentiate also a small number of classes only, e.g. (Zhang et al., 2018; Zhang et al., 2019; Huang et al., 2018). Classifying land use in multiple semantic levels and keeping consistent with the class hierarchy based on CNN have not been exploited so far, which motivates the developments in this thesis. To solve this problem, a new CNN is proposed. This CNN is not only limited for one-level land use classification, but is also extended for hierarchical land use classification. By using hierarchical predictions of one land use object, it is expected to facilitate the verification process with more details.

There are works in Computer Vision to solve the problem of hierarchical classification. However, they suffer from some limitations such as a complex training and inference procedure (Deng et al., 2014), no guarantee of consistency with the class hierarchy (Hu et al., 2016) and an additional RNN structure which may prevent training from converging in a good point in parameter space (Guo et al., 2018). In this thesis, different strategies are proposed to achieve hierarchical land use classification. Here, two CNN network variants are proposed. First, a multi-task learning variant is proposed to predict land use objects at multiple semantic levels independently. To guarantee the consistency of the class hierarchy, a so-called *Joint Optimization* strategy is proposed, in which predictions are made by selecting the tuple of classes over all semantic levels which has the maximum joint class score. Second, a variant is used to predict the class labels at the finest level of the class structure. These predictions are used to control the predictions at the courser levels by using a so-called *fine-to-coarse* post-processing step, so as to guarantee the consistency of the class hierarchy.

Regarding to the *Joint Optimization* strategy, a new loss function is proposed that respects the hierarchy of the class labels according to the object catalogue for each training object. The rationale is that the joint score of the reference hierarchical tuple is maximized and the joint scores of other tuples are minimized during training. In the loss function, penalty terms similar to focal loss are introduced so

as to mitigate the problem of the imbalanced class distribution in the datasets, due to the fact almost all datasets suffer from this problem.

Land use objects stored in geospatial databases often have irregular shapes and large variations in terms of their geometrical extent. For instance, a *road* object can be very large and thin, whereas *settlement* objects may cover both, very large and quite small areas. In the context of classification relying on CNN, inputs of regular and constant size are required. One way to achieve this goal is to follow Zhang et al. (2019) or Huang et al. (2018). The former uses non-semantic image segmentation and moment bounding box as pre-processing, whereas the latter requires a skeleton as input for pre-processing. In both works, the pre-processing produces key points, on the basis of which the rectangular patches are generated for CNN. One problem of both methods is that they do not use the shape information as input, which is expected to be helpful for the classification. In this thesis, a binary mask is proposed to represent the polygon shape. To obtain regular patches, two approaches to convert irregular polygons to regular patches are proposed, i.e. tiling and scaling. The tiling approach keeps the image resolution unchanged but will probably lose the shape information of large polygons. In contrast to tiling, the scaling approach keeps the shape information of polygons. It is expected that both approaches are complementary to each other when making predictions for polygons via an ensemble method. Another problem in (Zhang et al., 2018) is that the step of image segmentation might produce undesired polygons due to segmentation errors, which may prevent their results from being useful for practical application.

4. Methodology

This chapter presents the core of this thesis and proposes a framework for the verification of the geospatial databases. Section 4.1 gives an overview of the method. Section 4.2 presents the proposed CNN for land cover classification, whereas Section 4.3 presents the one for land use classification.

4.1. Overview

The proposed framework for the verification of the land use information for objects in a geospatial database is illustrated in Fig. 4.1. The approach consists of four main components: the input data, the CNN-based methods for land cover and hierarchical land use classification, and the verification of the geospatial database.

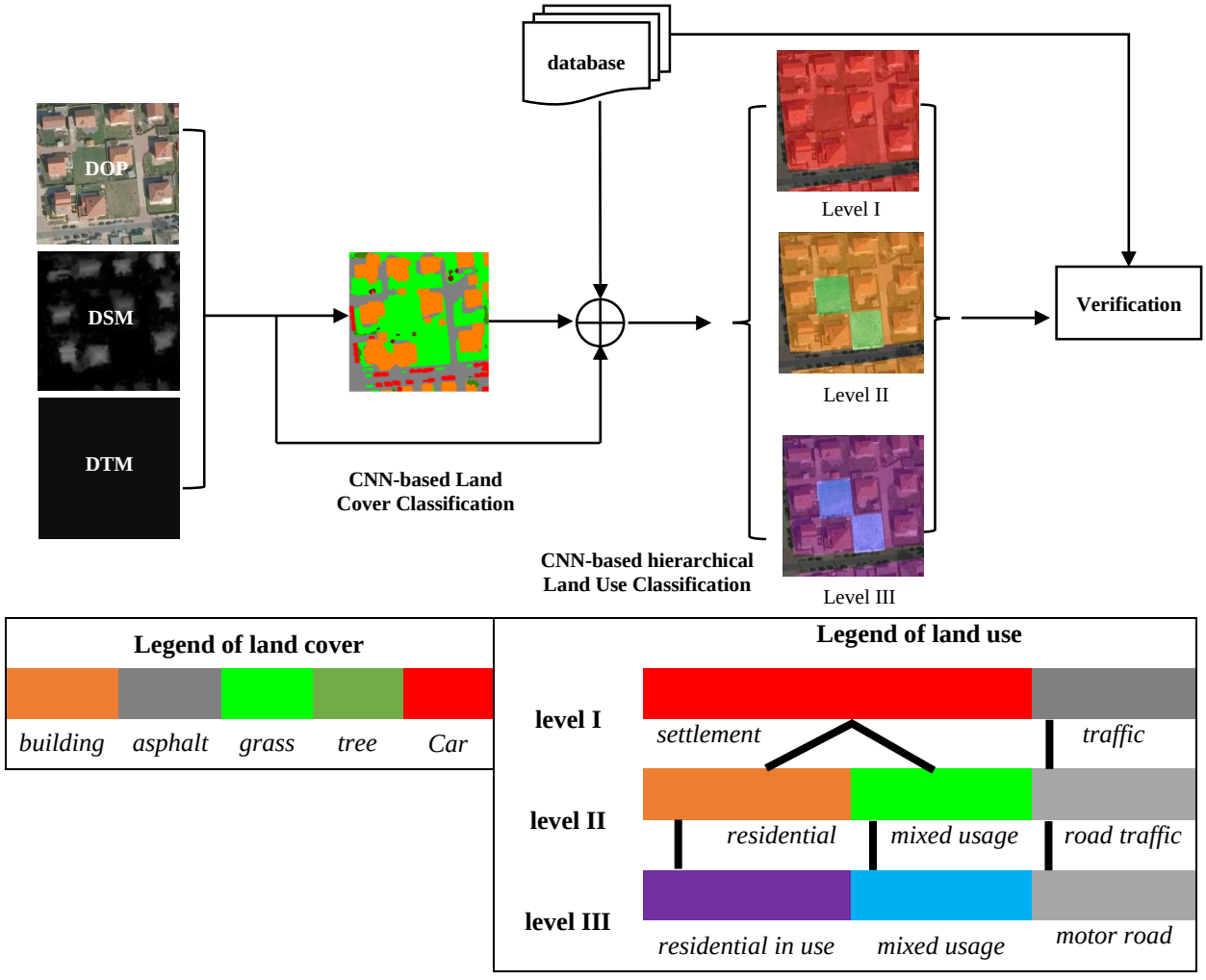


Figure 4.1: Illustration of the framework for the verification of a geospatial database in three hierarchical levels. In the legend of land use, black lines indicate the hierarchical relationships between categories. DOP: digital orthophoto; DSM: digital surface model; DTM: digital terrain model.

The CNN-based methods for both classification tasks are designed for input data consisting of high-resolution, multi-spectral aerial images, DSM and DTM. Furthermore, a geospatial database considering objects in the form of polygons is required for land use classification. The objects are associated with class labels encoding the land use according to a hierarchical object catalogue. To train a CNN, training data is also required for both classification tasks. The training data for hierarchical land use classification consists of objects with known land use information at multiple levels according to a hierarchical object catalogue (in this thesis the number is 3). To ensure the correctness, the information is assumed to be manually checked by experts. The training data for land cover classification consists of image tiles labeled at pixel level. The minimum geometric resolution of the input data should be 1 m or better, so that they are sufficient to detect objects such as *buildings*.

The core components of the workflow are the proposed CNNs for land cover and hierarchical land use classification. First, the input data is processed by a CNN to obtain pixel-wise land cover posteriors. Subsequently, the land cover posteriors along with the image, the DSM and the DTM serve as input for another CNN to predict land use of multiple hierarchical labels. Finally, the verification of the geospatial database is carried out based on the hierarchical classification results. However, the verification step is beyond the scope of this thesis, thus, it will not be discussed further. Section 4.2 presents the method for land cover classification, whereas the method for land use classification is presented in Section 4.3.

4.2. Land Cover Classification

The first component of the presented methods is a supervised classifier for land cover with the purpose of assigning each image pixel to a land cover category which describes the physical material of the earth's surface (e.g. *grass*, *asphalt*).

4.2.1. Network Architecture

The CNN developed in this thesis has a symmetric encoder-decoder structure and is referred to as *FuseNet-lite* (cf. Fig. 4.2). It requires two input images, each of size 256 x 256 pixels with three bands. One input is an RGB image whereas the other one is referred to as RID image, consisting of the Red and Infrared bands of the multi-spectral images and the height data (nDSM or DSM). The encoder delivers features which are extracted from the input images in two parallel branches, and those features are fused by 1 x 1 convolutions before decoding. In each encoder branch, there are four convolution blocks, each consisting of three convolutional layers followed by BN and a ReLU. At the end of each block, there is a max-pooling layer with window size of 2 x 2 and stride of 2. Symmetrically, the decoder part consists of four blocks, each starting with an upsampling layer that applies bilinear interpolation, followed by three convolutional layers, BN and a ReLU. The filter size of each convolution is 3 x 3. Optionally, at the end of each convolution block in the decoder part, there may be skip-connections (cf. Section 4.2.2 for network variants). It has to be noted that there is no skip-connection between the

outmost convolution blocks of encoder and decoder, because previous experiments have shown that these skip-connections are irrelevant (Yang et al., 2020a). The number of filters in each convolution block is shown underneath the block in Fig. 4.2. The total number of trainable parameters of *FuseNet-lite* is about 0.5 million, which is about 6% of the number of parameters of the network proposed in (Yang et al., 2020a). Experiments in (Yang et al., 2021) have shown that the reduction of parameters reduces overfitting to a certain degree while maintaining nearly the same level of classification accuracy. To predict the class labels at the end of the decoder in the resolution of the input image, a 1 x 1 convolutional layer converts the output of the previous layer to a vector of M_{LC} class scores for each of the $H \times W$ pixels of the input image, where M_{LC} denotes the number of classes to be differentiated. For each pixel *i* of the image, this results in a vector of $\boldsymbol{z_{LC}^{i}} = (z_{LC}^{i,1}, \ldots, z_{LC}^{i,M_{LC}})$ of class scores, where $\mathbb{C}_{LC} = \{C_{LC}^{1}, \ldots, C_{LC}^{M_{LC}}\}$ is the set of land cover classes and $z_{LC}^{i,c}$ is the class score for class C_{LC}^{c}. These class scores are normalized by a softmax function delivering the posterior probability $P_i(C_{LC}^{c}|X)$ for pixel *i* to take class label C_{LC}^{c} given the image data *X*:

$$P_i(C_{LC}^{c}|X) = softmax\left(\boldsymbol{z_{LC}^{i}}, C_{LC}^{c}\right) = \frac{\exp(z_{LC}^{i,c})}{\sum_{j=1}^{M_{LC}} \exp(z_{LC}^{i,j})} \tag{4.1}$$

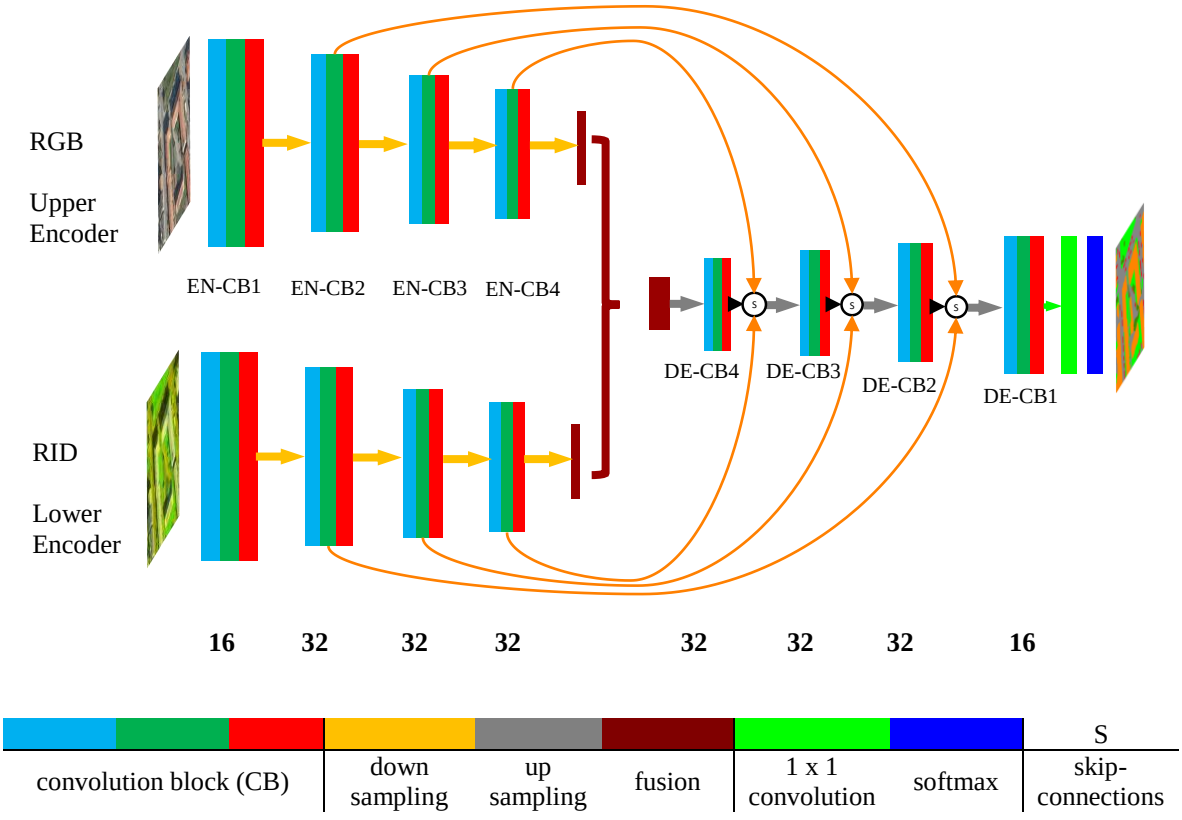


Figure 4.2: Architecture of *FuseNet-lite*. The number underneath each convolution block denotes the number of filters. EN-CB1, …, EN-CB4: Identifiers of convolution block in the encoder; DE-CB1, …, DE-CB4: Identifiers of convolution block in the decoder.

4.2.2. Network Variants

In the proposed architecture there are optional skip-connections between corresponding convolution blocks in the encoder and the decoder. Basically, there are three different options for connecting the encoder and the decoder, and consequently, there are three network variants.

4.2.2.1. Network without skip-connections

In the first scenario, no skip connections are used. This variant is referred to as *FuseNet-lite-NoSkip*, serving as a baseline for comparison to highlight the degree of the improvement caused by skip-connections in the task of land cover classification.

4.2.2.2. Network with elementwise addition skip-connections

In the second scenario, the most commonly applied skip-connections (Long et al., 2015) is used, the one based on elementwise addition of feature maps. The network variant based on this type of skip-connections is referred to as *FuseNet-lite-AddSkip*. Fig. 4.3 illustrates the procedure of this type of skip-connections. When linking encoder and decoder blocks at level *N*, the feature maps delivered by all three convolutional blocks of that level are considered. Suppose that there are $\mathcal{S}$ feature maps at each convolution layer of a convolution block at level N, thus, there are total of $3 \cdot \mathcal{S}$ feature maps at that convolution block (because each block has 3 convolution layers). Let $\boldsymbol{f}_{\boldsymbol{EN1}}^{j}, \boldsymbol{f}_{\boldsymbol{EN2}}^{j}, \boldsymbol{f}_{\boldsymbol{DE}}^{j}$ denote the three feature maps of the convolution block at the upper encoder, lower encoder and decoder of Fig. 4.2, respectively, with $j \in \mathcal{S}$. Thus, each of these feature maps has the dimension of $H \times W \times 3$. These feature maps are concatenated to form an intermediate feature map $\tilde{\boldsymbol{f}}^{j}$ with a dimension of $H \times W \times 9$, and the output feature map $\boldsymbol{h}^{j}$ with a dimension of $H \times W$ is determined according to eq. 4.2:

$$\boldsymbol{h}^{j} = \sum_{i=1}^{9} \tilde{\boldsymbol{f}}^{j,i} \tag{4.2}$$

4.2.2.3. Network with learnable skip-connections

In the third network variant, a new learnable way of integrating skip-connections is proposed (cf. Fig. 4.4). Instead of an elementwise addition of feature maps, a 3 x 3 depth-wise convolution is applied to every feature map.

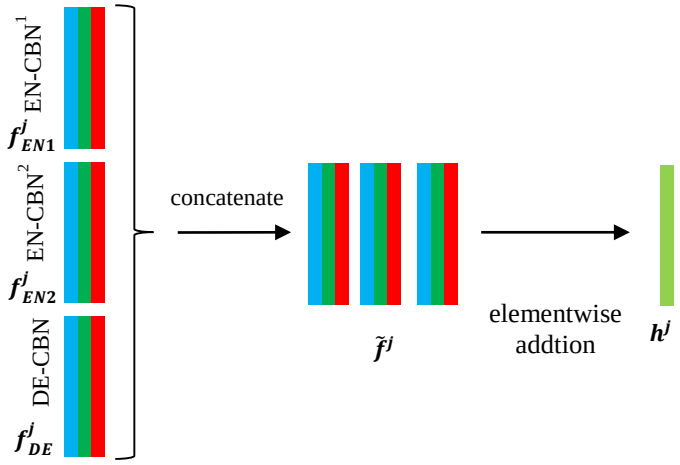


Figure 4.3: Skip-connections based on elementwise addition. EN-CBN[1], EN-CBN[2]: feature maps delivered by all convolutional blocks of the encoder blocks *N* of the upper and lower encoder branches in Fig. 4.2, respectively. DE-CBN: feature maps of all convolutional blocks of the decoder block N in Fig. 4.2. Refer to the main texts for the meanings of variables.

$$\boldsymbol{v}^{\boldsymbol{j}}_{\boldsymbol{idx}} = ReLU(\omega_{j,idx} * \boldsymbol{f}^{\boldsymbol{j}}_{\boldsymbol{idx}} + b_{j,idx}) \tag{4.3}$$

In eq. 4.3, $\boldsymbol{v}^{\boldsymbol{j}}_{\boldsymbol{idx}}$ is an intermediate feature map where the subscript *idx* can be *DE* (decoder), *EN1*(upper encoder) and *EN2* (lower encoder). The filter matrix $\omega_{j,idx}$ and the bias $b_{j,idx}$ are the parameters to be learnt, and the symbol * represents convolution. Note that these convolutions are only applied if a skip-connection is established. $\boldsymbol{v}^{\boldsymbol{j}}_{\boldsymbol{idx}}$ has the same dimension as $\boldsymbol{f}^{\boldsymbol{j}}_{\boldsymbol{idx}}$ i.e. $H \times W \times 3$. Let $\boldsymbol{v}_{\boldsymbol{idx}}$ denote all feature maps for subscript ***idx*** at block level N, thus, it has a dimension of $H \times W \times 3 \cdot \mathcal{S}$. Subsequently, the feature maps $\boldsymbol{v}_{\boldsymbol{EN1}}, \boldsymbol{v}_{\boldsymbol{EN2}}, \boldsymbol{v}_{\boldsymbol{DE}}$ are concatenated to form a tensor $\boldsymbol{u}$ whose dimension is $H \times W \times 9 \cdot \mathcal{S}$. After concatenation, a set of *D* 1 x 1 convolutions is used to deliver *D* output feature maps. Let $\boldsymbol{g}^{\boldsymbol{d}}$ denote the d^{th} output feature map. Denoting the elements at position (*i*, *k*) of feature maps $\boldsymbol{u}^{\boldsymbol{m}}$ and $\boldsymbol{g}^{\boldsymbol{d}}$ by $u^m_{i,k}$ and $g^d_{i,k}$, respectively, the elements of the combined feature map are computed according to:

$$g^d_{i,k} = ReLU\left(\sum_{m=1}^{9\cdot S} \theta^m_d \cdot u^m_{i,k} + b_d\right) \tag{4.4}$$

where θ^m_d and b_d are parameters to be learnt. The output feature maps form the input to the first convolutional layer of the next decoder block in Fig. 4.2. This is the standard configuration of the network defined in Section 4.2.1. Thus, it is referred to as *FuseNet-lite*.

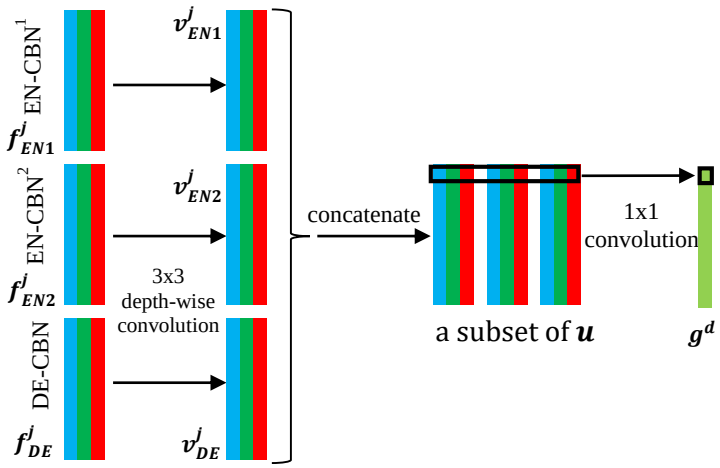


Figure 4.4: Learnable skip-connections. Refer to the main texts for the meanings of variables and to Fig. 4.3 for the subscripts.

4.2.3. Training

All parameters of the convolutional layers are learned during training, which relies on SGD and step leaning policy. The weight decay regularizer in eq. 2.10 is also used. The imbalanced class distribution poses a problem to the training. Consequently, the unknown parameters better fit to the well-represented classes and a bad performance on the underrepresented classes is to be expeted. To tackle that problem, an extended focal loss for multi-class classification on the basis of (Lin et al., 2017) is proposed:

$$L_{LC} = -\frac{1}{W \cdot H \cdot \Omega} \sum_{c,i,k} \left[y_c^{ik} \cdot \left(1 - P_i(C_{LC}^c | X_k)\right)^{\gamma} \cdot log\left(P_i(C_{LC}^c | X_k)\right) \right], \qquad (4.5)$$

where k is the index of an image, X_k is the k^{th} image in the mini-batch and Ω is the number of images in a mini-batch. The indicator variable y_c^{ik} is 1 if the training label of pixel i in image k is identical to C_{LC}^c and 0 otherwise, and the hyper-parameter γ, typically lying in the range of 1.0 to 5.0, controls the penalty term $(1 - P_i(C_{LC}^c | X_k))$. This term suppresses the influence of well-classified pixels (for which $(1 - P_i(C_{LC}^c | X_k))$ is close to zero), whereas it causes the loss to focus on pixels for which $P_i(C_{LC}^c | X_k)$ of the correct class is small. Using this loss, it is possible to mitigate the problem of imbalanced class distributions, because classes for which a large number of samples is available are expected to be mostly well classified during training, so that this loss will put a higher emphasis on the samples of underrepresented classes.

4.3. Hierarchical Land Use Classification

The second core component of the methodology is the land use classification. Its purpose is to assign each land use object multiple labels at different semantic levels, and these labels have a hierarchical relationship according to a hierarchical object catalogue. In the database, each land use object is represented by a polygon. One big challenge in classifying polygons is their large variation in terms of geometrical extent; for instance, *road* objects are thin and long, whereas *settlement* objects may cover both, very large and quite small areas (see Fig. 4.5). These variations pose problems for a CNN, as it requires regular-sized patches as input. Thus, for classifying polygons by a CNN, a way for preparing such regular patches has to be implemented.

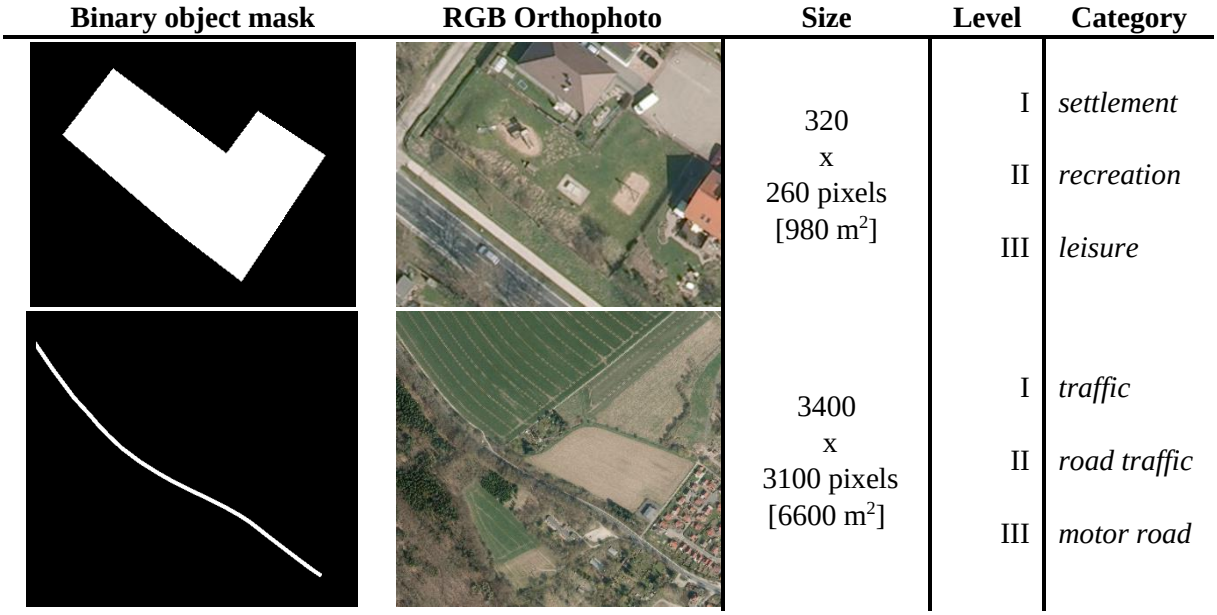


Figure 4.5: Two database objects with corresponding RGB orthophotos and categories in three semantic layers of (AdV, 2008), starting from the coarsest (I) to the finest one (III). The object shape is represented by a binary object mask at the same scale as the corresponding orthophoto. Note that the two objects are displayed at different scales. The ground sampling distance (GSD) of the orthophotos is 20 cm. The area in [m^2] refers to the area of the database object.

The polygons and their corresponding data (i.e. image data and land cover information delivered by *FuseNet-lite*) are used to prepare regular patches with a size of 256 x 256 pixels. These patches are used to train a CNN classifier. In the inference step, each patch belonging to an unseen polygon is predicted by that CNN. The class labels of that polygon are determined by a combination of the predictions for its patches.

Section 4.3.1 presents the representation of polygon shape, whereas Section 4.3.2 describes the implemented methods for preparing input patches. The network variants and the corresponding loss

functions for training are presented in Section 4.3.3 and Section 4.3.4, respectively. Finally, Section 4.3.5 describes the inference procedure at object level.

4.3.1. Polygon Shape Representation

Polygons are stored in vector format in the geospatial database. In order to consider the shape in a CNN, a raster representation of the resolution having the same GSD as the input images has to be prepared. In this thesis, two ways of generating this representation are compared. The first one consists of a binary mask in which a value of 255 indicates that a pixel belongs to the object, whereas pixels outside of the polygon are set to 0 (see Fig. 4.5). This type of representation is referred to as *polygon mask*. If this method is used, the binary mask is an additional input presented to the CNN. The second representation extracts the image data and land cover information and replaces the area outside the polygon by a predefined background color or pattern. In this case, no additional input is used by the CNN. In this way, the polygon shape is represented by the transition between image data and the background. The predefined background can be any color or pattern which has a contrast to the image data. This type of representation is referred to as *implicit polygon representation*. Compared to a polygon mask, the implicit representation loses the context information surrounding an object, which may affect the performance in a negative way.

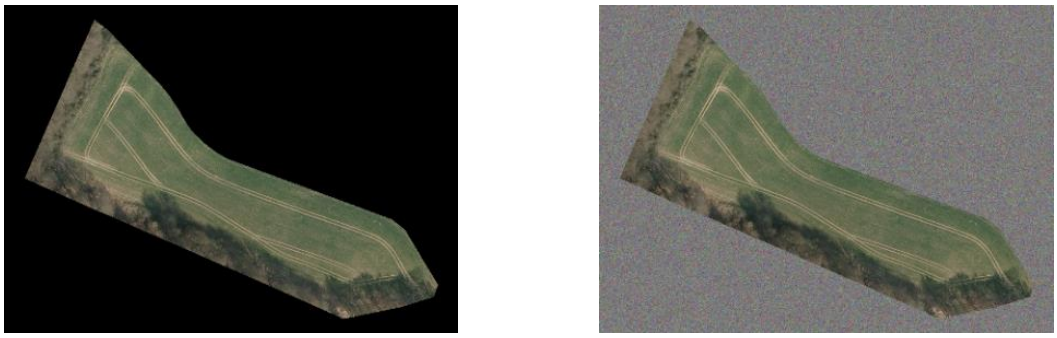


Figure 4.6: A land use object in implicit representation with black (left) and Gaussian background (right). Inside the polygon, only RGB data is shown.

With respect to implicit polygon representation, two kinds of background are compared in this thesis (cf. Fig. 4.6):

1). Black, i.e. the grey values of all bands are set to 0;

2). Gaussian noise. In this case, the grey value of each pixel in each band is $128 + v$ where $v \sim N(0, \sigma^2)$ is drawn from a Gaussian distribution with expectation of 0 and variance of σ^2. In this thesis, $\sigma = 27$ has been selected

For both representations, the total number of input bands can be formulated as $N_{input} = N_{shape} + N_{DOP} + N_{3D} + N_{LC}$. If the representation is a polygon mask, $N_{shape} = 1$, whereas $N_{shape} = 0$ for the case of implicit polygon representation. The number of bands of the DOP is $N_{DOP} = 4$ for multi-spectral images having RGB and NIR bands. $N_{3D} = 1$ if there is height data (DSM or nDSM). N_{LC} corresponds to the number of land cover classes, which is dataset dependent.

4.3.2. Patch Preparation

Independently from the shape representation of a polygon, two strategies are proposed for patch preparation, i.e. for the creation of regular-sized patches for the CNN. The first strategy, *tiling*, preserves the original image resolution, which implies that large polygons have to be split into smaller patches. The second strategy, *scaling*, rescales the polygons so that the they fit into the input window of the CNN. Furthermore, the patches generated from both strategies can be combined, which is referred to as *combination of tiling and scaling*.

4.3.2.1. Tiling

Using the tiling strategy, the first step is to check whether a polygon fits into a window of 256 x 256 pixels at the resolution of the original data. If this is the case, such a window is placed over the polygon so that the polygon center coincides with the center of the window, resulting in one tile. Such polygons are referred to as *small* polygons in this thesis. In contrast to *small* polygons, a *large* polygon does not fit into a single patch. Such polygons are split into a series of tiles of 256 x 256 pixels with an overlap of 50% in both directions.

For each tile, the proportion of its area that is inside the database object is checked. If this proportion is smaller than a threshold (set to 10% of the area of this tile), the tile will be excluded. This may still lead to a large number of tiles for *large* polygons. If more than N_{min} tiles remain after this procedure, 40% of them are randomly selected for being classified by the CNN to reduce the computational burden. $N_{min} = 3$ is applied in all experiments. Fig. 4.7 shows a tiling example of a very large road objects.

4.3.2.2. Scaling

The tiles created by the approach in Section 4.3.2.1 preserve the original resolution of the images. However, one problem is that the shape information of large polygons is lost. To solve this problem in the scaling approach, all polygons are rescaled to the CNN input size to preserve the shape information. In addition, there is another problem for small polygons if tiling is used for patch generation. If a small polygon covers only a small portion (e.g. 10%) of the area of a patch, the CNN input will be dominated

by the surroundings, possibly leading to misclassification. To solve this problem, in the scaling approach small polygons are scaled to the CNN input size using multiple scales, and each time different a context window is extracted.

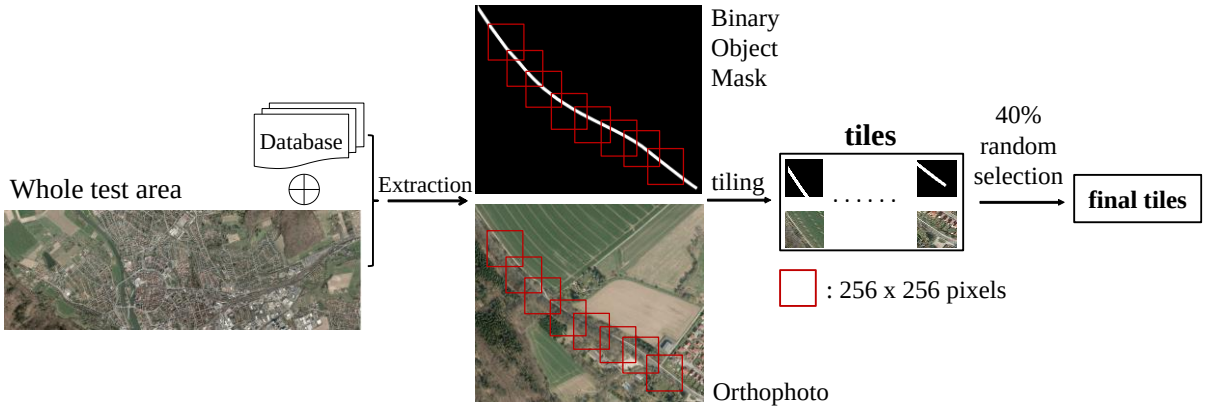


Figure 4.7: Example for *tiling* a large road object in one of the test datasets. Only the binary object mask representing the object shape and the RGB orthophoto are shown here to explain the principle, but the extracted tiles would contain all data mentioned in the main text.

If the scaling approach is used for small polygons, a scale $\alpha_0 = 1$ is defined so that the original input at the GSD of input image is used. Furthermore, the input data are scaled such that the object fits exactly into a window of the input size of the CNN, using the scale:

$$\alpha_1 = \frac{256}{\max(W,H)} \tag{4.6}$$

On the basis of α_1, other scales α_k are computed for $k \in \{2, 3, 4, 5\}$:

$$\alpha_k = \frac{1}{2^{k-1}} \cdot \alpha_1 \tag{4.7}$$

The minimum scale that can be considered is $\frac{1}{16} \cdot \alpha_1$. However, scales $\alpha_k < 1$ are not used. Thus, the number of scales applied for an object depends on the object size. For each scale α_k, a window centered at the object center is extracted from the input data and up-scaled to 256 x 256 pixels. In the upscaling process, the binary mask images are interpolated via nearest neighbor interpolation; for all other data, bilinear interpolation is applied. Fig. 4.8 shows an example of a small polygon with patches at multiple scales.

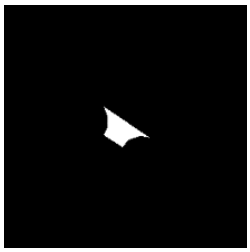
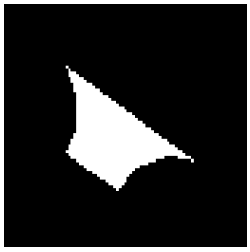
Binary object mask	RGB Orthophoto	Note
		original input with $\alpha_0 = 1$
		$\alpha_1 \approx 5.6$
		$\alpha_2 \approx 2.8$
		$\alpha_3 \approx 1.4$

Figure 4.8: An example for the patch generation for a small polygon. All images have a size of 256 x 256 pixels. The size of the polygon is about 45 x 45 pixels at the original resolution of the images (about 81 m^2 in area at a GSD of 20 cm).

4.3.2.3. Combination of tiling and scaling

The patches generated by the tiling and scaling approaches are supposed to be complementary for the hierarchical land use classification. Two strategies for their combination are proposed in this thesis. The first one is a combination at the prediction stage. In this case, two CNNs are trained, i.e. one for each patch generation method, and their predictions are combined via an ensemble method which will be explained in Section 4.3.5. The second way is to combine all patches generated by tiling and patches of small polygons. These combined set of patches is used to train the; this approach is referred to as

COM-TS. In both combination strategies, one common main purpose is to investigate whether the multi-scaling approach could improve the classification of small polygons.

4.3.3. Network Architecture

In this section, the networks used for land use classification are introduced. Firstly, the base network *LuNet-lite* is introduced, which was developed for the purpose of classification at one semantic level. Subsequently, a multi-task learning network based on *LuNet-lite* is presented, referred to as *LuNet-lite-MT*, which classifies one input patch into categories at multiple semantic levels. Finally, on the basis of *LuNet-lite-MT*, the approach for achieving consistency with the class hierarchy is proposed.

4.3.3.1. Base Network for Mask Representation: *LuNet-lite*

LuNet-lite is based on VGG-16, because the latter delivers promising results in most vision tasks. However, previous experiments have shown that reducing the number of parameters to a certain degree reduces overfitting, but can still maintain classification accuracy (Yang et al., 2021). Compared to about 53 million of trainable parameters in VGG-16, *LuNet-lite* only requires about 1 million trainable parameters.

LuNet-lite predicts one land use label for an input patch generated in the way described in Section 4.3.2. It consists of four main convolutional blocks before splitting into two branches, the outputs of which are combined at the end (cf. Fig. 4.9). Each of the main convolutional blocks consists of three convolution layers followed by BN and ReLU, and a pooling layer with window of 2 x2 and stride of 2 is appended to the last convolution layer. For each convolution operation, zero padding is used to keep the spatial resolution of the output identical to the one of the input. The upper branch starts with a max pooling layer, the aim of which is to extract features that are representative for the entire input image. Due to the variations of the size and position of the polygons inside the images, a second branch is proposed that just uses a region of interest (ROI) of the input image, i.e. a rectangle aligned with the image grid that tightly encloses the polygon, thus focusing on the most relevant regions in the image. As the size of the ROI varies, the output of the last common convolutional block inside the ROI is resized to a fixed size of 16 x 16 by bilinear interpolation (cf. Fig. 4.10). At the end of each branch, the spatial resolution of the feature maps is 8 x 8. An average pooling layer with a window of 8 x 8 is used to obtain a 64-dimentional feature vector. Finally, the feature vectors of both branches are concatenated to obtain a 128-dimensional feature vector, denoted as $\boldsymbol{f^{LU}}$, which is the input to the final FC layer that delivers a vector of class scores $\boldsymbol{z_{LU}} = \left(z_{LU^1}, \dots, z_{LU^M} \right)^T$, where $C_{LU^c} \in \mathbb{C}_{LU} = \{C_{LU^1}, \dots, C_{LU^M}\}$ is one of *M* land use classes and z_{LU^c} is the class score of an image in a mini-batch *X* for class C_{LU^c}. To obtain a probabilistic class score, the softmax function (eq. 4.8) is applied to the final vector of class scores:

$$P(C_{LU^c}|\mathrm{X}) = softmax(\boldsymbol{z_{LU}}, C_{LU^c}) = \frac{exp(z_{LU^c})}{\sum_{i=1}^{M} exp(z_{LU^i})} \tag{4.8}$$

4.3.3.2. *LuNet-lite* with Multi-Task Learning

On the basis of *LuNet-Lite* another network architecture, referred to as *LuNet-lite-MT,* is introduced. This network variant is able to classify an input object at multiple semantic levels.

Fig. 4.11 presents the extension of *LuNet-lite* required to obtain the multi-task learning network *LuNet-lite-MT* for $B = 3$ semantic levels. The 128-dimensional feature vector $\boldsymbol{f^{LU}}$ generated by the base network *LuNet-Lite* is passed on to $B = 3$ classification heads, each delivering a feature vector having a dimension corresponding to the number of categories to be discerned in that level. At category level l, a feature vector $\boldsymbol{z_{LU}^{l}} = \left(z_{LU,1}^{l}, \dots , z_{LU,M_l}^{l} \right)^T$ for the M_l classes $C_{LU,c}^{l} \in \{C_{LU,1}^{l}, \dots, C_{LU,M_l}^{l}\}$ of the set of land use classes is obtained via a fully connected layer processing the vector $\boldsymbol{f^{LU}}$. These vectors $\boldsymbol{z_{LU}^{l}}$ of all semantic levels form the input for the multi-task classification head, which applies a specific information flow for learning the semantic relationships of the categories of different levels implicitly. In the first layer, information flows from the coarser to the finer levels, whereas in the second layer, the information flow is reversed. The scheme of the information flow is inspired by the work of (Hu et al., 2016), though embedded in a completely different context and expanding the degree of interaction between the individual semantic levels.

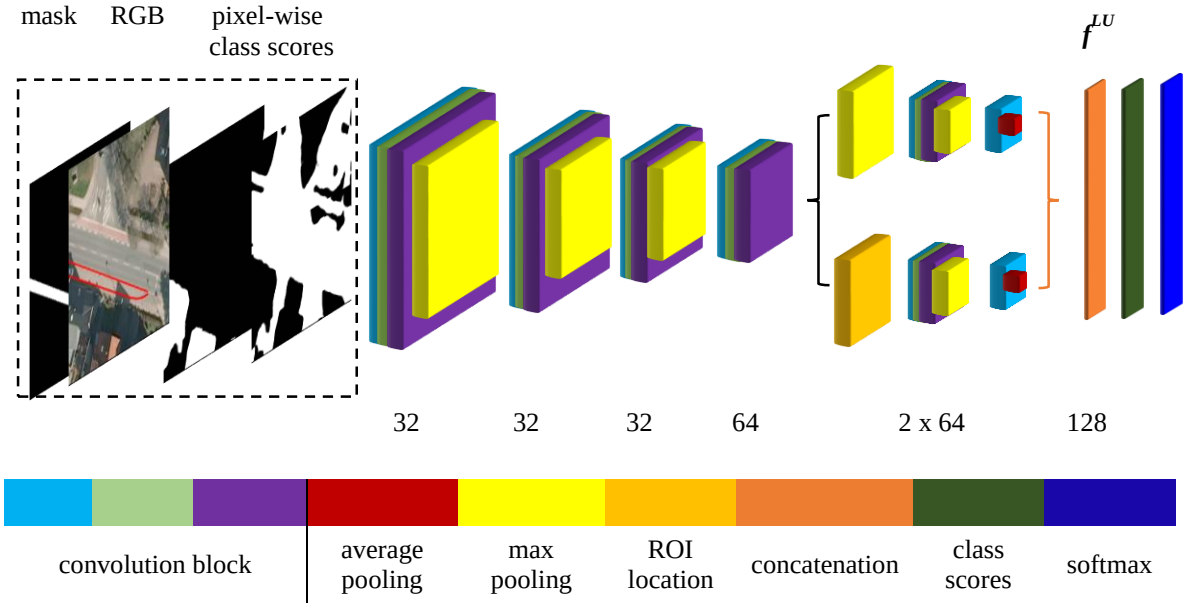


Figure 4.9: The architecture of *LuNet-lite*. The input data of object mask, RGB and land cover posteriors is only an example, as it can vary between different network variants. The number of kernels is shown underneath each convolution block.

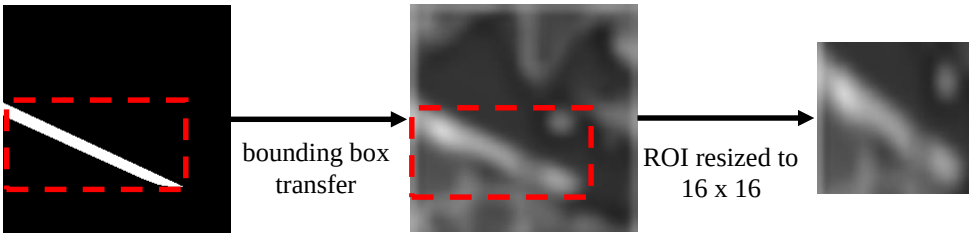


Figure 4.10: ROI location (red rectangle).

The first layer of the classification head determines intermediate vectors $\boldsymbol{z}_{LU}^{mid,l}$ at each level l. For $l > 1$, they are based on the output of the previous network layer for all coarser levels and for the same one; otherwise ($l = 1$), the scores are just copied, as presented in eq. 4.9.

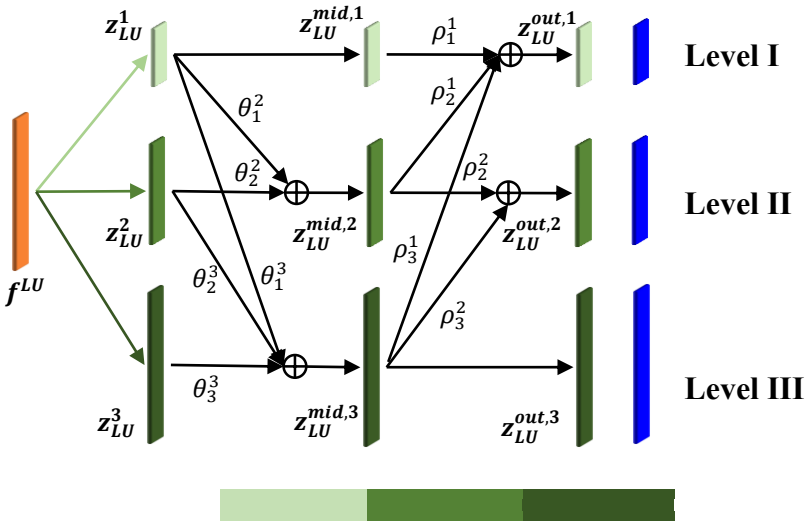


Figure 4.11: Extension of the architecture of *LuNet-lite* to obtain *LuNet-lite-MT* for $B = 3$ semantic levels (from I/coarsest to III/finest). $\boldsymbol{f}^{LU}$ is the feature vector determined by *LuNet-lite* (cf. Fig. 4.9). For the explanation of the mathematical symbols, please refer to the main text.

$$\boldsymbol{z}_{LU}^{mid,l} = \begin{cases} \sum_{i=1}^{l} \theta_i^l \cdot ReLU(\boldsymbol{z}_{LU}^{i}), & if\ l > 1 \\ \boldsymbol{z}_{LU}^{l}, & if\ l = 1 \end{cases} \tag{4.9}$$

In eq. 4.9, θ_i^l are the trainable parameters to be learned for that layer. The second layer of the classification head determines the final vectors $\boldsymbol{z}_{LU}^{out,l}$ for every level l. Whereas for the finest level (l =

B), these scores are just copies of the intermediate ones, the others (i.e. $l < B$) receive input from all finer levels and from the same one:

$$\boldsymbol{z}_{\boldsymbol{LU}}^{\boldsymbol{out,l}} = \begin{cases} \sum_{j=l}^{B} \rho_j^l \cdot ReLU(\boldsymbol{z}_{\boldsymbol{LU}}^{\boldsymbol{mid,j}}), & if\ l < B \\ \boldsymbol{z}_{\boldsymbol{LU}}^{\boldsymbol{mid,l}}, & if\ l = B \end{cases} \tag{4.10}$$

In eq. 4.10, ρ_j^l are the trainable parameters of that layer. As it is assumed that the vectors at each level are related to those at other levels due to the hierarchical relationship of the labels. By determining the parameters of these FC layers, the CNN is expected to learn the hierarchical semantic relations between the semantic levels to a certain degree. However, the consistency of the result with the class hierarchy is not guaranteed. Finally, the vectors $\boldsymbol{z}_{\boldsymbol{LU}}^{\boldsymbol{out,l}}$ of eq. 4.10 are normalized by a softmax function to obtain values that can be interpreted as posteriors. The probabilistic class scores of class $C_{LU,c}^l$ at level l is:

$$P\left(C_{LU,c}^l \middle| \mathrm{X}\right) = softmax\left(\boldsymbol{z}_{\boldsymbol{LU}}^{\boldsymbol{out,l}}, C_{LU,c}^l\right) = \frac{exp(z_{LU,c}^{out,l})}{\sum_{m=1}^{M_l} exp(z_{LU,m}^{out,l})} \tag{4.11}$$

4.3.3.3. Achieving Consistency with the Class Hierarchy

One problem of *LuNet-lite-MT* is that it does not guarantee consistency of the predictions with the class hierarchy. In this thesis, two approaches to achieve this consistency are proposed. The first one considers the constraints imposed on the class labels by the hierarchical object catalogue during training and inference on the basis of *LuNet-lite-MT*, whereas the second one applies a post-processing step to the predictions at the finest level delivered by *LuNet-lite*. In this section, both network variants are presented.

In the first scenario, *LuNet-lite-MT* delivers predictions of multiple class labels corresponding to different semantic levels simultaneously while considering semantic dependencies as they were learned from the training samples. However, there is no guarantee for the predictions to be consistent with the object catalogue hierarchy. To achieve consistency, a so-called *joint optimization* (JO) strategy is proposed to obtain the most probable set of class labels that is consistent with the class hierarchy based on the output of the multi-task head. The network employing the JO strategy on the basis of *LuNet-lite-MT* is referred to as *LuNet-lite-JO* (the abbreviation MT is omitted for the sake of simplicity).

Using the *JO* strategy, the combined class scores for all possible combinations of classes at different semantic levels is computed. Given the hierarchical class structure, a set of class labels that is consistent with that class structure is a *B*-tuple of class labels (a triple in the case of this thesis with $B = 3$) that corresponds to all labels along a path through the tree representing the class structure for one of the classes of level III (cf. the dark red path in Fig. 4.12). There are M_B valid tuples $T_i = \{C_{LU,i}^1, C_{LU,i}^2 C_{LU,i}^3\}$, one per class label differentiated at the finest level *B* of the hierarchy, where $C_{LU,i}^l$ is the class label of

the i^{th} tuple at level l. For each input image X_k and each tuple T_i, the joint class scores for the image to correspond to that tuple is computed according to:

$$P_{joint}^{i,k}(T_i, X_k) = \prod_{l=1}^{B} P\left(C_{LU,i}^{l} | X_k\right) \tag{4.12}$$

where $P\left(C_{LU,i}^{l} | X_k\right)$ is the output of the softmax function (eq. 4.11) for class $C_{LU,i}^{l}$ at the semantic level l. The classification output for an image X_k is the tuple T_i that maximises the joint class scores according to eq. 4.12. The JO strategy is illustrated in Fig. 4.12.

In the second scenario, a post-processing strategy called *fine-to-coarse* (F2C; cf. the blue path in Fig. 4.12) is applied to the predictions at the finest level B delivered by *LuNet-lite*: starting from the prediction at the finest level, the ancestor classes of that prediction at coarser levels are the corresponding predictions for the input patch. The F2C strategy also guarantees the consistency with class hierarchy of the final prediction. This variant is referred to as *LuNet-lite-F2C*, which is compared to the proposed JO strategy in the experiments.

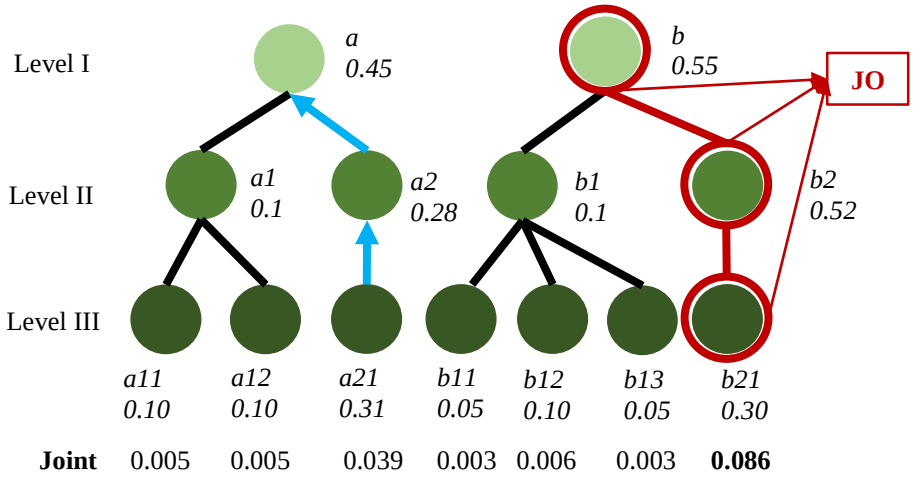


Figure 4.12: Illustration of the JO strategy and the F2C strategy using an example in which labels are predicted at three semantic levels. The lines indicate hierarchical relations between classes. Numbers underneath the class names give the posterior probabilities. Strategy JO selects the tuple of class labels consistent with the hierarchy that has the highest joint class score (dark red). *a, b*: labels at level I; *a1, …, b2*: labels at level II; *a11, …, b21*: labels at level III. Strategy F2C uses the predictions at the finest level to control the predictions at coarser levels (blue). Joint: joint class scores

4.3.3.4. Network Architecture for Implicit Representation

So far, all network variants rely on the ROI branch, which is supposed to extract helpful information in a relevant but small context. For implicit polygon representation, the area surrounding the polygon

does not provide useful context information. Consequently, the ROI branch, expanded to focus on the relevant area of a polygon, is not relevant. Fig. 4.13 shows the differences of the context region between the mask and the implicit polygon representations for a settlement object.

For patches based on implicit polygon representation, the ROI branch of all network variants is thus removed (cf. Fig. 4.14). As the purpose of using implicit polygon representation is to investigate how the representation of the polygons affects the classification, and because the main contribution in this thesis is the proposed JO strategy for hierarchical land use classification, only the network *LuNet-lite-JO* is adapted by removing the ROI branch, resulting in the network variant *LuNet-lite-BG-JO*. It has to be noted that the feature vector $\boldsymbol{f^{LU}}$ forming the input for classification head (cf. Fig. 4.11) is 64-dimensional instead of 128- dimensional in this variant.

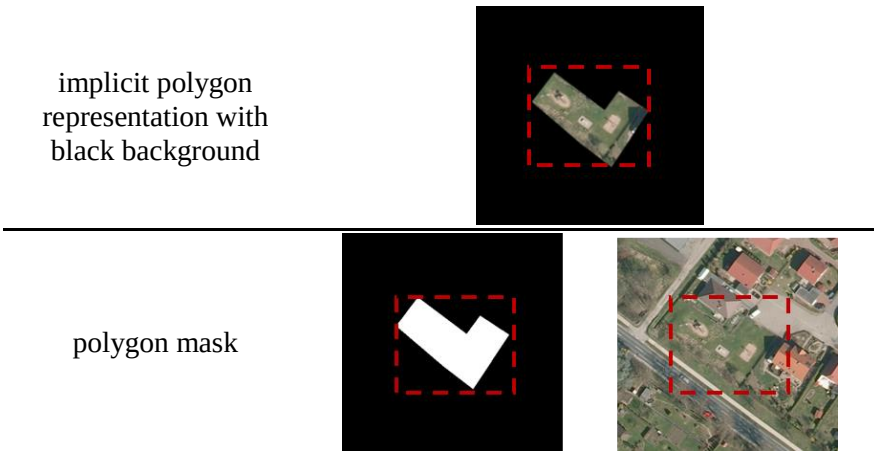


Figure 4.13: Difference between implicit polygon representation with black background (top) and polygon mask (bottom). Red rectangle: extraction of the ROI.

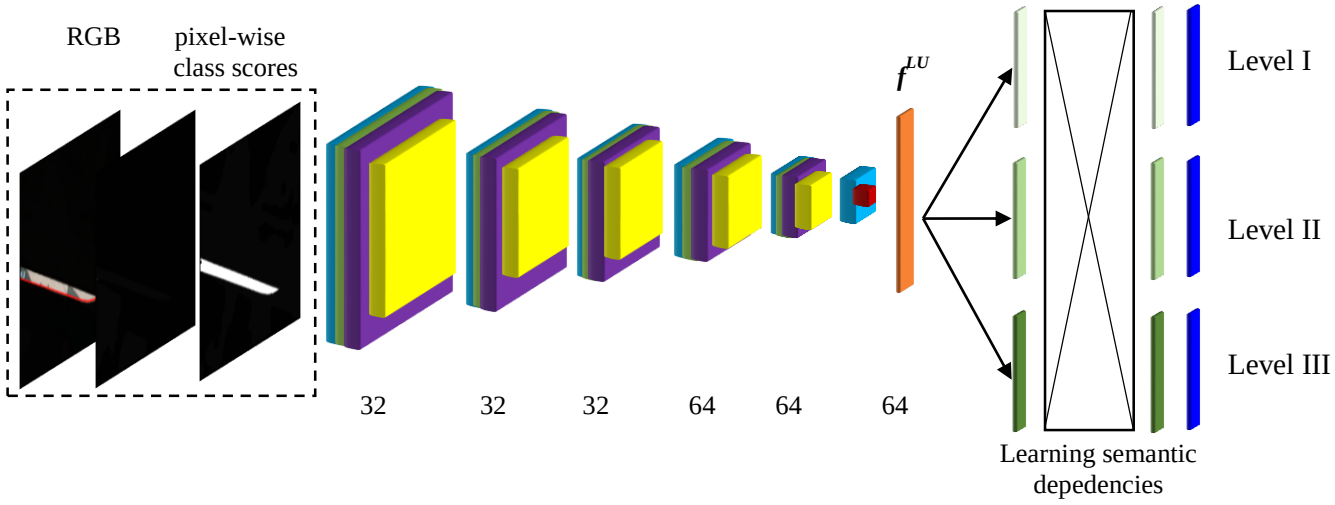


Figure 4.14: Architecture of *LuNet-lite-BG-JO*. The interpretation of the colours is the same as the one presented in Fig. 4.9. The block for learning semantic dependencies has the same configuration as the one presented in Fig. 4.11.

4.3.4. Training

In this section, the loss functions used for network training are presented. All trainable parameters (the weight matrices of all convolution kernels and the corresponding biases, the weights from FC layers) are learned from training data. Training of all network variants relies on SGD and step learning policy, and weight decay is used for regularizing trainable parameters (cf. eq. 2.10).

4.3.4.1. *LuNet-lite*

For this network, the loss function to be optimized during training is the extended focal loss (eq. 4.13), because there is the problem of imbalanced class distribution. The extended focal loss is supposed to mitigate that problem to a certain degree, similarly to the loss function proposed for *FuseNet-lite.*

$$L_{LU} = -\frac{1}{\Omega} \cdot \sum_{c,k} y_{LU^c}^{k} \cdot \left(1 - P(C_{LU^c}|X_k)\right)^{\epsilon} \cdot \log(P(C_{LU^c}|X_k)) \tag{4.13}$$

In eq. 4.13, X_k is the k^{th} image in a mini-batch, Ω is the number of images in a mini-batch, and $y_{LU^c}^{k}$ is 1 if the training label of X_k is C_{LU^c} and 0 otherwise. The probabilistic class score $P(C_{LU^c}|X_k)$ is determined according to eq. 4.8. The parameter ϵ is the hyper-parameter to control the influence of the penalty term (corresponding to γ in eq. 4.5).

4.3.4.2. *LuNet-lite-MT*

The loss function to train this multi-task learning network is an expansion of the extended focal loss in eq. 4.13. In this network variant, the class labels at all semantic levels are considered to be independent from each other, thus, the loss is the sum of the losses at each semantic level based on eq. 4.13.

$$L_{LU}^{MT} = -\frac{1}{\Omega} \cdot \sum_{l,c,k} \left[y_c^{l,k} \cdot (1 - P\left(C_{LU,c}^{l}|X_k\right))^{\epsilon} \cdot log\left(P\left(C_{LU,c}^{l}|X_k\right)\right)\right] \tag{4.14}$$

In eq. 4.14, $y_c^{l,k}$ is 1 if the training label of X_k is $C_{LU,c}^{l}$ at level *l* and 0 otherwise. The probabilistic class score $P\left(C_{LU,c}^{l}|X_k\right)$ is determined according to eq. 4.11. X_k and Ω have the same meaning as in eq. 4.13.

4.3.4.3. *LuNet-lite-JO* and *LuNet-lite-BG-JO*

For these two network variants, the constraints imposed on the class labels by the hierarchical object catalogue are considered during training. To achieve this goal, a novel loss function L_{LU}^{JO} consisting of two components $L_{LU}^{JO,P1}$ and $L_{LU}^{JO,P2}$ is proposed:

$$L_{LU}^{JO} = L_{LU}^{JO,P1} + L_{LU}^{JO,P2} \tag{4.15}$$

The joint probabilities $P_{joint}^{i,k}(T_i, X_k)$ (eq. 4.12) of the hierarchical tuples matching the ground truth are maximized in training, which is considered by the first term in eq. 4.15.

$$L_{LU}^{JO,P1} = -\frac{1}{\Omega} \cdot \sum_k \sum_i^{M_B} y_i^{B,k} \cdot (1 - P_{joint}^{i,k}(T_i, X_k))^{\epsilon} \cdot \log\left(P_{joint}^{i,k}(T_i, X_k)\right) \tag{4.16}$$

At the same time, the joint probabilities of the incorrect tuples are minimized, which leads to the second loss function term:

$$L_{LU}^{JO,P2} = -\frac{1}{\Omega} \cdot \sum_k \sum_i^{M_B} (1 - y_i^{B,k}) \cdot (P_{joint}^{i,k}(T_i, X_k))^{\epsilon} \cdot \log(1 - P_{joint}^{i,k}(T_i, X_k)) \tag{4.17}$$

In eqs. 4.16 and 4.17, $y_i^{B,k}$ is 1 if tuple *i* is the ground truth label of image X_k at the finest level *B* and 0 otherwise. Thus, it identifies the correct tuple T_i. In addition, the focal-loss-style penalty term used in eq. 4.13 is also introduced in both eqs. 4.16 and 4.17, with the goal of mitigating the problem of imbalanced class distribution to a certain degree.

4.3.5. Inference at Object Level

All mentioned network variants are used to predict class labels for an input image patch of 256 x 256 pixels. For a polygon that has exact one patch, the prediction of its patch directly delivers the prediction of that object. This case corresponds to small polygons that fit the patch size if the tiling approach is used to generate patches or to large polygons if the scaling approach is applied for that purpose. Otherwise, the polygon has multiple patches, which is the case for large polygons if the tiling approach is used, for small polygons if the scaling approach is used for patch generation and for all polygons if the combination of tiling and scaling approach is applied. For this case, the inference at object level differs between the network variants:

- Variant *LuNet-lite-F2C*: the product of the class scores at the finest level of all patches belonging to a polygon is computed and used to select the most probable class for the compound object for

level *B*. After the prediction at the finest level *B* is determined, the F2C strategy is applied to obtain the predictions at coarser levels for the compound object.

- Variant *LuNet-lite-MT*: the product of the class scores from all patches belonging to a polygon is computed and used to select the most probable class for the compound object at each semantic level independently. It has to be noted that the final prediction does not guarantee the consistency with the class hierarchy.
- Variant *LuNet-lite-JO* and *LuNet-lite-BG-JO*: First, the products of the class scores of the individual patches are computed for all semantic levels. This results in a vector of combined probabilistic class scores for every class at every semantic level. Afterwards, the joint scores of all consistent tuples of class scores (eq. 4.12) are computed by using the combined class scores, and the tuple that maximises this joint class score is selected as the prediction for the compound object.
- Ensemble: some variants of the method proposed in this thesis are based on the combination of the outputs of an ensemble of different network variants. Given the class scores delivered by these network variants for a polygon, the product of the class scores at each semantic level is computed. Subsequently, the joint scores of all consistent tuples are computed and the tuple having the maximum is selected as the prediction of that polygon.

5. Datasets and Test Setup

In this chapter the datasets used for the experiments in this thesis and the experimental setup are introduced. Section 5.1 describes the datasets. Section 5.2 presents the evaluation metrics used in this thesis and in Section 5.3 the experimental setups for land cover and land use classification are presented, where the scientific questions to be answered by the experiments are raised. The experiments and the evaluation of the results are presented in Chapter 6.

5.1. Datasets

Five datasets are used to perform experiments: Hameln, Schleswig, Mecklenburg-Vorpommern (referred to as MV in the remainder), Vaihingen and Potsdam. The first three datasets (Hameln, Schleswig, MV) are used for both land cover and land use classification. The Vaihingen and Potsdam datasets, published by the International Society for Photogrammetry and Remote Sensing (ISPRS) in the context of the 2D semantic labeling challenge (Wegner et al., 2015), are used to compare the proposed CNN for land cover classification to state-of-the-art methods. Tab. 5.1 presents an overview of these datasets and their application scenarios.

	Hameln	**Schleswig**	**MV**	**Vaihingen**	**Potsdam**
spectral bands	R, G, B, NIR	R, G, B, NIR	R, G, B, NIR	R, B, NIR	R, G, B, NIR
height model	DSM, DTM	DSM, DTM	DSM	nDSM	nDSM
#test image	1	1	256	33	38
size [km^2]	12	36	64	~1.6	3.4
size [pixel]	10000 x 30000	30000 x 30000	5000 x 5000	~2000 x 2500	6000 x 6000
GSD [cm]	20	20	10	9	5
capture time	Spring 2010	Summer 2013	August 2016	August 2008	September 2015
used in LC?	✓	✓	✓	✓	✓
used in LU?	✓	✓	✓	×	×

Table 5.1: Overview of the five datasets. LC: land cover classification; LU: land use classification; DSM: digital surface model; nDSM: normalized digital surface model; DTM: digital terrain model; R: red band, G: green band; B: blue band: NIR: near infrared band. GSD: ground sampling distance. #test image: number of test images. ✓: Yes; ×: No.

5.1.1. Hameln

This dataset was captured over the city of Hameln, Lower Saxony, Germany, in spring 2010. It covers mainly urban areas, e.g. residential areas with detached houses, densely built-up areas or industrial areas, but there are also rural areas including forest, cropland and grassland. The river Weser crosses the area as well. The whole test site covers an area of 2 x 6 km^2. In this dataset, high-resolution orthophotos with a GSD of 20 cm in four spectral bands (RGB-NIR), a digital terrain model (DTM) and a digital surface model (DSM) are available (cf. Tab. 5.1). The original aerial images and the DTM were provided by the Landesamt für Geoinformation and Landesvermessung Niedersachsen (LGLN). The DSM was generated from the aerial images by dense image matching. In all experiments in Hameln, a normalized digital surface model (nDSM), which is created by subtracting the DTM from the DSM, is used. Fig. 5.1 shows the RGB orthophoto of the whole area of Hameln.

To evaluate the results of land cover classification, a reliable reference at pixel level is required. This reference was generated by manually labeling 37 representative areas (red boxes in Fig. 5.1), each having a size of 1000 x 1000 pixels (corresponding to 200 x 200 m^2), covering areas of urban and rural characteristics. The eight land cover classes to be discerned are *building*, *sealed area (sealed.)*, *soil*, *grass*, *tree*, *water*, *car* and *clutter*. One of the areas for which land cover labels are available is shown in Fig. 5.2. The figure shows the RGB image, the nDSM and the reference labels. Additionally, this reference, referred to as *full reference*, is eroded by a circular disc of a radius of 3 pixels, so that the pixels near object boundaries can be excluded in the evaluation, which results in an *eroded reference*. Using the eroded reference and the full reference it is possible to evaluate the quality of the classification of pixels near the object boundaries.

Figure 5.1: Overview of the Hameln dataset (12 km^2). Red boxes: Labeled regions for land cover classification, resulting in a total labeled area of about 1.5 km^2.

The upper part of Fig. 5.3 shows the class distribution of all categories, which is not balanced. To describe the degree of the imbalance of the distribution, the so-called *imbalance degree* is introduced which is defined in eq. 5.1 following Johnson and Khoshgoftaar (2019):

$$\rho_{im} = \frac{\#Class_{MAX}}{\#Class_{MIN}} \tag{5.1}$$

Here, $\#Class_{MAX}$ denotes the number of samples of the class having the maximum amount of samples whereas $\#Class_{MIN}$ denotes the number of samples of the class having the minimum amount of samples. In Hameln, the imbalance degree of the land cover classes is $\rho_{im} = 44.7$.

The reference for evaluating land use classification was provided by the LGLN and was extracted from ALKIS (Amtliches Liegenschaftskatasterinformationssystem; AdV, 2008). ALKIS provides the land use of each object according to a hierarchical object catalogue. It has to be noted that the reference has been checked and corrected by experts, so that its correctness is guaranteed. In this thesis, a differentiation of categories at three semantic levels in line with that object catalogue is pursued. Due to the fact that some of the land use categories do not have sufficient training samples, some categories at the same semantic level of the original catalogue are merged. There are 4 classes at level I, 14 classes at level II and 21 classes at level III for 2945 polygons in Hameln. The lower part of Fig. 5.3 presents the class distribution at all semantic levels, and the corresponding classes and the number of objects at the finest level are presented in Tab. 5.2. Clearly, the distribution of the classes is not balanced. The imbalance degrees are $\rho_{im} = 23.2$ at level I, $\rho_{im} = 112.4$ at level II and $\rho_{im} = 105.6$ at level III.

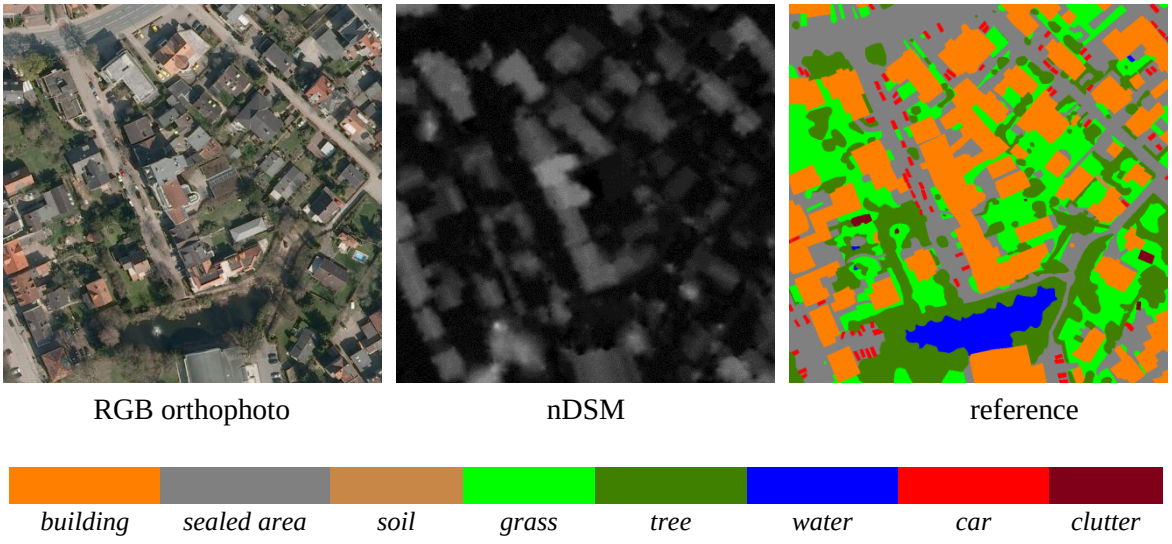


Figure 5.2: An example of a labeled area in Hameln with the RGB orthophoto, the nDSM and the corresponding full reference, each having size of 1000 x 1000 pixels. In the labeled image, each color indicates a land cover class.

5.1.2. Schleswig

The Schleswig dataset is provided by the Landesamt für Vermessung und Geoinformation Schleswig-Holstein (LVermGeo SH). It was acquired over the town Schleswig and its surroundings in Schleswig-Holstein, Germany, in summer 2013, covering an area of 6 x 6 km^2. Unlike the Hameln

dataset, the main characteristic of Schleswig is rural, dominated by cropland, grassland and forest. However, there are also some urban structures, for instance several small villages, a small town center.

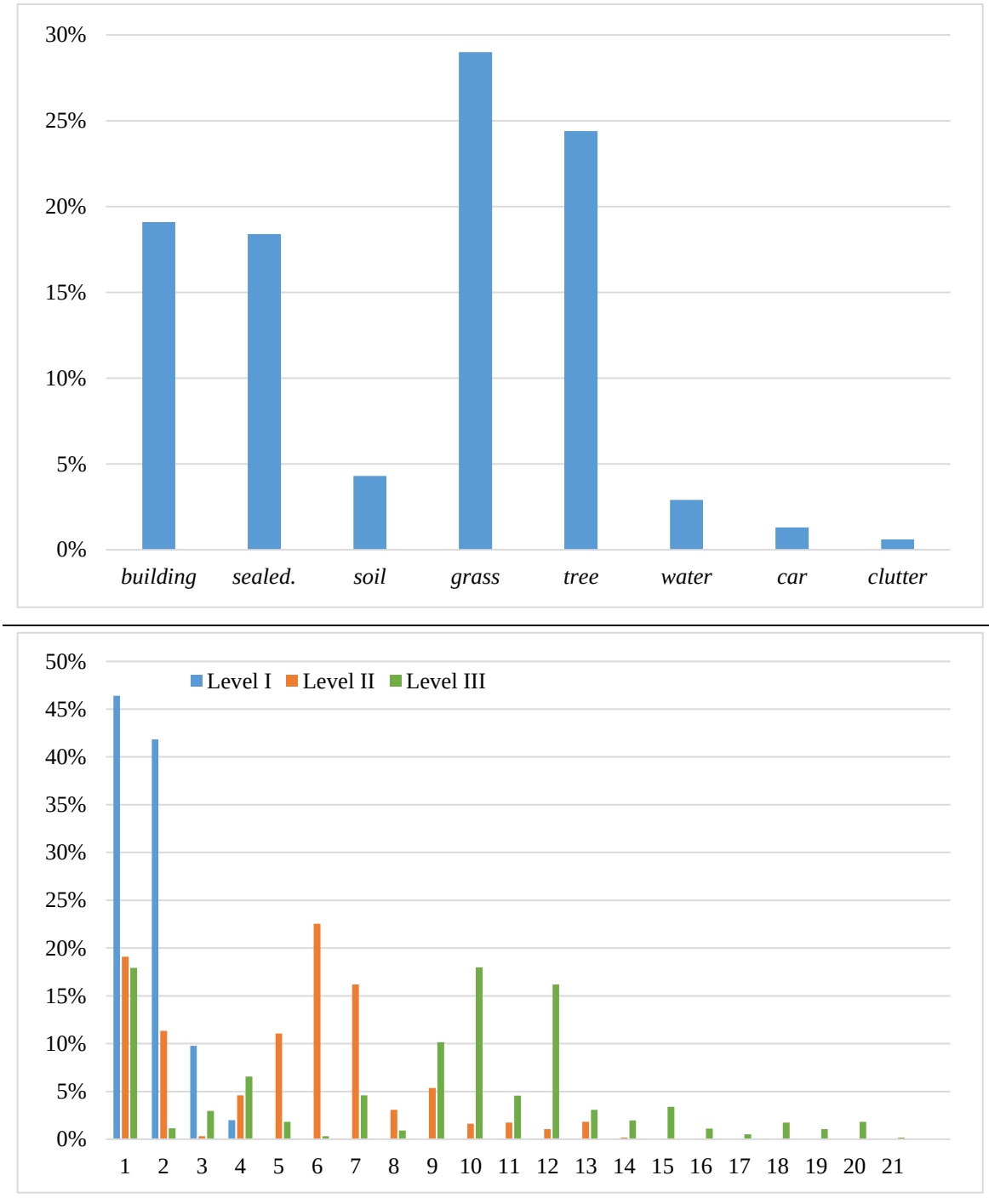


Figure 5.3: Class distribution of the dataset Hameln. Top: eight land cover categories; bottom: land use categories at three semantic levels. Here, only the numerical class IDs are shown on the X-axis for lack of space. The corresponding names are given in Tab. 5.2.

level I	level II	level III	#Hameln	#Schleswig	#MV
settlement [1]	*residential (res.) [1]*	*residential in use (res.use) [1]*	528	803	169
		extended residential (ext. res.) [2]	34	61	19
	industry (ind.) [2]	*factory (fact.) [3]*	87	39	61
		business (busi.) [4]	193	158	10
		infrastructure (infra.) [5]	54	62	27
	mixed usage (mix) [3]	*mixed usage (mix) [6]*	9	127	83
	special usage (special) [4]	*special usage (special) [7]*	135	207	17
	recreation (rec.) [5]	*leisure (leis.) [8]*	27	64	19
		park [9]	299	365	145
traffic [2]	*road traffic (ro.traf.) [6]*	*motor road (mo.road) [10]*	530	732	219
		traffic guided area (traf.area) [11]	134	75	70
	path & way (path) [7]	*path & way (path) [12]*	477	287	238
	parking lot (park.lot) [8]	*parking lot (park.lot) [13]*	91	76	13
vegetation [3]	*agriculture (agr.) [9]*	*farm land (farm) [14]*	58	214	366
		garden / fallow land (garden) [15]	100	440	312
	forest [10]	*hardwood or softwood (h/s.wood) [16]*	33	154	206
		hardwood and softwood (h&s.wood) [17]	15	134	50
	grove [11]	*grove [18]*	51	88	122
	moor or swamp (moor) [12]	*moor or swamp (moor) [19]*	31	116	165
water bodies [4]	*flowing water (flow.wat) [13]*	*flowing water bodies (flow.wat.bo.) [20]*	19	131	193
	stagnant water (stag.wat.) [14]	*stagnant water bodies (stag.wat.bo.) [21]*	40	12	52

Table 5.2: Hierarchical land use class structure and statistics about the distribution of objects in Hameln, Schleswig and MV. The class labels are given at three semantic levels (I-III) according to (AdV, 2008). Abbreviations that are used in the remainder of this thesis are shown in parentheses () and numerical class IDs in each level are shown in brackets []. The three rightmost columns give the number of land use objects of the given category in level III in the corresponding datasets.

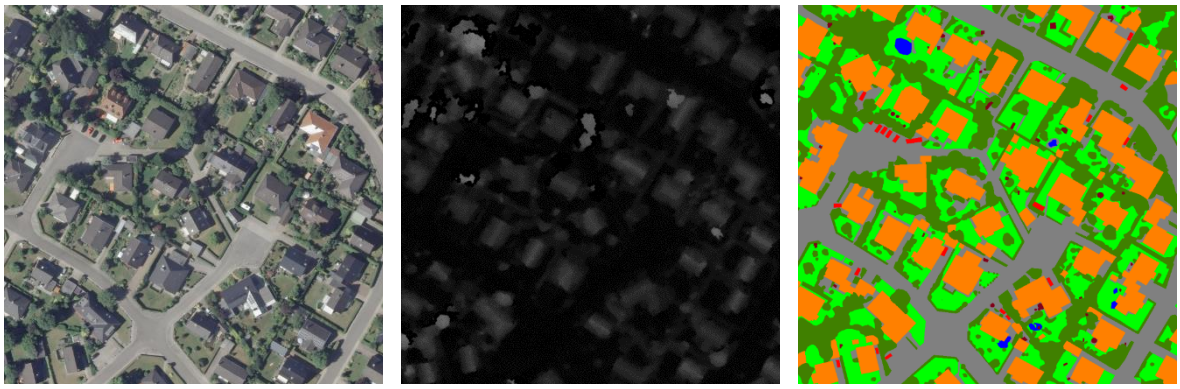
In addition to the mentioned structures, lakes, creeks and a small part of the sea are also included in this area. Fig. 5.4 shows the RGB orthophoto of Schleswig. Like in Hameln, a multi-spectral DOP, a DSM and a DTM are available for Schleswig. The DOP consists of 4 bands, i.e. R, G, B and NIR, with a GSD of 20 cm. The DSM was generated from the aerial images by dense image matching. By subtracting the DTM from the DSM, a nDSM is generated and used for all experiments using the Schleswig dataset. For land cover classification, a pixel-wise reference was generated by manually labeling 26 selected areas of both urban and rural characteristics (red boxes in Fig. 5.4). Each selected area has a size of 1000 x 1000 pixels (i.e. 200 x 200 m^2). The class structure for land cover classification is the same as in Hameln. One of the areas for which land cover labels are available is shown in Fig. 5.5. The figure shows the RGB image, the nDSM and the reference label image. Like in Hameln, this reference, referred to as *full reference*, is eroded by a circular disc of a radius of 3 pixels to obtain the *eroded reference*.

Figure 5.4: Overview of the Schleswig dataset (36 km^2). Red boxes: labeled areas for land cover classification, resulting in a total labeled area of about 1.1 km^2.

The corresponding class distribution of land cover classes in the full reference is shown in the upper part of Fig. 5.6. Obviously, the distribution of the classes is not balanced. The corresponding imbalance degree is $\rho_{im} = 39.3$, which is similar to the one of the Hameln dataset.

There is also ALKIS data provided by LVermGeo SH, which was checked and corrected by experts and serves as reference for evaluating the land use classification. The hierarchical class structure is identical to the one used in Hameln, and it is also shown in Tab. 5.2 along with the number of objects at the finest level. In Schleswig there are 4345 polygons in a total. The lower part of Fig. 5.6 presents the class distribution at all semantic levels. Clearly, the class distribution is not balanced, and the imbalance degrees are $\rho_{im} = 13.2$ at level I, $\rho_{im} = 21.1$ at level II and $\rho_{im} = 20.6$ at level III, lower than the ones of the Hameln dataset at each corresponding level.

Comparing the RGB orthophotos in Fig. 5.1 and Fig. 5.4, it is clear that there are differences between Hameln and Schleswig: deciduous trees have dense canopies in Schleswig, whereas this is not the case in Hameln. This is mainly caused by the different capturing seasons (spring in Hameln vs. summer in Schleswig). The difference in the distribution of the data of the same object types (e.g. trees and grass) indicates that there is a domain gap between both datasets. On the other hand, urban structures such as buildings and sealed areas share similar characteristics in appearance and structure.



RGB orthophoto nDSM reference

Figure 5.5: An example of a labeled area in Schleswig with the RGB orthophoto, the nDSM and the corresponding full reference, each having size of 1000 x 1000 pixels. Refer to Fig. 5.2 for the interpretation of the colors in the labeled image.

5.1.3. Mecklenburg-Vorpommern (MV)

The MV dataset was provided by the Landesamt für innere Verwaltung Mecklenburg-Vorpommern (LaiV-MV). The data was captured in August 2016 and covers an area of 64 km^2 with a GSD of 10 cm. The characteristics of MV are very similar to Schleswig: rural structures are dominant, and there are some small villages. The area covered by vegetation (e.g. grass, tree) is similar in appearance to the one in Schleswig and different from the one in Hameln. LaiV-MV provided a multi-spectral DOP and a DSM generated by laser scanning. To evaluate land cover classification, 46 tiles were manually labeled,

resulting in a total area of 11.5 km^2 of labeled data. Fig. 5.7 presents the whole area of the MV dataset and the labeled areas.

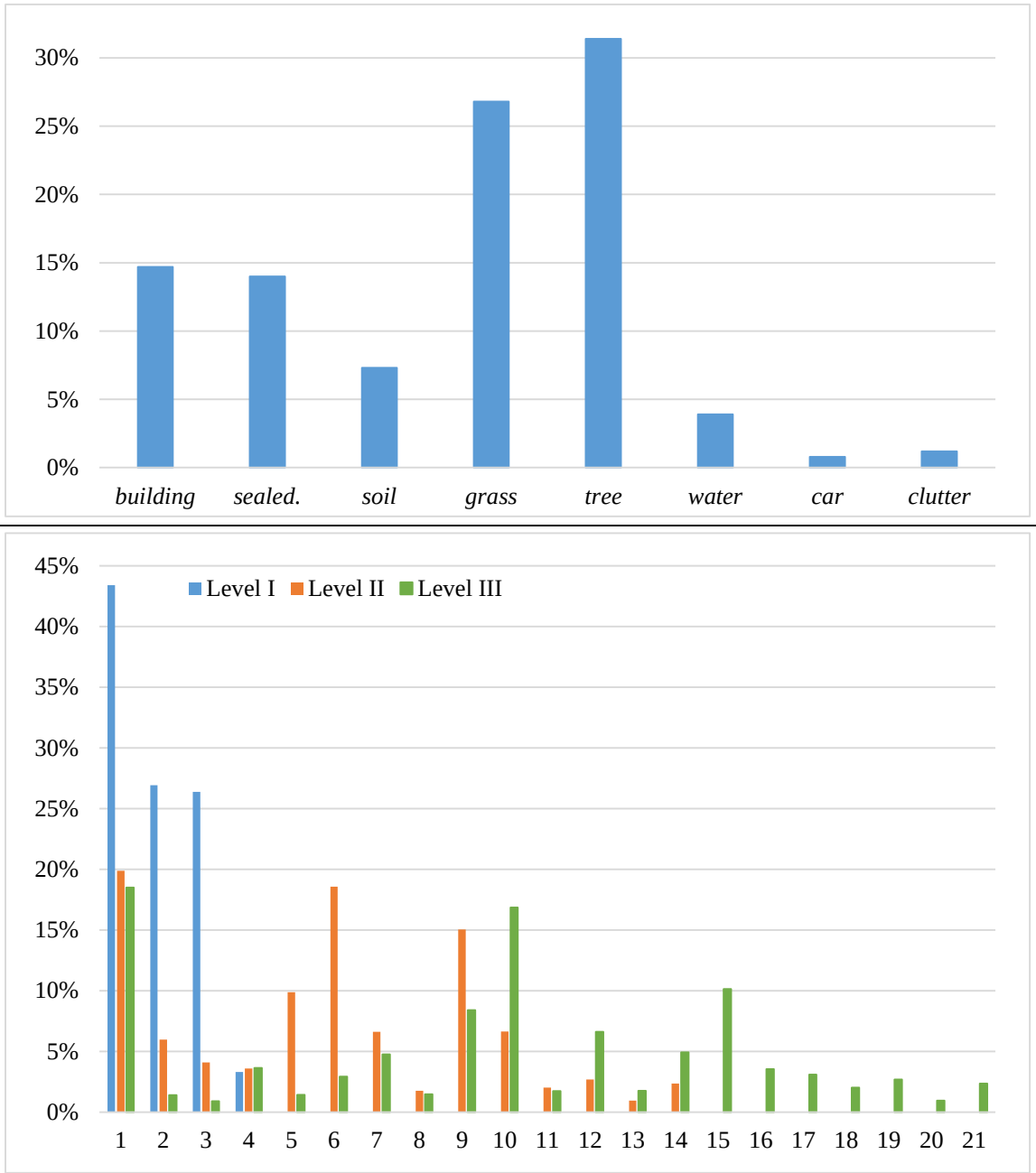


Figure 5.6: Class distribution of Schleswig. Top: eight land cover categories; bottom: land use categories at three semantic levels. Only the numerical class IDs are shown on the X-axis. The class names are given in Tab. 5.2.

Figure 5.7: Overview of the MV dataset (64 km^2). Red boxes: labeled area for land cover classification. Each has a size of 5000 x 5000 pixels, resulting in a total labeled area of about 11.5 km^2.

In MV, the land cover class structure differs from the one in Hameln and Schleswig, which is caused by a larger diversity of the land cover types. The following ten land cover classes are distinguished: *building*, *sealed area (sealed.)*, *unpaved road (un.road)*, *soil*, *crops*, *grass*, *tree*, *water*, *railway* and *clutter*. One of the areas for which land cover labels are available is shown in Fig. 5.8. The figure shows the RGB image, the DSM and the reference label. Like in Hameln and Schleswig, this reference, referred to as *full reference*, is eroded by a circular disc of a radius of 3 pixels to obtain the *eroded reference*. The original labeled images have severe problem of class imbalance, because most of them are dominated by one single class (e.g. crops). To build a relatively balanced dataset, those images are split into tiles of 500 x 500 pixels, resulting in 4600 tiles. The tiles having only one class are excluded for further processing. Finally, 524 tiles remain for training and testing.

RGB orthophoto DSM reference

building *sealed area* *unpaved road* *soil* *crops* *grass* *tree* *water* *railway* *clutter*

Figure 5.8: An example of a labeled area with the RGB orthophoto, the DSM and the corresponding full reference. Each image has a size of 5000 x 5000 pixels. Each color in the reference image indicates a land cover class.

The class distribution of the remaining tiles is shown in the upper part of Fig. 5.9, and the corresponding imbalance degree is $\rho_{im} = 16.8$.

LaiV-MV also provided ALKIS data to evaluate land use classification. Here, the hierarchical land use class structure is the same as the one in Hameln and Schleswig. Tab. 5.2 gives the class structure and the number of objects at the finest level in MV. There are 2840 polygons in MV. The lower part of Fig. 5.9 presents the class distributions at all semantic levels, and the imbalance degrees are $\rho_{im} = 1.6$ at level I, $\rho_{im} = 27.3$ at level II and $\rho_{im} = 57.4$ at level III. One can see that the four categories at level I have a very similar number of samples, thus, at this level there is a low degree of imbalance.

5.1.4. Vaihingen and Potsdam

The Vaihingen and Potsdam datasets are provided in the context of the ISPRS 2D Semantic Labeling Challenge. Both mainly consist of urban structures. As there is only a pixel-wise land cover reference, the datasets are only used for evaluation of land cover classification. In Vaihingen, there are 33 tiles available for training and testing, each having a size of about 2000 x 2500 pixels and assigned with a unique ID (see the top part of Fig. 5.10); in Potsdam, there are 38 tiles in total, each having a size of 6000 x 6000 pixels and also assigned with a unique ID (see the bottom part of Fig. 5.10).

Image data containing the NIR, R and G bands with a GSD of 9 cm are provided in Vaihingen, whereas the R, G, B and NIR bands with a GSD of 5 cm are provided in Potsdam. The images consisting of the NIR, R and G bands are called color infrared (CIR) images. In addition, nDSMs for both sites, generated from aerial images by dense image matching, are provided by Gerke et al. (2015).

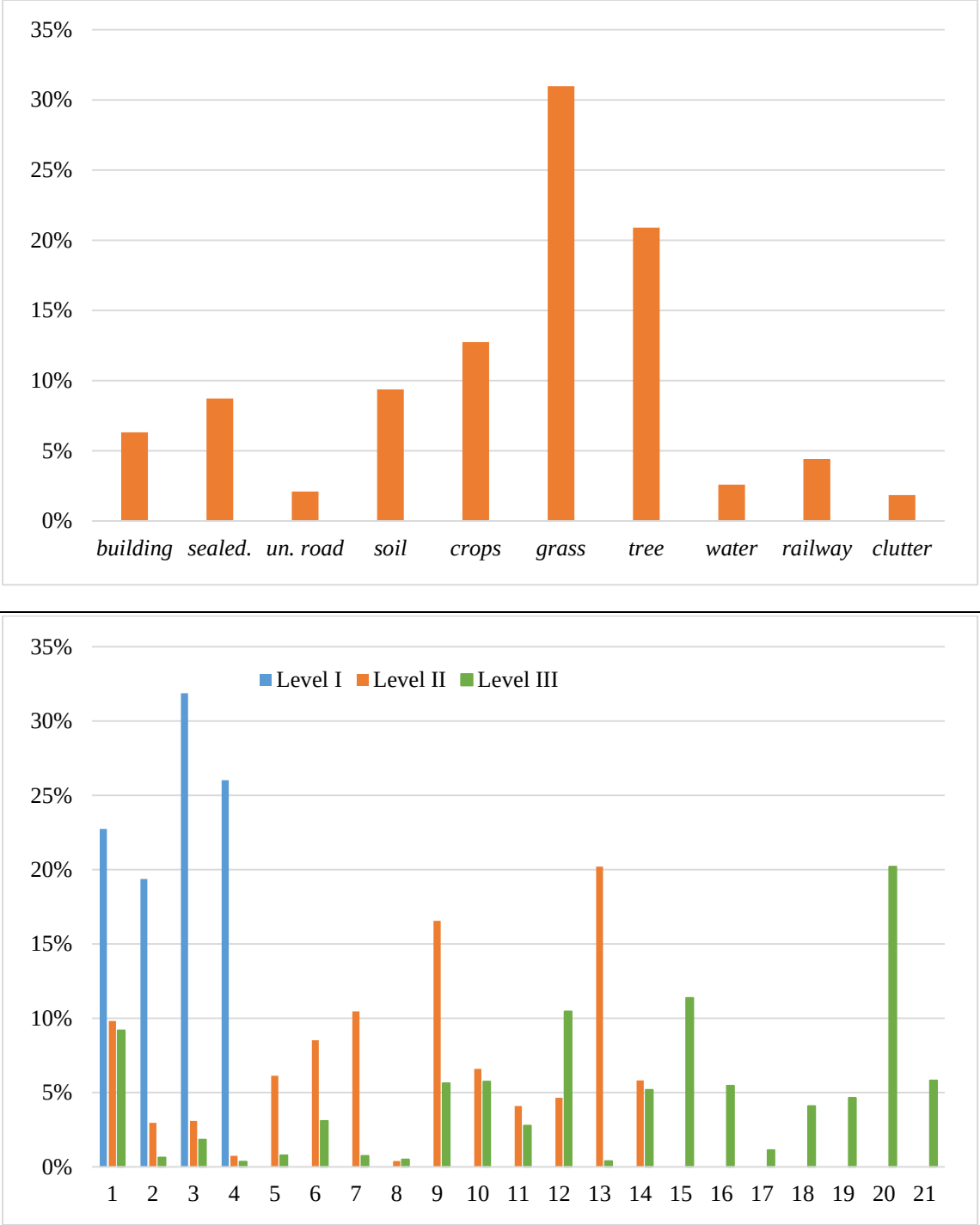


Figure 5.9: Class distribution of MV. Top: ten land cover categories; bottom: land use categories at three semantic levels. Only the numerical class IDs are shown on the X-axis. The class names are given in Tab. 5.2.

For both datasets, the same land cover classes are to be discerned, namely *impervious surface (impervious)*, *building*, *low vegetation (low veg.)*, *high vegetation (high veg.)*, *car* and *clutter*. Beyond the full reference of both datasets, an eroded reference is provided as well, which was generated by eroding the full reference with a circular disc of a radius of 3 pixels. The evaluation using both datasets based on the eroded reference is conducted to compare the results of the land cover classification method presented in this thesis with state-of-the-art methods. Fig. 5.11 presents examples of both datasets with CIR images, nDSMs and labeled tiles. Like in Hameln, Schleswig and MV, there is also the problem of class imbalance in both datasets. Fig. 5.12 presents the class distributions. The imbalanced degree is $\rho_{im} = 39.9$ in Vaihingen and $\rho_{im} = 57$ in Potsdam.

Figure 5.10: Overview of Vaihingen (top) and Potsdam (bottom).

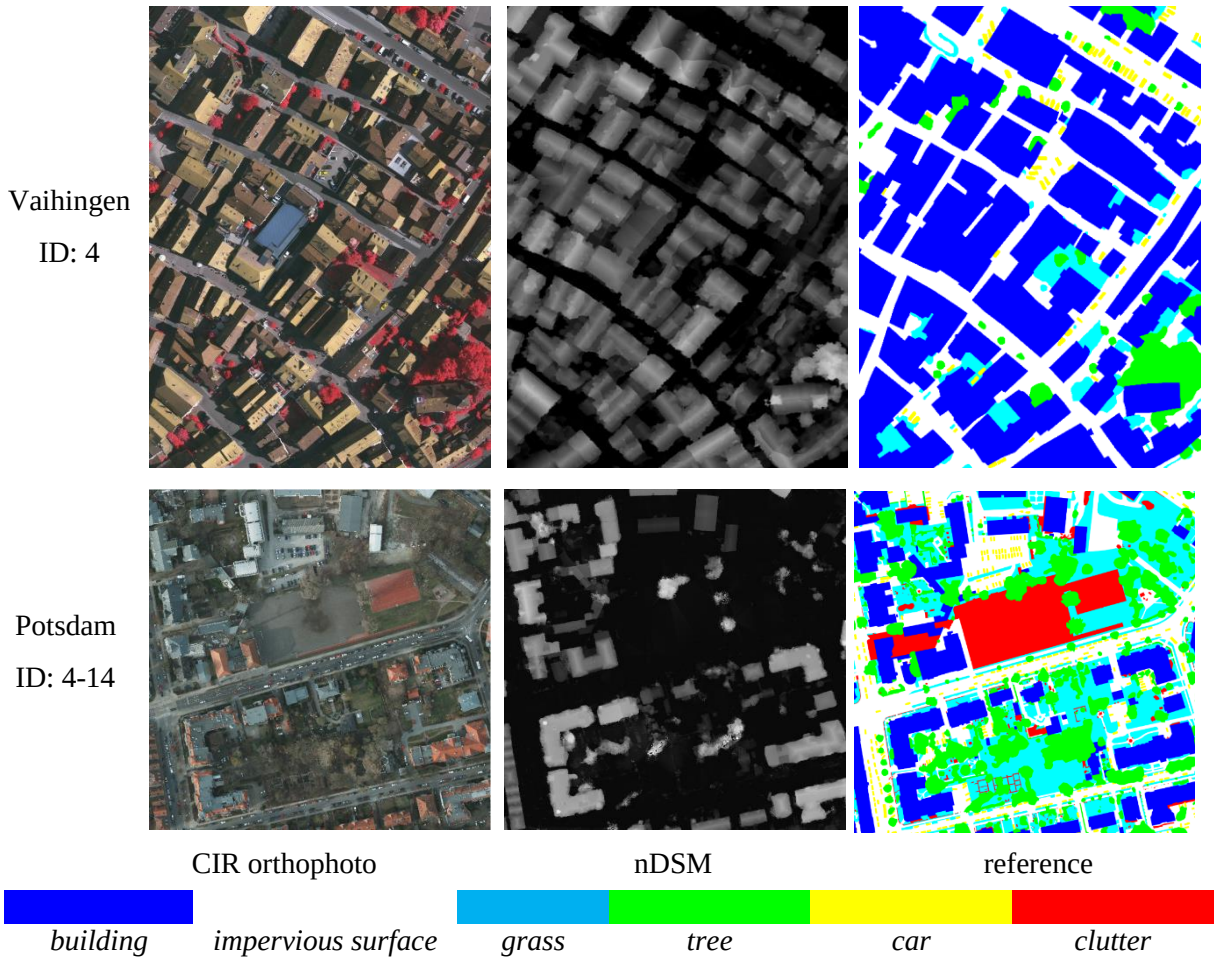


Figure 5.11: Examples for labeled areas with one CIR orthophoto, a nDSM and the corresponding full reference of Vaihingen (top) and Potsdam (bottom). Each color in the reference image indicates a land cover class.

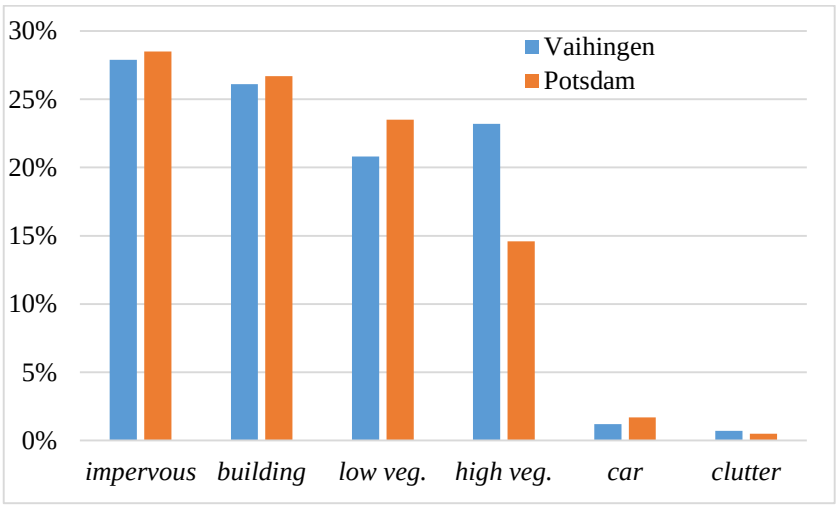


Figure 5.12: Class distributions of Vaihingen and Potsdam.

5.2. Evaluation Metrics

The evaluation is based on the confusion matrix, which forms the basis for measuring the performance of an algorithm. In this matrix, each row represents the primitives of a reference class while each column represents the primitives of a predicted class. For a multiclass classification with $\mathbb{C} = \{C^1, \dots, C^M\}$ classes, the element c_{ij} is the number of primitives, i.e. pixels in land cover classification and polygons in land use classification, belonging to reference class C^i which were classified as class C^j. Tab. 5.6 presents an example of a confusion matrix for $M = 3$:

		Classes: **classification**		
		C^1	C^2	C^3
Classes: **reference**	C^1	c_{11}	c_{12}	c_{13}
	C^2	c_{21}	c_{22}	c_{23}
	C^3	c_{31}	c_{32}	c_{33}

Table 5.6: Confusion matrix for the case of $M = 3$. The numbers on the main diagonal correspond to correct classifications.

The numbers in the main diagonal (i.e. $i = j$) correspond to the correct classification results, and the off-diagonal (i.e. $i \neq j$) elements correspond to misclassifications. On the basis of confusion matrix, the overall accuracy (OA) can be derived according to eq. 5.2:

$$OA = \frac{\sum_i^M c_{ii}}{\sum_j^M \sum_i^M c_{ij}} \cdot 100\% \tag{5.2}$$

The OA is the percentage of correctly classified primitives. To derive class-specific metrics, the precision and recall of a class are determined according to eq. 5.3 and eq. 5.4:

$$Precision_i = \frac{c_{ii}}{\sum_j^M c_{ji}} \cdot 100\% \tag{5.3}$$

$$Recall_i = \frac{c_{ii}}{\sum_j^M c_{ij}} \cdot 100\% \tag{5.4}$$

The F1 score is the harmonic mean of precision and recall:

$$F1_i = \frac{2 \cdot Precision_i \cdot Recall_i}{Precision_i + Recall_i} \tag{5.5}$$

Finally, the mean F1 score, denoted as mF1, for all classes is determined by using the F1 scores of all classes:

$$mF1 = \frac{1}{M}\sum_{i=1}^{M} F1_i \tag{5.6}$$

For all experiments, the determined quality measures are the overall accuracy, the F1 score for each category and the mean F1 scores. In land cover classification, OA and F1 scores are determined at pixel level whereas in land use classification they are determined at object level.

5.3. Experimental Setup

In this section the experimental setup for land cover classification (Section 5.3.1) and land use classification (Section 5.3.2) are described, which includes the data pre-processing, data augmentation and setup for the experiments to answer the corresponding scientific questions.

5.3.1. Land Cover Classification

For land cover classification, all datasets presented in Section 5.1 are used. The test setup for land cover classification is presented in Section 5.3.1.1. Section 5.3.1.2 gives an overview of all experiments. In Sections 5.3.1.3 - Section 5.3.1.7, the setup of a series of experiments is described, each designed to answer a specific set of scientific questions.

5.3.1.1. Test Setup

Previous experiments reported in (Yang et al., 2019) have shown that the image consisting of red, infrared and height data, referred to as RID image, is very well-suited to support pixel-wise land cover classification. For Hameln, Schleswig, Vaihingen and Potsdam datasets, RID images are created by combing the R, NIR and nDSM data. Because there is only a DSM available in MV, the RID images consist of the R, NIR and DSM data. The RGB and RID images of all datasets serve as the input of the proposed *FuseNet-lite* network, which requires an input of 256 x 256 pixels. However, the available images of all datasets are much larger than that. Therefore, large images need to be split into patches of 256 x 256 pixels.

For the Hameln, Schleswig and MV datasets, each large image (1000 x 1000 pixels in Hameln and Schleswig, 5000 x 5000 pixels in MV) is split into non-overlapping tiles of 500 x 500 pixels, resulting in 148, 104 and 524 smaller tiles, respectively. Subsequently, the tiles of 500 x 500 pixels are split into

four overlapping patches of size 256 x 256, corresponding to the input size. For the Vaihingen and Potsdam datasets, windows of 256 x 256 pixels with an overlap of 128 pixels in both spatial dimensions are extracted, resulting in 4426 and 50784 patches in Vaihingen and Potsdam, respectively.

For all datasets, the generated patches are augmented to enlarge the total amount of training patches. Firstly, all patches of 256 x 256 pixels are flipped horizontally and vertically, and then rotated by 90°, 180° and 270°. However, the resultant number of training patches is still small in Hameln, Schleswig, MV and Vaihingen. Two additional random rotations by an angle in [3°, 20°] for all patches generated so far are performed for these datasets. Finally, there are 14208, 9984, 110400, 53112 and 304704 patches in Hameln, Schleswig, MV, Vaihingen and Potsdam, respectively.

In the experiments for land cover classification, two test scenarios are differentiated. In a first set of experiments, the prediction variability of *FuseNet-lite* and the influence of the hyperparameters on the classification results are investigated. To achieve that goal, the Hameln and Vaihingen datasets are used. In Hameln, about 30% of the tiles of 500 x 500 pixels are selected as test data, 60% of the tiles are selected as training data and the remaining 10% of the tiles are used for validation. In Vaihingen, the division of training, validation and test data is presented in Tab. 5.3, which follows the setting in (Yang et al., 2021).

Second, to evaluate how well *FuseNet-lite* performs for each individual dataset, all datasets are used for training and testing. For the Hameln, Schleswig and MV datasets, 3-fold cross-validation is applied. The tiles of 500 x 500 are randomly split into three subsets of equal size, and in each test run, one subset is used for testing and other two subsets are used for training. It has to be noted that about 10% of all training tiles are selected to form a validation set in each test run. For Vaihingen and Potsdam, the division of training, validation and test data is presented in Tab. 5.3, following the previous experiments in (Yang et al., 2021). Training will be stopped if the training loss does not decrease for about 10 epochs.

In the inference procedure, the class labels for a patch of 256 x 256 pixels are predicted six times for the original image and variants that are flipped and rotated (90°, 180° and 270°) like the training images, and the final classification is based on the combined scores, which are obtained by multiplying the six probabilistic scores for every patch.

Usage	IDs in Vaihingen	IDs in Potsdam
Training	1, 3, 11, 13, 15, 17, 21, 26, 28, 32, 34, 37	02-10, 02-12, 03-10, 03-11, 03-12, 04-11, 04-12, 05-10, 05-11, 05-12, 06-07, 06-08, 06-09, 06-11, 06-12, 07-07, 07-08, 07-09, 07-10, 07-11, 07-12
Validation	5, 7, 23, 30	02-11, 06-10, 04-10
Testing	2, 4, 6, 8, 10, 12, 14, 16, 20, 22, 24, 27, 29, 31, 33, 35, 38	02-13, 02-14, 03-13, 03-14, 04-13, 04-14, 04-15, 05-13, 05-14, 05-15, 06-13, 06-14, 06-15, 07-13

Table 5.3: The IDs of image tiles used for training, validation and testing in Vaihingen and Potsdam. Refer to Fig. 5.10 for an overview of the location of each tile.

5.3.1.2. Overview of all Experiments

In the context of land cover classification, there are some scientific questions to be answered by the experiments. An overview of all experiments is presented in Tab. 5.4. The research questions and the related sets of experiment are described in the corresponding sections.

Experiment Set	Section	Used Dataset	Used Network
Variability-LC	5.3.1.3	Hameln; Vaihingen	*FuseNet-lite*
Hyper-LC	5.3.1.4	Hameln; Vaihingen	
Learn-Skip	5.3.1.5	MV; Hameln; Schleswig; Vaihingen; Potsdam	
No-Skip			*FuseNet-lite-NoSkip*
Add-Skip			*FuseNet-lite-AddSkip*
ComHS-LC	5.3.1.7	Hameln; Schleswig	*FuseNet-lite*
ComVP-LC		Vaihingen; Potsdam	

Table 5.4: Overview of all experiments in the context of land cover classification. The explanation of each experiment set is presented in the corresponding section.

5.3.1.3. Prediction Variability of *FuseNet-lite*

When training a CNN, all trainable parameters are randomly initialized, and then their values are updated during training. Due to the random initialization, the final weights of the parameters will be different, which will have an impact on the final predictions for the same test data. This gives rise to one scientific question to be answered by experiments:

- When using the same training and test data, how large is the variability of the predictions made by *FuseNet-lite* due to its random initialization?

To investigate the variability of the results, ten experiments are run using the same training, validation and test data, corresponding to the set of experiments called Variability-LC in Tab. 5.4. The same hyperparameters are used for all experiments and they are shown in Tab. 5.5. In each experiment, all trainable parameters are initialized randomly. After obtaining multiple predictions on the same test data, means and standard deviations of OA and the mean F1 score are computed. It is expected that small standard deviations are good for the application. To conduct these experiments, the Hameln and Vaihingen datasets described Section 5.3.1.1 are used.

5.3.1.4. Impact of the Hyperparameter Settings

There are some hyperparameters of *FuseNet-lite* which are required to be set before training and which are not updated during training. Tab. 5.5 presents an overview of these hyperparameters. The values of some hyperparameters were set to values found empirically. This applies to the value of the weight of the regularization term λ, the decay factor for learning rate δ, the total number of training epochs E_{total}, and the value of the decay epoch of learning rate E_{decay}. These parameters were fixed by using values found empirically.

The remaining hyperparameters (base learning rate η, mini batch size Ω and penalty term γ of the proposed loss in eq. 4.5) are selected for investigation, so as to know to which degree they affect the classification performance. By a set of experiments described in Section 6.1.2, the following scientific questions should be answered:

a. How do the hyperparameters affect the classification performance of *FuseNet-lite*?
b. To what degree can the proposed focal loss contribute to the classification, in particular for the underrepresented classes? It has to be noted that, for $\gamma = 0$, the focal loss corresponds to the standard cross entropy loss.

As there is a large amount of possible combinations of the investigated hyperparameters, it is infeasible to run experiments for all combinations. Thus, a simplification is made: while investigating one hyperparameter, all other ones are set to default values, which are presented in Tab. 5.5. For the three hyperparameters, the values used in this test series are presented in Tab. 5.6. The Hameln and Vaihingen datasets described Section 5.3.1.1 are used for conducting these experiments. This set of experiments is called Hyper-LC in Tab. 5.4.

Symbol	Description	Default Values
η	base learning rate	0.1
Ω	mini batch size	10
γ	the weight of the penalty term in eq. 4.5	1
δ	decay factor of learning rate	0.1
λ	weight for regularizing the trainable parameters	0.0005
E_{decay}	decay epoch interval of learning rate	15
E_{total}	total number of training epochs	30

Table 5.5: The hyperparameters of *FuseNet-lite* with their default values applied for land cover classification.

Symbol	Description	Investigation Values
η	base learning rate	0.001; 0.01; 0.1; 1
Ω	mini batch size	3; 5; 10; 15; 20
γ	the weight of the penalty term in eq. 4.5	0.5; 0; 1; 2; 5

Table 5.6: The hyperparameters to be investigated and the values used in the experiments.

5.3.1.5. Effectiveness of the learnable Skip-Connections

One of the contributions of this thesis in the context of land cover classification is the development of the proposed learnable skip-connections, which are expected to lead to a better performance than the commonly used connections based on elementwise addition. In this regard, the scientific questions to be answered by experiments described in Section 6.1.3 are:

a. To which degree can the skip-connections contribute to the classification?
b. Is the performance increased by the proposed learnable skip-connections in comparison to skip-connections using elementwise addition?
c. Do skip-connections improve the identification of pixels near object boundaries?

To answer these questions, a series of experiments is conducted based on *FuseNet-lite*, *FuseNet-lite-AddSkip* and *FuseNet-lite-NoSkip* over all datasets. Firstly, baseline results on each dataset are obtained by training and testing *FuseNet-lite*, corresponding to the set of experiments called Learn-Skip in Tab. 5.4; afterwards, *FuseNet-lite-AddSkip* and *FuseNet-lite-NoSkip* are trained and tested for each dataset, which corresponds to the sets of experiment called Add-Skip and No-Skip in Tab. 5.4, respectively. Based on the results of Learn-Skip, Add-Skip and No-Skip, question (a) can be answered by comparing them. Question (b) can be answered by comparing the results of Learn-Skip and Add-Skip. The final question (c) is answered by using both the full reference and the eroded reference to evaluate all results for the pixels near object boundaries.

5.3.1.6. Performance of *FuseNet-lite*

With respect to land cover classification, the core part is the evaluation of *FuseNet-lite* on different datasets. Therefore, the scientific questions to be answered by the experiments in Section 6.1.4 are:

a. How well does *FuseNet-lite* perform on each dataset?
b. Can the results of *FuseNet-lite* reach the level of other state-of-art methods?

To answer these questions, the results of the experiments of Learn-Skip are used. The question (a) is answered by evaluating the classification results on the datasets Hameln, Schleswig, MV, Vaihingen and Potsdam with the corresponding full reference and comparing the final results over all datasets. Subsequently, to answer question (b) the results achieved of the Vaihingen and Potsdam datasets are evaluated using the eroded references, which allows a comparison with state-of-art methods. The comparison considers the scoreboard of the ISPRS benchmark (Wegner et al., 2017), but also other recent publications (the scoreboard has not been updated anymore since 2018).

5.3.1.7. Combining Datasets

CNN-based methods require large amounts of training data to learn the unknown parameters. However, labeling large amounts of data is economically expensive and time-consuming. The availability of many public datasets (with reference) makes it possible to combine different datasets to form a larger dataset. In this regard, one scientific question to be answered by experiments in Section 6.1.5 is:

- Can the classifier trained on a combined dataset deliver better results than a classifier trained on the individual datasets?

The classification performance is expected to benefit from enlarging the datasets, because the number and the variability of the training samples per class is increased. As Hameln and Schleswig have the same land cover class structure, they are combined to form a new dataset. The new dataset is again divided into three subsets for cross validation. The main difference of Hameln and Schleswig is the different appearance of vegetation objects caused by different capturing seasons (spring vs. summer). Urban structures such as *building*, *asphalt* have a similar appearance and similar structural properties. The Vaihingen and Potsdam datasets are also combined because they have the same class land cover structure. The main differences of these two datasets are the different spatial resolutions of images (9 cm vs. 5 cm) and the different capturing time (August vs. September). As both datasets are mainly dominated by urban structures, classes such as *building* or *impervious surface* have a similar appearance and similar structural properties. It has to be noted that the differences make objects of the same type look different. The different appearance of objects results in domain gap between datasets. The experiment using the combined Hameln and Schleswig datasets corresponds to the experiment set called ComHS-LC in Tab. 5.4, whereas the one related to the Vaihingen and Potsdam datasets corresponds to the experiment set called ComVP-LC in Tab. 5.4.

5.3.2. Land Use Classification

For land use classification, three of the datasets presented in Section 5.1 are used, namely Hameln, Schleswig and MV.

5.3.2.1. Input Configurations

In this thesis, different input configurations are used for land use classification. This section gives a brief overview of all input configurations based on the involved datasets, which are presented in Tabs. 5.7 and 5.8.

In some experiments, mainly focusing on the impact of polygon representation, for patch generation all available data and land cover posteriors are used and only the tiling approach is applied. Thus, $N_{DOP} = 4$ and $N_{3D} = 1$ for all datasets, and $N_{LC} = 8$ for the Hameln and Schleswig datasets, whereas $N_{LC} = 10$ for the MV dataset. Depending on the representation, $N_{shape} = 1$ for the mask representation whereas $N_{shape} = 0$ for the implicit representation. Tab. 5.7 presents an overview of the input configurations for this scenario with the total number of bands of each type of input (i.e. N_{input}).

In other experiments, three cases of representing land cover information are used. In these experiments, only mask representation of polygons is considered, so that $N_{shape} = 1$ is valid for all datasets in these experiments. In the first case, no land cover information is used, i.e. only image and height data are used. Thus, $N_{DOP} = 4$, $N_{3D} = 1$, $N_{LC} = 0$ for all datasets. This case corresponds to *none* in the column of Land Cover in Tab. 5.8. Second, land cover labels are used, resulting in $N_{DOP} = 4$, $N_{3D} = 1$, $N_{LC} = 1$ for all datasets, which corresponds to *label* in the column of Land Cover in Tab. 5.8. Third, only land cover posteriors are used, resulting in $N_{DOP} = 0$ and $N_{3D} = 0$ for all datasets, and $N_{LC} = 8$ for the Hameln and Schleswig datasets, whereas $N_{LC} = 10$ for the MV dataset. This case corresponds to *posteriors* in the column of Land Cover in Tab. 5.8. Furthermore, the combination approach presented in Section 4.3.3.3, i.e. the COM-TS, combines patches generated by tiling and scaling. This only applies for cases *none* and *posteriors* (in the experiment sections it will show that the ensemble of the network variants using these two input configurations delivers the best results, thus, the COM-TS approach is only applied for the two cases). All cases with the total number of bands of each input configuration are presented in Tab. 5.8. In Tabs. 5.7 and 5.8, each input configuration is assigned a unique name, which is used in the subsequent sections.

5.3.2.2. Test Setup

For all input configurations presented in Tabs. 5.7 and 5.8, data augmentation is applied. All generated patches are flipped horizontally and vertically. In addition, random rotations are performed as well. In the context of tiling, the number of random rotation angles depends on polygon size. For

large polygons, one random angle is selected for each interval of 30°, whereas one random angle is selected for each interval of 5° for small polygons. Thus, the number of random rotation angles is 12 for large polygons and 72 for small polygons. Subsequently, patches are generated for the rotated polygons using the tiling approach. These different parameters result from the difference in the number of polygons that have to be split, which is about six times larger than the number of small polygons that need not be split. In the context of scaling, for all polygons one random rotation angle is selected for each interval of 10°, resulting in 36 random rotation angles. Afterwards, the scaling operation is performed to obtain scaled patches. Tab. 5.9 presents an overview of the number of patches generated by tiling and scaling after performing augmentation over all involved datasets.

Input Configuration	**Patch Generation Approach**	**Representation**	N_{input}		
			Hameln	**Schleswig**	**MV**
Mask-T-Full	Tiling	mask	14	14	16
Black-T-Full		implicit with black color	13	13	15
Gauss-T-Full		implicit with Gaussian pattern			

Table 5.7: Input configurations for different shape representation methods. All available data and land cover posteriors are used.

Input Configuration	**Patch Generation Approach**	**Land Cover**	N_{input}		
			Hameln	**Schleswig**	**MV**
Mask-T-NoLC	Tiling	*none*	6	6	6
Mask-T-LabelLC		*label*	7	7	7
Mask-T-ProbLC		*posteriors*	9	9	11
Mask-S-NoLC	Scaling	*none*	6	6	6
Mask-S-LabelLC		*label*	7	7	7
Mask-S-ProbLC		*posteriors*	9	9	11
Mask-Com-NoLC	COM-TS	*none*	6	6	6
Mask-Com-ProbLC		*posteriors*	9	9	11

Table 5.8: Overview of the input configurations, all based on mask representation. Refer to main text for details.

Approach	**Hameln**	**Schleswig**	**MV**
Tiling	354178	479978	1184177
Scaling	122174	163818	42221
COM-TS	391550	548740	1192469

Table 5.9: Overview of the number of patches generated by tiling, scaling and COM-TS in Hameln, Schleswig and MV.

Similarly to land cover classification, two test scenarios are investigated for land use classification. First, to investigate the prediction variability of *LuNet-lite-JO* and the impact of the hyperparameters, the Schleswig dataset is used, because it has the maximum number of polygons. For that purpose, about 40% of the land use objects are randomly selected to form the test data, 50% are used for training and the remaining 10% are used for validation. Second, to evaluate the performance of the proposed *LuNet-lite-JO* network and to compare different network variants, the Hameln, Schleswig and MV datasets are used in a set of experiments with 6-fold cross-validation. Each dataset is split into six blocks. The block size is 2 km^2, 6 km^2 and about 10.6 km^2 for Hameln, Schleswig and MV, respectively. The number of land use objects is 2945, 4523 and 2840 in Hameln, Schleswig and MV, respectively. In each test run, five blocks are used for training and the rest one for testing. In each run, about 15% of the training samples (i.e. database objects) are randomly chosen for validation, and the rest is used for determining the network parameters. Training will be stopped if the training loss does not decrease for about 3 epochs. In all cases, land cover information determined by *FuseNet-lite* for each individual dataset is used if the input of a network variant requires this information.

5.3.2.3. Overview of all Experiments

In the context of land use classification, there are some scientific questions to be answered by the experiments. An overview of all experiments is presented in Tab. 5.10. The research questions and the related sets of experiment are described in the corresponding sections.

Experiment Set	Section	Used Dataset	Used Input Configuration	Used Network
Variability-LU	5.3.2.4	Schleswig	Mask-T-Full	*LuNet-lite-JO*
Hyper-LU	5.3.2.5	Schleswig	Mask-T-Full	
Baseline-JO	5.3.2.6	MV; Hameln; Schleswig	Mask-T-Full	
Exp-MT				*LuNet-lite-MT*
Exp-F2C				*LuNet-lite-F2C*
Inves-BG	5.3.2.7		Black-T-Full; Gauss-T-Full	*LuNet-lite-JO*
Inves-LC	5.3.2.8		Mask-T-NoLC; Mask-T-LabelLC; Mask-T-ProbLC	
Inves-Scaling	5.3.2.9		Mask-S-NoLC; Mask-S-ProbLC	
Inves-ComTS	5.3.2.9		Mask-Com-NoLC; Mask-Com-ProbLC	
ComHS-LU	5.3.2.11	Hameln; Schleswig	Mask-T-NoLC; Mask-T-ProbLC; Mask-S-NoLC; Mask-S-ProbLC	

Table 5.10: Overview of all experiments in the context of land use classification. The input configurations are presented in Tabs. 5.7 and 5.8. The explanation of each experiment set is presented in the corresponding section.

5.3.2.4. Prediction Variability of *LuNet-lite-JO*

The first scientific question with respect to land use classification is:

- When using the same training and test data, how large is the variability of the predictions made by *LuNet-lite-JO* due to its random initialization?

To answer this question, ten experiments are conducted using the Schleswig dataset described in Section 5.3.2.2, corresponding to the set of experiment called Variability-LU in Tab. 5.10. The input configuration Mask-T-Full in Tab. 5.7 is used in these experiments. The same training, validation and test data are used in all experiments. The same hyperparameters are used for all experiments; the used values are shown in Tab. 5.11. All trainable parameters are initialized randomly. After obtaining multiple predictions on the same test data, means and standard deviations of OA and mean F1 score are computed. It is expected that small standard deviations are good for the application.

Symbol	Description	Default Values
η	base learning rate	0.001
Ω	mini batch size	30
ϵ	the weight of the penalty term in eqs. 4.16-4.17	1
δ	decay factor of learning rate	0.1
λ	weight for regularizing the trainable parameters	0.0005
E_{decay}	decay epoch interval of learning rate	4
E_{total}	total number of training epochs	8

Table 5.11: The hyperparameters of *LuNet-lite-JO* with their default values applied for land use classification.

5.3.2.5. Impact of the Hyperparameter Settings

Similarly to land cover classification, the hyperparameters to train *LuNet-lite-JO* are investigated as well to know their impact on the results. The corresponding scientific questions to be answered by the experiments in Section 6.2.2 are:

a. How do these hyperparameters affect the performance of *LuNet-lite-JO*?
b. To which degree can the proposed focal-loss-style penalty term in loss function (eqs. 4.16 and 4.17) contribute to the classification, in particular for underrepresented classes?

As for land cover classification, three hyperparameters are selected for investigation, which are presented in Tab. 5.12 along with the investigated values. As there is a large amount of possible

combinations of the hyperparameters, it is infeasible to run experiments for all combinations. Thus, a simplification is made: while investigating one hyperparameter, all others are set to be default values presented in Tab. 5.11. These experiments correspond to the set of experiments called Hyper-LU in Tab. 5.10. Again, The input configuration Mask-T-Full in Tab. 5.7 is used for this investigation.

Symbol	Description	Values
η	base learning rate	0.0001; 0.001; 0.01; 0.1; 1
Ω	mini batch size	2; 5; 10; 20; 30
ϵ	the weight of the penalty term in eqs. 4.16-4.17	0.5; 0; 1; 2; 5

Table 5.12: The hyperparameters to be investigated and values used in these experiments.

5.3.2.6. Impact of Joint Optimization

With respect to hierarchical land use classification, the main contribution of this thesis is the proposed JO strategy to achieve hierarchical consistency of the predictions. In this regard, the scientific question to be answered by the experiments in Section 6.2.3 is:

- Does the land use classification benefit from the proposed joint optimization (*LuNet-lite-JO*) strategy, in comparison with multi-task learning (*LuNet-lite-MT)* and F2C post-processing (*LuNet-lite-F2C*)?

To answer this question, a series of experiments is conducted with each dataset. At first, *LuNet-lite-JO* is trained and tested using the input configuration Mask-T-Full for each dataset separately, serving as comparison baseline and corresponding to the experiment set called Baseline-JO in Tab. 5.10. Subsequently, *LuNet-lite-MT* and *LuNet-lite-F2C* are trained and tested using the same data, corresponding to the experiment sets called Exp-MT and Exp-F2C in Tab. 5.10, respectively. The settings of hyperparameters for *LuNet-lite-MT* and *LuNet-lite-F2C* are the same as for *LuNet-lite-JO* (cf. Tab. 5.11). Finally, the results obtained for each dataset in these sets of experiments are compared to highlight the influence of the proposed JO strategy on the classification.

5.3.2.7. Impact of the Polygon Representation

With respect to the polygon representation, the scientific questions to be answered by experiments in Section 6.2.4 are:

a. Is the representation by a binary mask superior to the implicit one?
b. In implicit polygon representation, does the background color or pattern affect the performance?

The experiments of the set called Inves-BG in Tab. 5.10 are conducted, which require patches based on black background and Gaussian background, corresponding to the input configurations Black-T-Full and Gauss-T-Full in Tab. 5.7, respectively, and the network variant *LuNet-lite-BG-JO*. For each dataset, the results obtained from the experiment set Inves-BG are compared to those obtained in Baseline-JO to answer both questions.

5.3.2.8. Impact of Land Cover Information

In this thesis, land cover information serves as additional input for the land use classification. It is expected that the pixel-level semantic information will positively contribute to the land use classification. Regarding to the contributions of land cover information, the scientific questions to be answered by the experiments in Section 6.2.5 are:

a. Does the land cover information positively contribute to land use classification?
b. What is the optimal way to integrate the land cover information?

In this investigation, inputs based on mask representation and generated by the tiling approach are used. There are three possible ways to integrate the land cover information into land use classification:

i. In the baseline (Baseline-JO in Tab. 5.9), land cover posteriors are stacked with image data to form multi-band data. This corresponds to the input configuration Mask-T-Full in Tab. 5.7, and the corresponding patches serve as input for training and testing *LuNet-lite-JO.*
ii. Land cover labels (i.e. single band data) are concatenated to the image data (i.e. the input configuration Mask-T-LabelLC in Tab. 5.8). The corresponding patches serve as input for training and testing *LuNet-lite-JO,* corresponding to the set of experiments called Inves-LC in Tab. 5.10. In this case, the information about the uncertainty of each land cover class is lost, which may have a negative impact on the land use classification.
iii. *LuNet-lite-JO* is separately trained and tested using the patches having no land cover information, corresponding to the input configuration Mask-T-NoLC in Tab. 5.8, and with patches having only land cover posteriors, corresponding to the input configuration Mask-T-ProbLC. Both correspond to the experimental set called Inves-LC in Tab. 5.10. Thus, there are two network variants based on *LuNet-lite-JO.* Finally, the results achieved by both network variants are combined in an ensemble (cf. Section 4.3.5). This experiment is referred to as ENS-T-LC in Tab. 5.10.

Comparing the results obtained in Baseline-JO, Inves-LC and ENS-T-LC for all involved datasets leads to answers to both questions.

5.3.2.9. Impact of the Patch Generation

With respect to the impact of the patch generation approach on land use classification, two questions are to be answered by experiments described in Section 6.2.6:

a. Does the scaling approach improve the classification?
b. Does the multi-scale approach improve the classification of small objects?

The experiments of the experiment set called Inves-Scaling in Tab. 5.10 are conducted, using the patches of the input configurations Mask-S-NoLC and Mask-S-ProbLC. For both input configurations, two network variants based on *LuNet-lite-JO* are created, respectively. The results obtained by both network variants are combined with the results achieved by the other two network variants using the input configurations Mask-T-NoLC and Mask-T-ProbLC presented in Section 5.3.2.8 in an ensemble (cf. Section 4.3.5). This experiment is referred to as ENS-TS-LC, which combines results achieved by four network variants. Comparing the results of ENS-T-LC and ENS-TS-LC leads to the answer for question (a).

To answer question (b), the experiments of the experiment set called Inves-ComTS are conducted, which require the input configurations of Mask-Com-NoLC and Mask-Com-ProbLC as input. Again, two network variants based on *LuNet-lite-JO* are trained and tested, corresponding to the two input configurations, respectively. Finally, the results achieved by the two network variants are combined in an ensemble, referred to as ENS-ComTS-LC. The comparison between the results of ENS-T-LC and ENS-ComTS-LC leads to answer of this question.

5.3.2.10. Evaluation on all Datasets

The evaluation on all datasets is performed by using the results delivered by ENS-TS-LC (the reasons for using these results are given in Section 6.2.7). The questions to be answered in Section 6.2.7 are:

a. How does the number of classes to be discerned affect the performance?
b. How does the object size affect the classification?

Question (a) is answered by analyzing the classification results at different semantic levels for each dataset, and question (b) will be answered by analyzing the relationship between the classification result and the object size.

5.3.2.11. Combining Datasets

Similarly to land cover classification, datasets are also combined in land use classification to assess whether enlarging the training datasets could lead to a positive contribution to the classification performance or not. The scientific question to answered by experiments in Section 6.2.8 is:

- Does training on a combined dataset deliver better results than training on each individual dataset?

The Hameln and Schleswig datasets are used for combination. The experiments of the experiment set called ComHS-LU in Tab. 5.10 are conducted. The data corresponding to the input configurations Mask-T-NoLC, Mask-T-ProbLC, Mask-S-NoLC and Mask-S-ProbLC are combined, respectively. Subsequently, four network variants based on *LuNet-lite-JO* are trained and tested using the patches of the corresponding input configuration as input.

All achieved results by the four network variants are combined in an ensemble, referred to as ENS-TS-LC-ComHS. Comparing the results of ENS-TS-LC-ComHS and ENS-TS-LC in Hameln and Schleswig leads to an answer to this question.

6. Experiments

In this chapter all experiments mentioned in chapter 5 are described, and the obtained results are evaluated. The chapter consists of two parts: the first part presents the evaluation of the proposed method for land cover classification, consisting of the experiments mentioned in Section 5.3.1; the second part gives the evaluation of the proposed method for land use classification, consisting of the experiments mentioned in Section 5.3.2.

6.1. Evaluation of Land Cover Classification

6.1.1. Prediction Variability of *FuseNet-lite*

To answer the scientific question raised in Section 5.3.1.3, *FuseNet-lite* is trained ten times on the Hameln and Vaihingen datasets separately, and the ten predictions on each test dataset are used to compute the means and standard deviations of OA and the mean F1 scores. The settings of the hyperparameters are presented in Tab. 5.5.

Fig. 6.1 presents the mean and standard deviations of OA and the mean F1 scores achieved on the test data of both datasets. The test OA is about 90% in Hameln and 88% in Vaihingen, and the standard deviation is about 0.1% for both datasets. This low number indicates that the predictions made by *FuseNet-lite* are very stable in terms of OA. The standard deviation of the mean F1 scores is about 1.3% in Hameln and 0.5% in Vaihingen. Compared to the standard deviations of OA, the larger numerical values are mainly due to the imbalanced class distribution in both datasets. In Hameln, the classes *car* and *clutter* have much fewer training samples (about 1.2% and 0.6% pixels in the training dataset, respectively) in comparison with other classes such as *building* and *sealed area*, so that they are underrepresented. A similar problem is also found in Vaihingen, where the classes *car* and *clutter* only correspond to 1.1% and 0.9% of the whole training samples, respectively. Compared to the deviations of other well-represented classes in terms of F1 score, their standard deviations are larger, which also affect the standard deviations of the mean F1 scores (see Fig. 6.2). Prominently, the standard deviation of class *clutter* is much larger than those of other classes in both sites; the number is about 2.5% in Vaihingen and about 10% in Hameln. Because objects belonging to the class *clutter* are very heterogeneous, they are hard to differentiate even for humans.

In conclusion, the answer to the question raised in Section 5.3.1.3 is that the variability of the predictions made by *FuseNet-lite* is low, indicated by a low standard deviation of about $\sigma_{OA}^{LC} = 0.1\%$ in terms of OA and a standard deviation of about $\sigma_{mF1}^{LC} = 1\%$ in terms of mean F1 score.

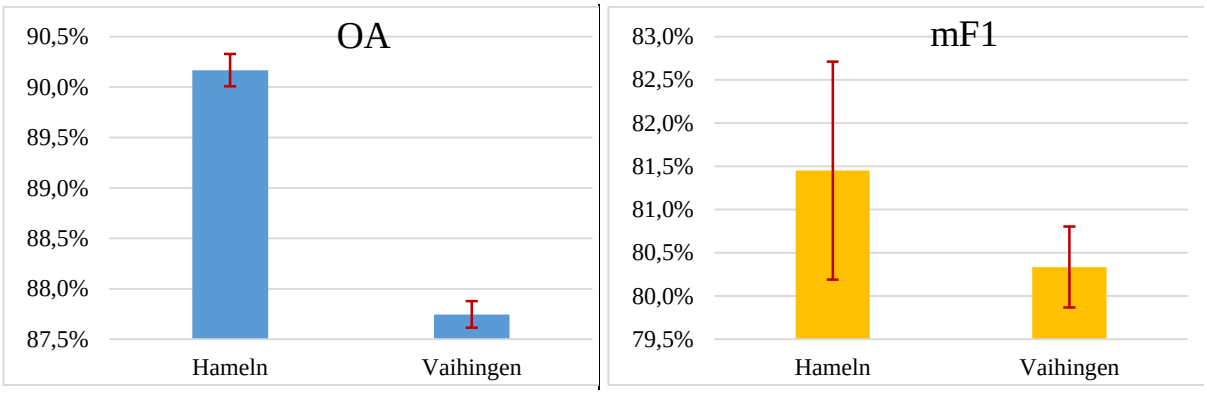


Figure 6.1: OA (left) and mF1 scores (right) with their standard deviations (dark red) in Hameln and Vaihingen, based on ten predictions made by *FuseNet-lite*.

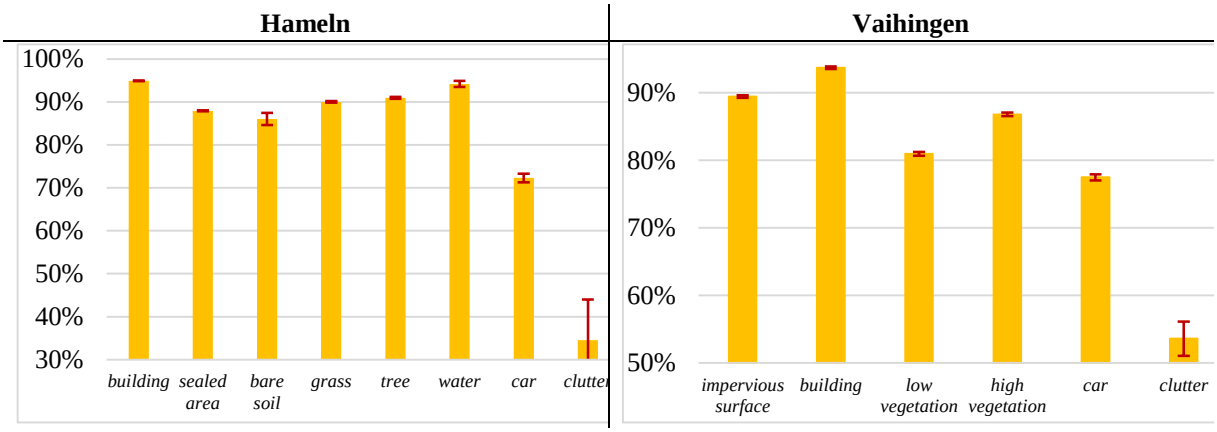


Figure 6.2: F1 scores and the corresponding standard deviations (dark red) of each class in Hameln (left) and Vaihingen (right).

6.1.2. Investigations of the Hyperparameter Settings

For the purpose of investigating the impact of hyperparameters on the CNN, the Hameln and Vaihingen datasets are used again. The settings of the experiments are described in section 5.3.1.4. Three hyperparameters, the base learning rate η, the mini batch size Ω and the weight of the penalty term for the focal loss (eq. 4.5) γ, are analyzed.

6.1.2.1. Base Learning Rate

Theoretically, the base learning rate η influences how well and how fast the training procedure converges. A proper base learning rate leads the model to converge quickly whereas an improper one

would make the training process diverge or lead to a slow convergence (Goodfellow et al. 2016). In the latter case, a large training error is expected. Thus, it is meaningful to know what is a proper value for the base learning rate. While investigating η, the values of mini batch size and the penalty term in eq. 4.5 are set to $\Omega = 10$ and $\gamma = 1$ (cf. Tab. 5.5). There are four candidate values for η presented in Tab. 5.6. Fig. 6.3 shows the training and validation errors for different values of η as a function of the training epoch in Hameln and Vaihingen, and Fig. 6.4 shows the OA and mean F1 scores in both test sets. It has to be noted that in this thesis, the training and validation errors are defined as $1 - OA$ on the training and validation sets, respectively.

On the basis of Fig. 6.3, two observations can be made. First, for the values of $\eta = 0.1, 0.01, 0.001$, a higher value of η leads to smaller training and validation errors as the training epoch increases in both datasets. However, using $\eta = 1.0$, the training and validation errors are much larger than the corresponding ones using $\eta = 0.1$. This fact reveals that using too large or too small base learning rate may prevent the training from achieving a good convergence. Second, for the settings of $\eta = 0.1, 0.01, 0.001$, the difference between validation error and training error gets smaller as η decreases.

Switching the focus on the performances in the test sets of both datasets (cf. Fig. 6.4), for the settings of $\eta = 0.1, 0.01, 0.001$, as the base learning rate decreases, the classification performance (both OA and the mean F1 scores) decreases as well. For instance, comparing the OAs achieved for $\eta = 0.1$ and $\eta = 0.001$, the former one outperforms the latter one by about 3% in Hameln and about 2% in Vaihingen. The differences in terms of the mean F1 score achieved for $\eta = 0.1$ and $\eta = 0.001$ are even larger: about 10% in both datasets. Furthermore, the corresponding mean F1 scores achieved for $\eta = 1.0$ are lower than those achieved for $\eta = 0.1$ in both datasets, and the OA for $\eta = 1.0$ is also lower than the one for $\eta = 0.1$ in Hameln, but slightly larger in Vaihingen.

The results show that if the base learning rate is too large or too small, it will have negative impact on the classification, which is the answer to the question (a) raised in Section 5.3.1.4. For $\eta = 1.0$, the difference between validation error and training error is small, but it takes longer for a model to converge. For instance, in Vaihingen there is a larger oscillation of the training errors than for those for other values of η before epoch 15. When using $\eta = 0.001$, there is small difference (about 0.5%) between validation error and training error, but with very large training errors, indicating that the trained model probably under-fits to the training data.

6.1.2.2. Mini Batch Size

As presented in Tab. 5.6, the investigated values of the minibatch size Ω are 3, 5, 10, 15, and during the investigation of Ω, the values of the base learning rate and the weight of the penalty term in eq. 4.5 are set to $\eta = 0.1$ and $\gamma = 1$, respectively. For different values of Ω, Fig. 6.5 presents the training and validation errors as a function of training epochs, and Fig. 6.6 shows the results on the test sets (OA and the mean F1 scores).

In Fig. 6.5, a tendency can be observed: as Ω increases, the training errors decrease in most cases. For instance, in Hameln the training error at the final epoch 30 is about 11% for $\Omega = 3$, whereas it is

about 10% for $\Omega = 20$. In Vaihingen, the corresponding numbers are about 10% for $\Omega = 3$ and about 7.5% for $\Omega = 20$.

The validation errors do not vary much for different values of Ω after the 15th training epoch. The differences between the maximum and minimum validation errors are less than 1% in both datasets. Looking at the differences between validation error and training error, they are small if Ω is small. Switching the focus on the test results in Fig. 6.6, in Hameln, there is a difference between the OAs for different values of Ω. The maximum OA of about 90% occurs for $\Omega = 10$ and $\Omega = 15$, whereas the minimum OA of about 89% occurs for $\Omega = 3$, thus, a difference of 1.0% is observed (about $10 \cdot \sigma_{OA}^{LC}$) In Vaihingen the corresponding difference between the maximum OA and the minimum OA is about 0.5% (about $5 \cdot \sigma_{OA}^{LC}$). However, the absolute numerical differences between the maximum mean F1 score ($\Omega = 15$) and the minimum mean F1 score ($\Omega = 3$) are much larger, which are 7.2% in Hameln and 7.5% in Vaihingen, corresponding to about $7 \cdot \sigma_{mF1}^{LC}$.

From the results described above, one can see that the mini batch size does affect the performance of *FuseNet-lite*. Larger mini batch sizes deliver better results in most cases, which is the answer to the question (a) raised in Section 5.3.1.4. It has to be noted that there might be a dependency between the mini batch size and the base learning rate, which is not investigate in this thesis.

6.1.2.3. The Weight of the Penalty Term in the Focal Loss

The primary goal of applying the focal loss (eq. 4.5) is to mitigate the problem of an imbalanced class distribution to a certain degree. Tab. 5.6 shows that there are five investigated values for γ, which are 0, 0.5, 1, 2 and 5. While investigating the influence of γ, the values of the base learning rate and the mini batch size are set to $\eta = 0.1$ and $\Omega = 10$, respectively. It has to be noted that the setting $\gamma = 0$ corresponds to training based on the standard cross entropy loss.

Fig. 6.7 presents the training and validation errors for different values of γ as a function of the training epoch in both datasets. In Hameln, the training process converges similarly for different values of γ. In Vaihingen, this is similar except for $\gamma = 5$. It seems that a large value of γ prevents the model from overfitting, achieving a relatively smaller difference between training and validation errors. From a theoretic point of view, a large value of γ leads to small magnitudes of losses, making the gradients for weight updating small. Thus, the weights are updated by small values in each iteration, which has an impact on the convergence speed when training a CNN. This is also observed in Fig. 6.7.

The classification results for the test sets of both datasets are presented in Fig. 6.8. Looking at the OAs, for $\gamma = 5$ it is worse than the one achieved when using other values for γ in both datasets: it is about 0.3% smaller than the best OA of 90.4% in Hameln whereas it is about 0.2% smaller than the best OA of 87.8% in Vaihingen. Turning the focus on the mean F1 scores, the mean F1 scores for $\gamma = 5$ are also not the best ones, and in Hameln it is the lowest mean F1 score. Compared to the best one achieved for $\gamma = 0$ it is smaller by 2.4%. In Vaihingen the mean F1 score for $\gamma = 5$ is about 0.9% smaller than the best one achieved for $\gamma = 2$.

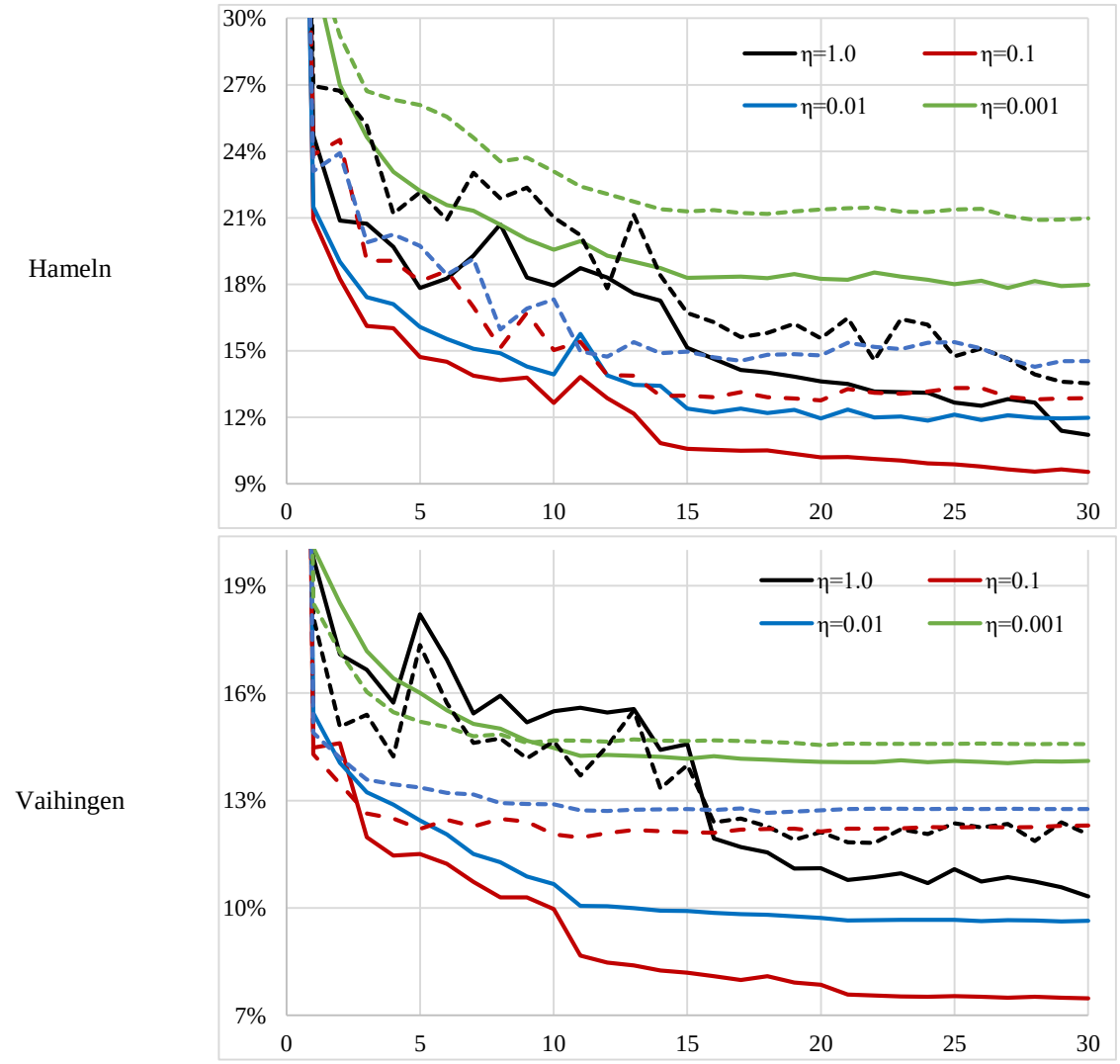


Fig. 6.3: Training and validation errors vs. training epochs for different values of η in Hameln and Vaihingen. X-axis: training epoch; Y-axis: training errors (solid lines) and validation errors (dashed lines).

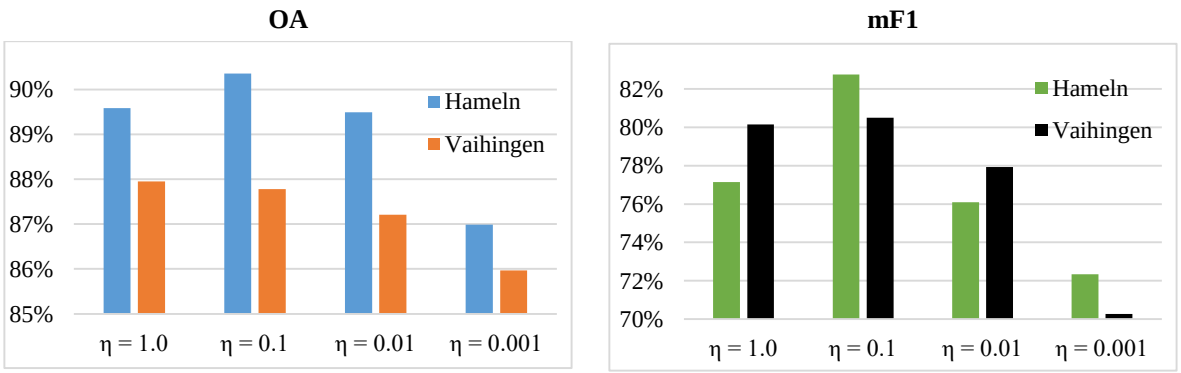


Fig. 6.4: OA and mF1 scores in the test sets of Hameln and Vaihingen for different values of η.

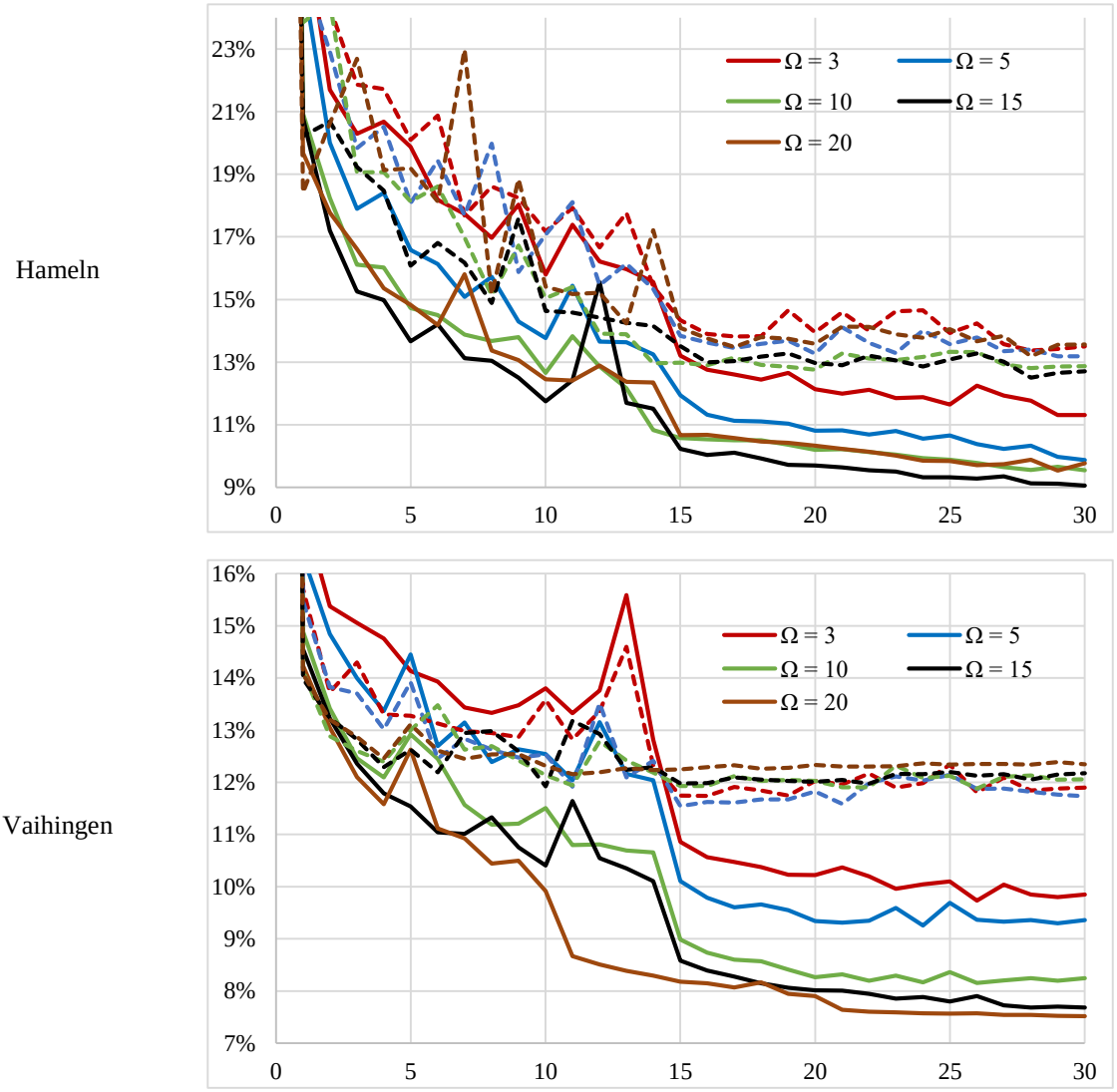


Figure 6.5: Training and validation errors vs. training epochs for different values of Ω in Hameln and Vaihingen. X-axis: training epoch; Y-axis: training errors (solid lines) and validation errors (dashed lines).

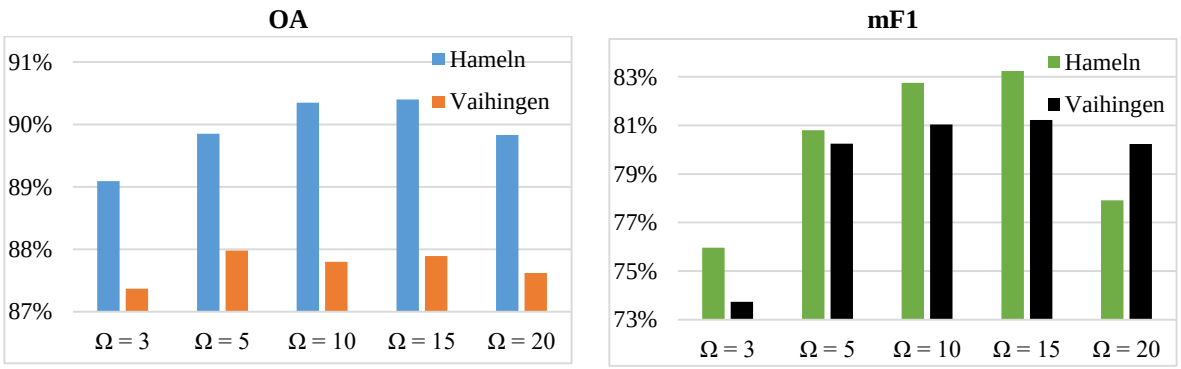


Figure 6.6: OA and mF1 scores in the test sets of Hameln and Vaihingen for different values of Ω.

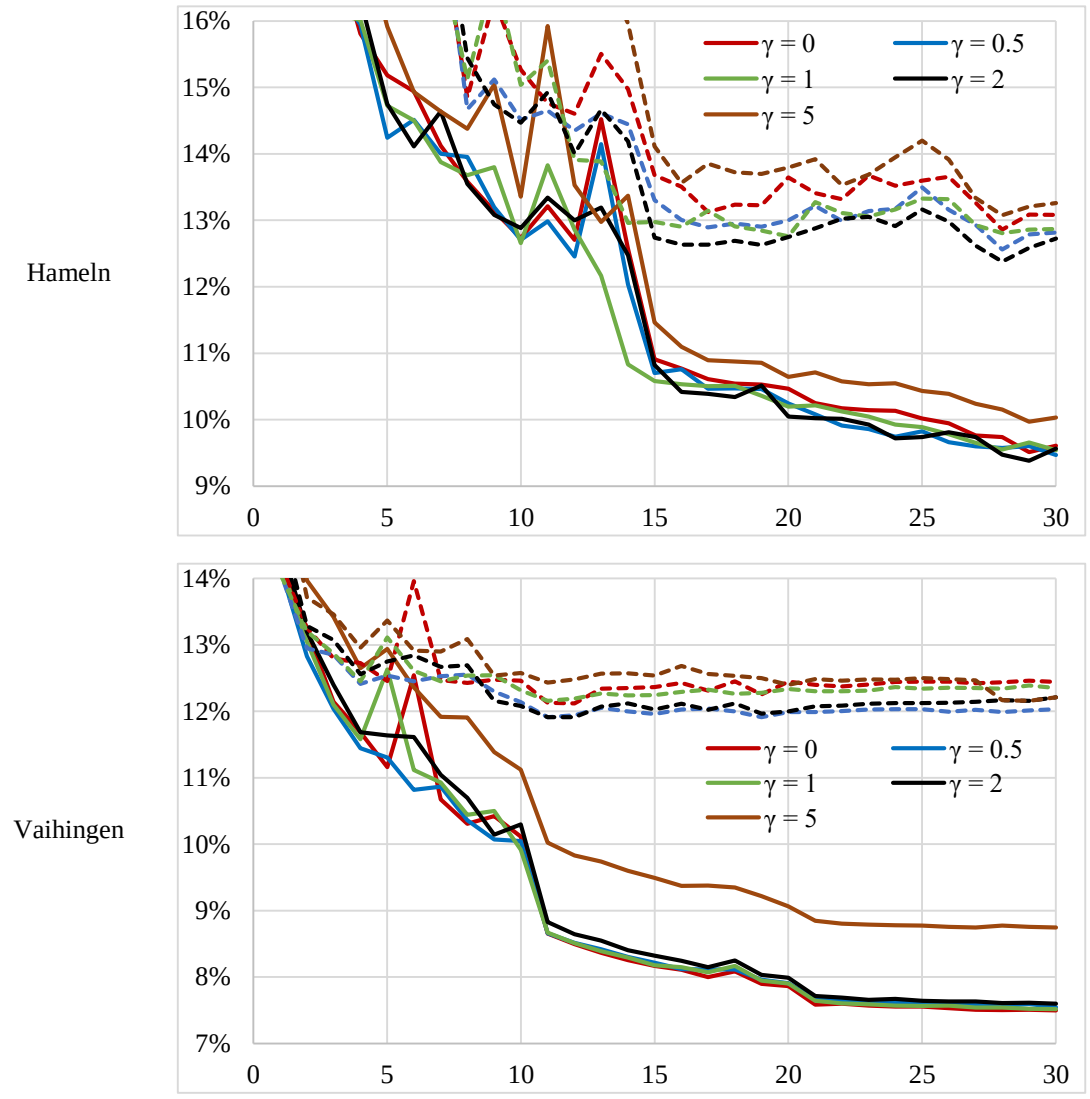


Figure 6.7: Training and validation errors vs. training epochs for different values of γ in Hameln and Vaihingen. X-axis: training epoch; Y-axis: training errors (solid lines) and validation errors (dashed lines).

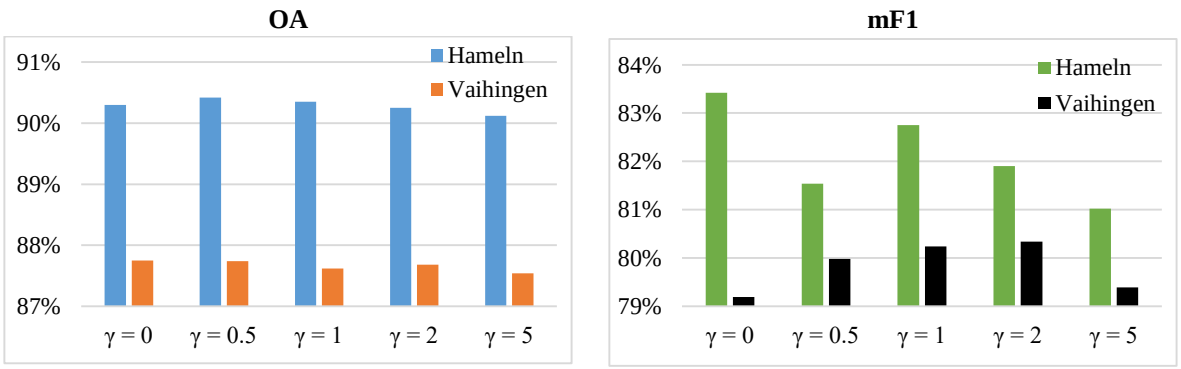


Figure 6.8: OA and mF1 scores in the test sets of Hameln and Vaihingen for different values of γ.

Nonetheless, $\gamma = 0$, which corresponds to training based on the standard cross entropy loss, delivers good results in terms of OA in both datasets. In Hameln, the maximum OA is delivered by using the $\gamma = 0.5$, and the difference to the OA delivered by using $\gamma = 0$ is 0.1%, corresponding to $1 \cdot \sigma_{OA}^{LC}$. In Vaihingen, the maximum OA is delivered by using $\gamma = 0$, and is about 0.1% larger than the second best one delivered by using $\gamma = 0.5$. Switching on the mean F1 scores: in Hameln, using the value of $\gamma = 0$ obtains the maximum mean F1 which is about 0.6% larger than the second best one by using $\gamma = 1$. This absolute difference is about $0.6 \cdot \sigma_{mF1}^{LC}$. In Vaihingen, the maximum mean F1 score is delivered by using $\gamma = 2$, and comparing to the one of using $\gamma = 0$, the difference is about 1.2%, corresponding to $1.2 \cdot \sigma_{mF1}^{LC}$. With these observations, it seems that using a value of $\gamma > 0$ has a positive effect on the classification performance, but to a limited degree.

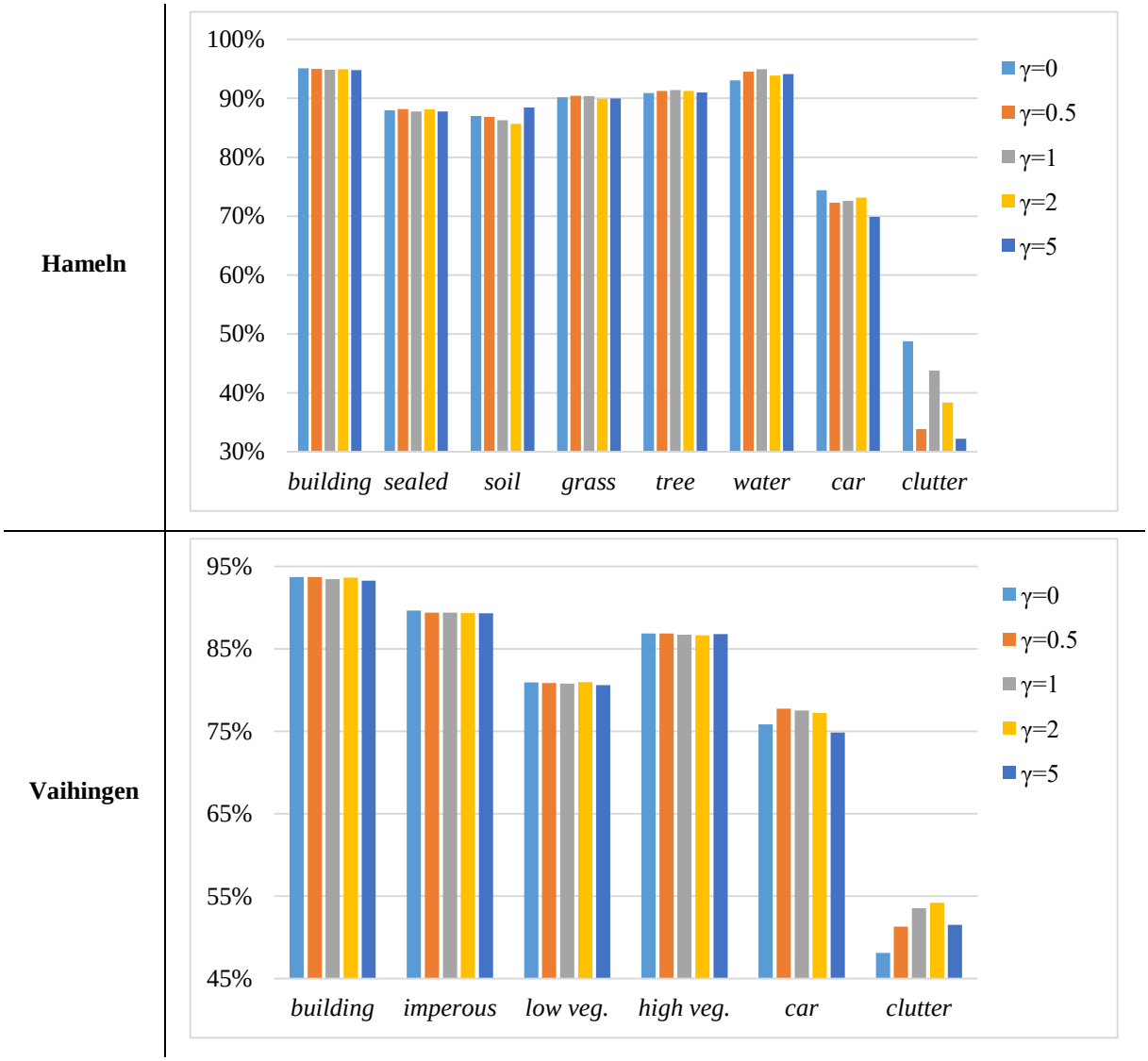


Figure 6.9: F1 scores of all individual classes as a function of the values of γ in Hameln and Vaihingen.

This fact contradicts to the findings of Lin et al. (2017) a little, where the focal loss performs much better than standard cross entropy loss in a binary classification. The reason for the contradiction is that the class distribution of the dataset they investigated (two classes: object and non-object) is even more extremely imbalanced than the one in Hameln and Vaihingen. The imbalance degree in Hameln and Vaihingen is $\rho_{im} = 44.7$ and $\rho_{im} = 41.5$, respectively, whereas the one in (Lin et al., 2017) is $\rho_{im} \approx 1000$. Switching the focus to the mean F1 scores, there are improvements (> 1%) in Vaihingen for $\gamma = 1$ and $\gamma = 2$ in comparison with results delivered by the standard cross entropy loss, but in Hameln the standard cross entropy loss delivers still better results than other values for $\gamma > 0$.

Looking at the F1 scores of all individual classes for different values of γ in both datasets (cf. Fig. 6.9), the results achieved by the value of $\gamma = 0$ have the best F1 scores for the underrepresented classes *car* and *clutter* in Hameln. In Vaihingen, a value of $\gamma > 0$ delivers better F1 scores for the underrepresented classes *car* and *clutter* than the value of $\gamma = 0$ by up to +6.1%.

In conclusion, the weight of the penalty term does impact the classification performance in Hameln and Vaihingen and a value of $\gamma > 0$ has a positive effect to a limited degree, which is the answer to the question (a) raised in Section 5.3.1.4. One possible reason is that the class distributions are not extremely imbalanced. The proposed focal loss does not outperform standard cross entropy loss in some cases, which is the answer to the question (b) raised in Section 5.3.1.4. Although there are better F1 scores for the underrepresented classes *car* and *clutter* using the values of $\gamma > 0$ in Vaihingen, it is not the case in Hameln. A deterministic conclusion about, whether the proposed focal loss could improve the classification of underrepresented classes in this thesis or not, cannot be made. Further investigations are required by using datasets having different imbalance degrees of class distribution. This is the answer to the question (b).

6.1.3. Effectiveness of the learnable Skip-Connections

To answer the questions raised in section 5.3.1.5, all experiments referred to as Learn-Skip, Add-Skip and No-Skip in Tab. 5.4 are conducted. It has to be noted that the datasets of Hameln, Schleswig and MV are used for conducting experiments with 3-fold cross validation. Tabs. 6.1 – 6.3 present overviews of the test results based on the evaluation using the full reference for all datasets. Tab. 6.4 presents the results for pixels near object boundaries and outside object boundaries by using both the full and the eroded references. Tab. 6.4 also presents the percentages of pixels near object boundaries and outside boundaries for all datasets.

On the basis of Tabs. 6.1 – 6.3, there are two conclusions which can be drawn. First, network variants with skip-connections (elementwise addition or learnable) outperform the network variant without skip-connections by up to +2.7% in terms of OA (in Potsdam) and +8.2% in terms of mean F1 score (in Schleswig). This conclusion serves as the answer for the question (a) raised in Section 5.3.1.5; second, the network variant with learnable skip-connections outperforms the variant based on elementwise addition with respect to the OA and the mean F1 score over all datasets. The improvements are up to +0.8% for OA (in MV) and +1.4% for the mean F1 score (in Schleswig), and the corresponding improvements in terms of standard deviation are $8 \cdot \sigma_{OA}^{LC}$ and $1.4 \cdot \sigma_{mF1}^{LC}$. Looking at the individual

classes in each dataset, it is found that the F1 scores of most classes benefit from the skip-connections, and the minority classes, which have few samples, are improved by a large margin. In Hameln. compared to the network variant without skip-connections, the improvements of the F1 scores for the class *car* are +12.5% and 13% by the variants with elementwise addition and learnable skip-connections, respectively. The corresponding improvements in *clutter* are +26.1% and +31.2%, which is even larger. In Schleswig, similar improvements are observed for the F1 scores of the class car (+31.8% and +39.9%). There are also larger improvements of minority classes in the other datasets (Vaihingen, Potsdam and MV).

Comparing skip-connections by elementwise addition to the proposed learnable one, the F1 scores of most classes are improved by the latter, and again the largest improvements are achieved for the minority classes. The maximum improvements of F1 scores over all datasets are +5.1% for *clutter* in Hameln, +9.1% for *car* in Schleswig, +6.5% for *clutter* in Vaihingen and +3.2% for *clutter* in MV. The answer to the question (b) raised in Section 5.3.1.5 is that the proposed learnable skip-connections perform better than the elementwise addition one, and in particular for underrepresented classes, although the improvement of OA is small.

To answer the question (c) raised in Section 5.3.1.5, the results in Tab. 6.4 are analyzed. One can see that less than 10% pixels are near object boundaries for all datasets. Looking at the results (OAs and the mean F1 scores) for the pixels near object boundaries, compared to the variant without skip-connections, the variant with elementwise addition skip-connections improves the classification by +1.1% (in MV) to +4.1% (in Hameln) in terms of OA and by +0.9% (in MV) to +5.8% (in Hameln and Schleswig) in terms of mean F1 score. Again, the network variant with learnable skip-connections outperforms the one with elementwise addition in most cases, though only by a small margin. Fig. 6.10 shows examples delivered by the three network variants in all datasets. One can find that skip-connections help improving the object boundary delineation. Furthermore, the classification of the pixels outside object boundaries over all datasets is also improved by the network variants with skip-connections (up to +2.7% for OA in Potsdam and +7.3% for mean F1 score in Schleswig).

Experiment Set	**F1 [%]**								OA [%]	mF1 [%]
	build.	*seal.*	*soil*	*grass*	*tree*	*water*	*car*	*clutter*		
Hameln										
No-Skip	92.5	84.3	81.8	87.3	88.3	93.3	59.2	11.1	87.3	74.7
Add-Skip	93.9	86.8	**83.6**	88.0	88.6	94.9	71.7	37.2	88.6	80.6
Learn-Skip	**94.2**	**87.0**	82.8	**88.2**	**88.7**	**95.2**	**72.2**	**42.3**	**88.8**	**81.3**
Schleswig										
No-Skip	89.2	80.0	77.9	81.9	90.4	89.0	20.9	9.1	84.6	67.3
Add-Skip	90.6	82.6	**78.3**	**83.5**	91.2	90.0	51.7	**25.0**	86.4	74.1
Learn-Skip	**92.2**	**84.2**	77.6	83.1	**91.3**	**91.3**	**60.8**	23.7	**86.5**	**75.5**

Table 6.1: Overview of test results achieved in three sets of experiment (cf. Tab. 5.4) in Hameln and Schleswig. All results rely on an evaluation based on the full reference.

Experiment Set	F1 [%]						OA [%]	mF1 [%]
	build.	*imper.*	*low veg.*	*high veg.*	*car*	*clutter*		
Vaihingen								
No-Skip	93.0	88.3	80.0	86.7	75.3	46.2	87.0	78.3
Add-Skip	**93.7**	89.6	81.3	**87.0**	76.2	47.9	**87.9**	79.3
Learn-Skip	93.6	**89.7**	**81.4**	86.8	**77.5**	**54.4**	**87.9**	**80.6**
Potsdam								
No-Skip	90.3	87.4	83.3	84.7	88.5	46.5	85.4	80.1
Add-Skip	**95.7**	90.4	84.4	85.1	89.7	51.8	88.0	82.8
Learn-Skip	95.5	**90.6**	**84.6**	**85.2**	**90.1**	**52.6**	**88.1**	**83.1**

Table 6.2: Overview of test results achieved in three sets of experiment (cf. Tab. 5.4) in Vaihingen and Potsdam. All results rely on an evaluation based on the full reference.

Experiment Set	F1 [%]										OA [%]	mF1 [%]
	build.	*seal.*	*soil*	*grass*	*tree*	*water*	*un. road*	*crops*	*railway*	*clutter*		
No-Skip	88.9	80.5	**79.2**	79.2	88.0	95.0	41.1	84.6	92.7	60.1	83.3	79.2
Add-Skip	90.5	80.6	78.2	82.2	88.4	94.9	43.3	84.8	93.4	62.1	83.7	79.8
Learn-Skip	**91.5**	**82.3**	78.9	**83.0**	**88.9**	**95.1**	**46.0**	**85.5**	**93.6**	**65.3**	**84.5**	**81.1**

Table 6.3: Overview of test results achieved in three sets of experiment (cf. Tab. 5.4) in MV. All results rely on an evaluation based on the full reference.

Dataset	Experiment Set	Near Object Boundaries			Outside Object Boundaries		
		OA [%]	mF1 [%]	#Pixels [%]	OA [%]	mF1 [%]	#Pixels [%]
Hameln	No-Skip	52.2	44.5	8.8	90.6	78.2	91.2
	Add-Skip	**56.3**	**50.3**		91.7	84.2	
	Learn-Skip	**56.3**	50.2		**91.8**	**85.0**	
Schleswig	No-Skip	48.5	38.7	9.0	88.1	71.0	91.0
	Add-Skip	52.4	44.5		**89.7**	78.1	
	Learn-Skip	**53.9**	**47.3**		**89.7**	**78.3**	
MV	No-Skip	49.7	47.1	5.1	85.1	81.0	94.9
	Add-Skip	50.8	48.0		85.5	81.5	
	Learn-Skip	**51.5**	**48.5**		**86.3**	**82.8**	
Vaihingen	No-Skip	57.5	51.6	9.1	89.9	81.6	90.9
	Add-Skip	58.9	54.1		90.8	82.5	
	Learn-Skip	**59.1**	**54.2**		**90.9**	**83.7**	
Potsdam	No-Skip	57.1	54.7	7.2	87.6	82.9	92.8
	Add-Skip	59.4	58.1		90.2	85.5	
	Learn-Skip	**59.7**	**58.2**		**90.3**	**85.7**	

Table 6.4: Test results in three sets of experiments (cf. Tab. 5.4) for all datasets. The evaluation is based on the full and the eroded references. The percentage of pixels near and outside boundary areas are shown as well.

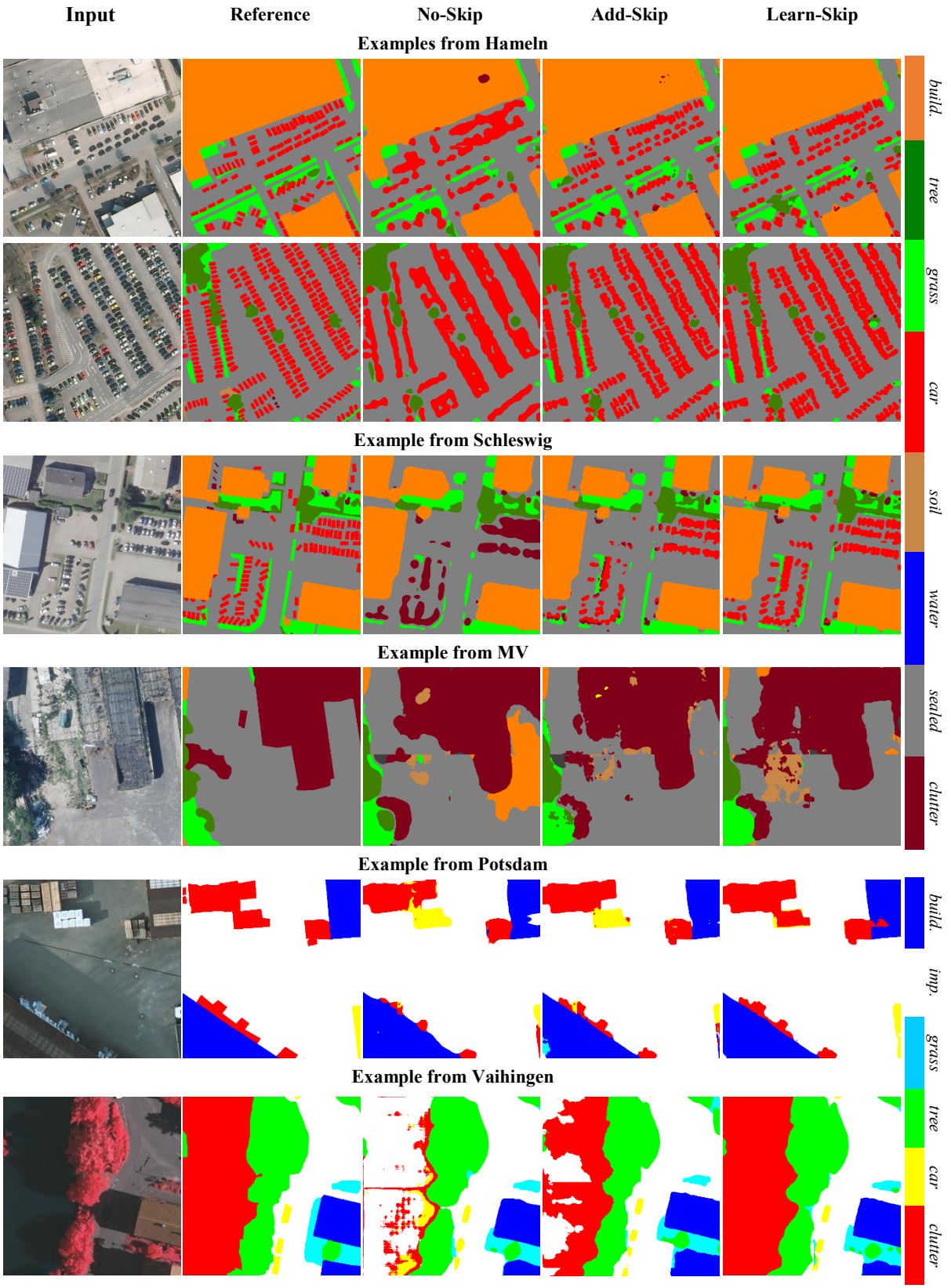


Figure 6.10: Some examples from three sets of experiments for land cover classification (cf. Tab. 5.4) for all datasets. In Hameln, Schleswig and MV, the first column shows RGB images whereas CIR images are shown for Vaihingen and Potsdam.

6.1.4. Evaluation on the individual Datasets

In this section, the results of all experiments referred to as Learn-Skip in Tab. 5.4 are evaluated. Section 6.1.4.1 presents the results for Hameln, Schleswig and MV, and Section 6.1.4.2 presents the results for Vaihingen and Potsdam. Section 6.1.4.3 gives answers to the questions raised in Section 5.4.1.4.

6.1.4.1. Hameln, Schleswig and MV

Tab. 6.1 presents the evaluation results for Hameln and Schleswig, and Tab. 6.3 presents the results for MV. The proposed network *FuseNet-lite* shows quite good performance in these three datasets with OA between 84% and 89%. Taking a closer look at the classification results in Hameln and Schleswig, except for *car* and *clutter*, which are underrepresented, all classes are identified quite well, with F1 scores of more than 75%. In Schleswig, pixels belonging to class *soil* are not identified well, although there are many training samples for this class (about 8% samples in the entire dataset). *Soil* is erroneously classified as *grass*, and according to the confusion matrix about 24% *soil* samples are classified as *grass*. Fig. 6.11 shows some classification results from urban and rural areas in Hameln and Schleswig. From the rural example in Schleswig, it is clear to see that the *soil* areas in the rural area in Schleswig should be labeled as *grass*. In Hameln, such labeling errors do not occur. The urban areas in both sites are classified very well (with very high F1 scores for urban classes such as *building* or *sealed area*). In MV, there are ten classes to be discerned. Generally speaking, the classification results are quite good with an OA of about 85% and a mean F1 score of about 81%. Compared to other classes, the class *unpaved road* is very difficult to identify correctly, which is indicated by a very low F1 score of 46%. Only about 2% of the training samples belong to that class, which may be one reason for the low performance. In addition, *unpaved road* can easily be confused with *sealed area* due to the similar appearance of these classes (about 24% of the test samples of *unpaved road* are classified as *sealed area*). The examples from MV shown in Fig. 6.11 show this similarity. It has to be noted that for the land use classification, land cover information obtained by *FuseNet-lite* in each individual dataset (i.e. Hameln, Schleswig and MV) is used.

6.1.4.2. Vaihingen and Potsdam

These two datasets are published in the ISPRS 2D Labeling Challenge with the purpose of evaluating advanced methods for land cover classification. To investigate the performance of *FuseNet-lite*, classification and evaluation on the two public datasets is performed. In the first step, the evaluation is performed on the basis of the full reference, and Tab. 6.2 presents the numerical results on the test data.

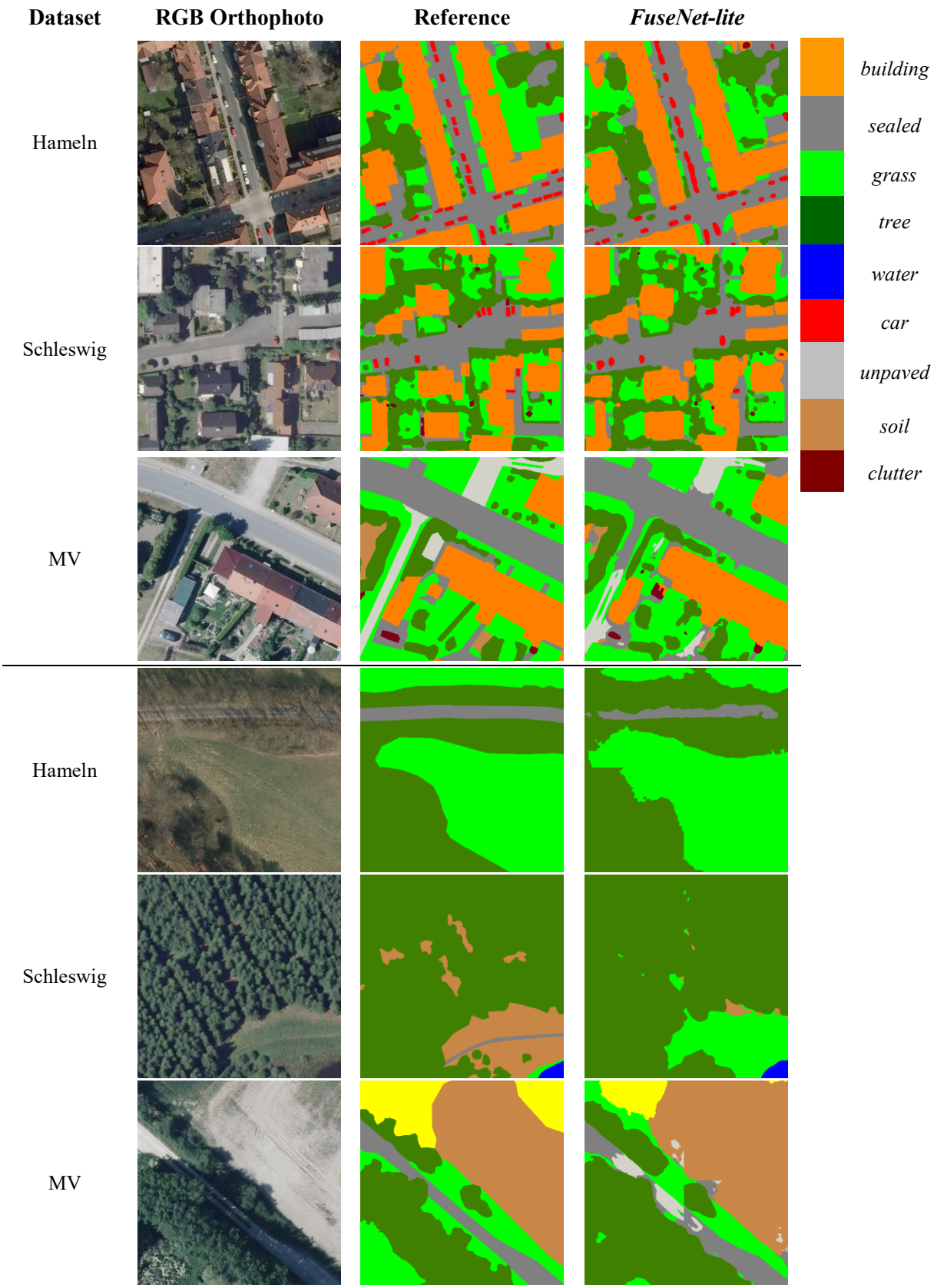


Figure 6.11: Visual examples of the results delivered by *FuseNet-lite* in urban and rural areas in Hameln, Schleswig and MV.

FuseNet-lite performs very well in both sites, with an OA of about 88% and a mean F1 score of more than 80%. Both evaluation metrics are quite close to those in Hameln and Schleswig, indicating a consistently good performance of the proposed method. Looking on the classes individually, most of them are differentiated very well. For instance, the F1 scores of the classes *building*, *impervious surface* and *high vegetation* (trees in most cases) are larger than 85% in both sites. In Vaihingen, compared to well-represented classes, the class *car* is relatively hard to be correctly classified (F1 = 77.5%), mainly due to its small amount of training samples (about 1.2% samples for training); in both sites, the identification of class *clutter* is problematic, which is likely to be the reason that it contains too many heterogeneous objects which are quite difficult to differentiate even for humans.

To assess the difference between *FuseNet-lite* and the state-of-the-art methods, comparisons with other methods listed in the scoreboard of the ISPRS 2D labeling challenge are made. For that purpose, the eroded reference is used for evaluation in both sites. The comparison considers the scoreboard of the ISPRS benchmark (Wegner et al., 2017), but also other recent publications (the scoreboard has not been updated anymore since 2018). Tab. 6.5 provides an overview of all numerical results.

Network	**Input**	**#Param.**	**F1 [%]**					**mF1 [%]**	**OA [%]**
			build.	*imp. surf.*	*low veg.*	*high veg.*	*car*		
Vaihingen									
HUSTW (Sun et al., 2019)	CIR, nDSM	$1.3 \cdot 10^7$	96.1	93.3	86.4	90.8	74.6	88.2	91.6
NLPR3	-	-	95.6	93.0	85.6	90.3	84.5	89.8	91.2
EaNet (Zhang et al., 2020)	CIR	$> 6 \cdot 10^7$	96.2	93.4	85.6	90.5	88.3	90.8	91.2
CASIA2 (Liu et al., 2018)	CIR	$> 6 \cdot 10^7$	96.0	93.2	84.7	89.9	86.7	90.1	91.1
DDCM (Liu et al., 2020)	CIR	$> 2 \cdot 10^7$	95.3	92.7	83.3	89.9	82.4	88.7	90.4
DLR_9 (Marmanis et al., 2018)	CIR, nDSM	$8.1 \cdot 10^8$	95.2	92.4	83.9	89.9	81.2	88.5	90.3
VFuseNet (Audebert et al., 2018)	CIR, nDSM	$> 8 \cdot 10^7$	94.4	91.0	84.5	89.9	86.3	89.2	90.0
FuseNet-lite	CIR, nDSM	$4.6 \cdot 10^5$	95.6	92.4	85.0	90.1	83.5	89.3	90.9
Potsdam									
HUSTW (Sun et al., 2019)	RGB, IR, nDSM	$1.3 \cdot 10^7$	96.7	93.8	88.0	89.0	96.0	92.7	91.6
BAMTL (Wang et al., 2021)	RGB, nDSM	$> 6 \cdot 10^7$	-	-	-	-	-	90.9*	91.3
CASIA2 (Liu et al., 2018)	CIR	$> 6 \cdot 10^7$	97.0	93.3	87.7	88.4	96.2	92.5	91.1
DDCM (Liu et al., 2020)	RGB	$> 2 \cdot 10^7$	96.8	93.3	87.6	89.4	95.0	92.4	91.1
VFuseNet (Audebert et al., 2018)	RGB, IR, nDSM	$> 8 \cdot 10^7$	96.3	92.7	87.3	88.5	95.4	92.0	90.6
FuseNet-lite	CIR, IR, nDSM	$4.6 \cdot 10^5$	96.7	92.6	87.0	87.8	94.9	91.8	90.3

Table 6.5: Results for Vaihingen and Potsdam and comparison to state-of-art methods based on eroded reference. Mean F1 scores with * consider class *clutter* as well. #Param.: number of trainable parameters. For the method described by NLPR3 in the scoreboard, no publication is found. "-": no value is given.

In both test sites, the best OA of 91.6% is delivered by the method described as HUSTW. Although *FuseNet-lite* does not outperform this method, the difference in OA achieved by *FuseNet-lite* is within 1% of the best results. In Vaihingen, the predictions are quite close to the best ones in terms of both OA and mean F1 score. *FuseNet-lite* slightly outperforms HUSTW in terms of the mean F1 score by about 1%, mainly due to the improved recognition of the class *car*. In Potsdam, there is no large difference (-

0.9%) between the mean F1 score of *FuseNet-lite* and the best one, and a difference of - 1.3% between the OA of *FuseNet-lite* and the best one is observed.

Turning the focus on the number of trainable parameters, *FuseNet-lite* is much smaller in terms of the number of parameters than all others by a factor of about 100. All works mentioned in the table are based on encoder-decoder network structures and most of them use an encoder based on ResNet (He et al., 2016). For instance, EaNet, CASIA2 and BAMTL are based on ResNet101; DDCM is based on ResNet50 and VFuseNet is based on ResNet34. The decoder part is symmetric to the encoder. Due to the large number of trainable parameters in ResNet, such methods suffer from having to determine a large number of unknown parameters from the rather limited amount of training data. Generally speaking, *FuseNet-lite* requires only about 1% or even a smaller percentage of trainable parameters than the compared methods, while it still performs on par with state-of-the-art methods. Fig. 6.12 presents some predicted examples from both sites for visualization. Some misclassifications are highlighted by the red rectangles, indicating that *FuseNet-lite* has problem of classifying some challenging areas.

6.1.4.3. Answers to the Questions raised in Section 5.3.1.6

In this section, five individual datasets showing different data characteristics and different class structures were used to evaluate *FuseNet-lite*. In summary, *FuseNet-lite* delivers quite good predictions with an OA in the order of 85% - 90% over all datasets, which gives the answer to the question (a) raised in Section 5.3.1.6. Compared to state-of-the-art methods involving the publicly available datasets Vaihingen and Potsdam, *FuseNet-lite* delivers classification results which are on par with the best methods though slightly worse, which is the answer to the question (b) raised in Section 5.3.1.6.

6.1.5. Training on the combined Datasets

Between different datasets there is a domain gap caused e.g. by different capturing seasons, sensor types or environmental conditions. In this thesis, different datasets are combined despite their domain gap to investigate whether adding data from other domains could positively contribute to the classification or not. Because the number of available training samples of each class is enlarged after combining the datasets, a positive contribution is expected. The considered combination criterion is that the combined datasets must have the same land cover class structure. Thus, the Hameln and Schleswig datasets are combined, and so are the Vaihingen and Potsdam datasets. The experiment sets ComHS-LC and ComVP-LS in Tab. 5.4 were conducted. The results based on training on combined datasets for Hameln and Schleswig and the differences to the results based on individual training are presented in Tab. 6.6, and Tab. 6.7 shows the same information for Vaihingen and Potsdam.

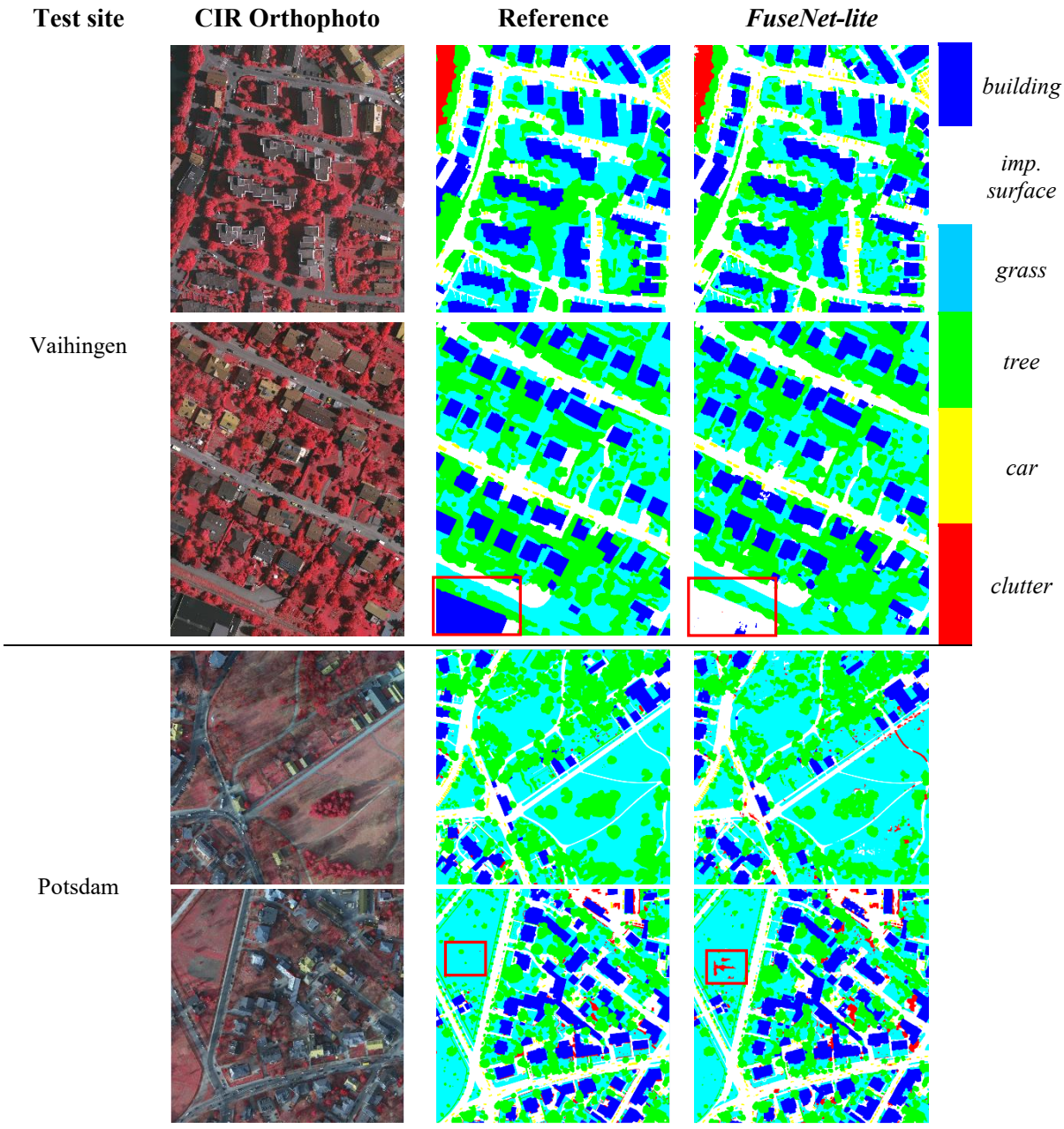


Figure 6.12: Examples of the predictions of *FuseNet-lite* in Vaihingen and Potsdam. Some misclassifications are highlighted by red rectangles.

The classification results in Schleswig are improved by +0.7% and +2% in terms of OA and mean F1 score, respectively. In Hameln, there is only a slight improvement of +0.3% in terms of the mean F1 score and no change of OA. Considering individual classes, in Hameln most classes are slightly better recognized except *grass* and *water*, which have drops of F1 scores of 0.3% and 1.4%, respectively. Similar improvements can be observed in Schleswig, except that the F1 score of class *water* drops by 1.9%. In both sites, the decrease of the F1 score of class *water* in the combined dataset may be due to the different appearance of this class in Hameln and Schleswig. The lower half of Fig. 6.13 shows examples of *water* inside forest areas in both sites. In Schleswig, water pixels are affected by the shadow

of trees, whereas in Hameln they are exposed to the sun. This improvement of the results by joint training are in line with the findings of Kaiser et al. (2017), who found that using a large amount of data could improve the performance of land cover classification. Although there is a large difference in the appearance of vegetation objects (i.e. *grass*, *tree*), their separability is not affected too much after combining the data from the two test sites. The upper half of Fig. 6.13 shows two examples involving trees, indicating that training based on the combined dataset delivers similar results as individual training.

Looking at the results in Vaihingen and Potsdam, training based on the combined dataset delivers almost the same results in terms of OA and mean F1 score as those delivered by training solely on each individual dataset. The combination does not degrade the performance, yet it does not boost it either.

In conclusion, the results delivered by combined datasets do not outperform the individual ones by a large margin, which is the answer to the question raised in Section 5.3.1.7. The potential of improving classification by combining datasets is still shown, but an appropriate strategy may be required to cope with the problem of domain gap, e.g. trying to learn domain-independent representations for each category from different domains. For instance, this could be achieved by adding a regularization on the intermediate feature maps (e.g. the feature maps at the end of the encoder) belonging to the same category can be considered. However, the design of such a strategy is beyond the scope of this thesis.

6.1.6. Discussion

In the context of land cover classification, an encoder-decoder based CNN called *FuseNet-lite* is proposed. It uses a learnable skip-connections to connect the feature maps in the lower layers of the encoder to the ones in the higher layers of the decoder. Based on the results of the experiments using five datasets, the learnable skip-connections slightly outperforms the most frequently used one based on elementwise addition. Furthermore, an extended focal loss for multi-class classification is proposed to mitigate the problem of imbalanced class distribution, by extending the original one proposed in (Lin et al., 2017). The results have shown that the focal loss has a limited positive effect on the classification performance, and the results do not largely outperform the ones delivered by the standard cross entropy loss as described in (Lin et al., 2017). The reason might be due to that the imbalance degrees of the datasets in this thesis are much smaller than the one in (Lin et al., 2017).

The results of the experiments using the Vaihingen and Potsdam datasets have shown that *FuseNet-lite* is comparable to state-of-art methods. The difference in OA achieved by *FuseNet-lite* is within 1.3% of the best published results in both datasets, whereas the difference in the mean F1 score achieved by *FuseNet-lite* is within 1% to the best ones. However, most of state-of-the-art methods rely on a ResNet-based structure, resulting in a large number of unknown trainable parameters. Compared to them, *FuseNet-lite* requires a much smaller number of parameters, namely only about $\frac{1}{100}$. This small number of parameters can prevent the CNN from overfitting the training data to a certain degree. However, further research is required to improve the classification performance of *FuseNet-lite*.

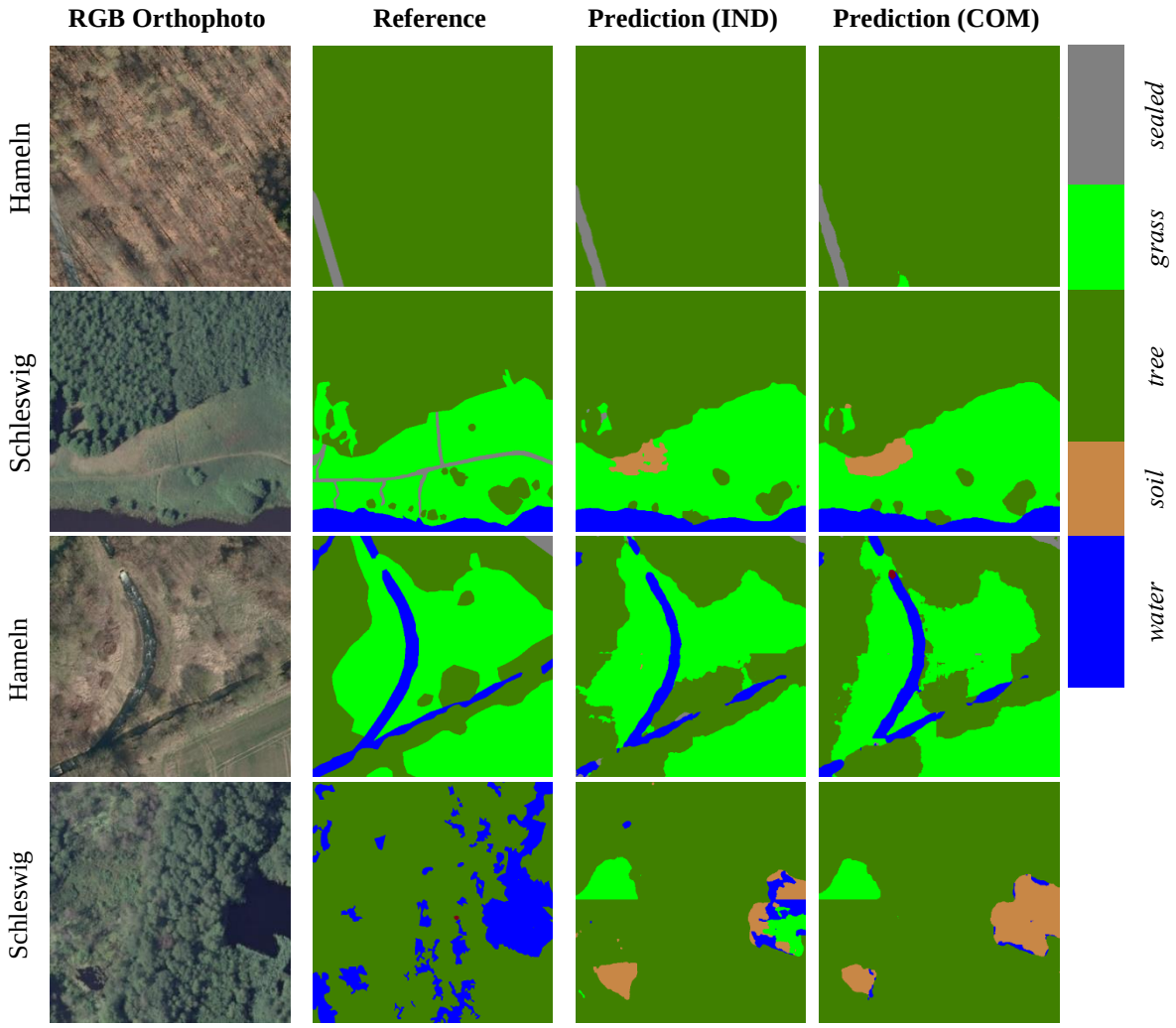


Figure 6.13: Examples for the different visual appearance of *tree* and *water* in Hameln and Schleswig, respectively, and corresponding classification results of *FuseNet-lite*. Prediction (IND): the network was trained only using samples from the considered test site; Prediction (COM): the network was trained using a combination of samples from both test sites.

Test Site		**F1 [%]**								mF1 [%]	OA [%]
		building	*sealed area*	*soil*	*grass*	*tree*	*water*	*car*	*clutter*		
Hameln	COM	94.3	87.2	83.4	87.9	88.7	93.8	72.2	45.4	81.6	88.8
	Δ	+0.1	+0.2	+0.6	-0.3	0	-1.4	0	+3.2	+0.3	0
Schleswig	COM	92.1	84.3	78.3	83.9	91.3	89.4	60.7	40.0	77.5	87.2
	Δ	-0.1	+0.1	+1.7	+0.8	0	-1.9	-0.1	+16.3	+2.0	+0.7

Table 6.6: Classification results in Hameln and Schleswig. COM: results based on combined training; Δ: difference between the results of combined training and individual training, the latter being shown in Tab. 6.1. Green color denotes an increase and red color denotes a decrease of the corresponding metric.

Test Site		F1 [%]						mF1 [%]	OA [%]
		building	*impervious surface*	*low vegetation*	*high vegetation*	*car*	*clutter*		
Vaihingen	COM	93.8	89.8	81.0	87.1	77.5	56.2	80.9	88.0
	Δ	+0.2	+0.1	-0.4	+0.3	0	+1.8	+0.3	+0.1
Potsdam	COM	96.0	90.6	84.4	84.5	89.0	52.5	82.8	88.1
	Δ	+0.5	0	-0.2	-0.7	-1.1	-0.1	-0.3	0

Table 6.7: Classification results in Vaihingen and Potsdam. COM: results based on combined training; Δ: difference between the results of combined training and individual training, the latter being shown in Tab. 6.2. Green color denotes an increase and red color denotes a decrease of the corresponding metric.

6.2. Evaluation of Land Use Classification

6.2.1. Prediction Variability of *LuNet-lite-JO*

In order to investigate the variability of the results, ten experiments were conducted based on the same training, validation and test data, corresponding to the set of experiments called Variability-LU in Tab. 5.10. For all experiments, the hyperparameter settings are kept fixed as presented Tab. 5.11, and the Schleswig dataset described in Section 5.3.2.2 is used. Means and standard deviations of the OA and the mean F1 scores achieved on the test set are reported.

Fig. 6.14 presents the standard deviations of OA and the mean F1 scores over the three semantic levels. Looking at the OA, as the semantic level and, thus, the number of classes to be discerned, increases, the standard deviations of OA increase slightly, from about 0.6% at Level I to 0.9% at Levels II and III. In particular, they are denoted as $\sigma_{OA}^{LI} = 0.6\%, \sigma_{OA}^{LII} = 0.9\%, \sigma_{OA}^{LIII} = 0.9\%$. It seems that with more classes to be discerned, the prediction variability of *LuNet-lite-JO* becomes slightly larger, but it is still low. Similar results are also observed for the standard deviations of the mean F1 scores: the more classes are to be discerned, the larger the predictions made by *LuNet-lite-JO* vary. The corresponding standard deviations of the mean F1 scores are about 1% at Level I and 1.5% at Levels II and III. In particular, they are denoted as $\sigma_{mF1}^{LI} = 1\%, \sigma_{mF1}^{LII} = 1.5\%, \sigma_{mF1}^{LIII} = 1.5\%$. Similarly to the observation in land cover classification, the mean F1 scores have larger standard deviations than the OA, which is very likely to be caused by the imbalanced class distribution.

Fig. 6.15 presents the class distribution of all semantic levels and the standard deviations of all individual F1 scores on test data. One can see that there is a tendency of negative correlation (the *r* values in the figure) between the percentage of samples belonging to a class and the standard deviations of the F1 scores. If there is a sufficient amount of trainings samples for a class, its prediction is more stable than if it is underrepresented. For instance, at level I, the class *settlement* (class ID = 1, first row in Fig. 6.15) corresponds to 38% of the whole training data and achieves a standard deviation of its F1

score of about 0.8%, whereas the class *water bodies* (class ID = 4) is underrepresented (less than 5% of the training samples), and has a larger standard deviation of F1 scores of about 2.7%, indicating that its prediction is more variable. Similar outcomes can also be seen from the results at levels II and III. In summary, the standard deviations of mean F1 scores are largely affected by the standard deviations of underrepresented classes.

The answer to the question raised in Section 5.3.2.4 is that *LuNet-lite-JO* is relatively stable when it makes predictions. In particular, in terms of OA the standard deviations are less than 1%, and at the coarsest level the lowest value of about 0.6% can be achieved. In terms of mean F1 scores, there is larger deviation as the number of classes increases, and a standard deviation of about 1.5% can be achieved for all semantic levels.

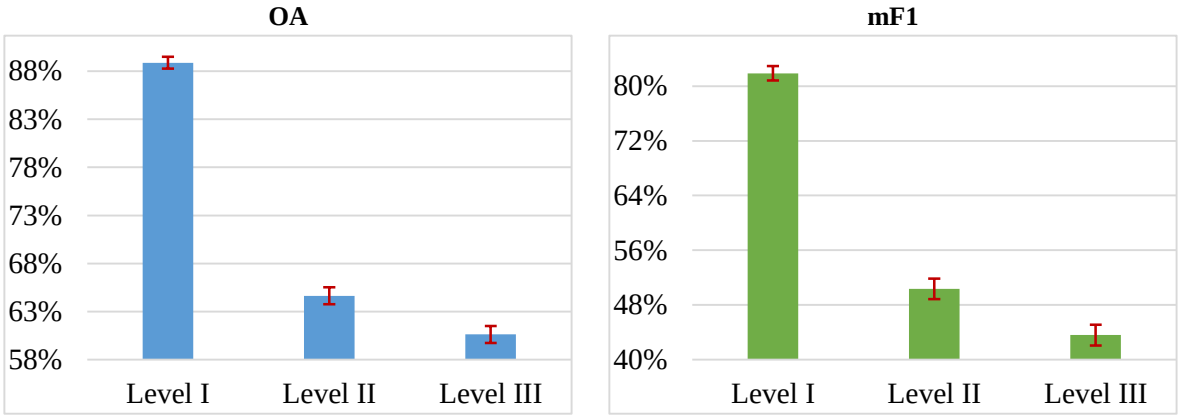


Figure 6.14: Mean OA and the mean F1 scores with their standard deviations (dark red) over all semantic levels (Level I, II and III) achieved by *LuNet-lite-JO* for the Schleswig dataset. X-axis: training epoch; Y-axis: OA and mF1 score in [%].

6.2.2. Investigations of the Hyperparameter Settings

In this section, the impact of the three hyperparameters described in Section 5.3.2.5 on the land use classification are investigated. All experiments in the set called Hyper-LU in Tab. 5.10 are conducted and analysed.

6.2.2.1. Base Learning Rate

While investigating the impact of the the base learning rate η on the results, the values for the mini batch size Ω and the weight of the penalty term ϵ in the loss (eqs. 4.16-4.17) are set to 30 and 1, respectively. There are five investigated values of η to be tested, which are 1, 0.1, 0.01, 0.001 and 0.0001. Fig. 6.16 presents the training and validation errors as a function of training epoch, and Fig. 6.17 presents the test OA and mean F1 scores for different values of η.

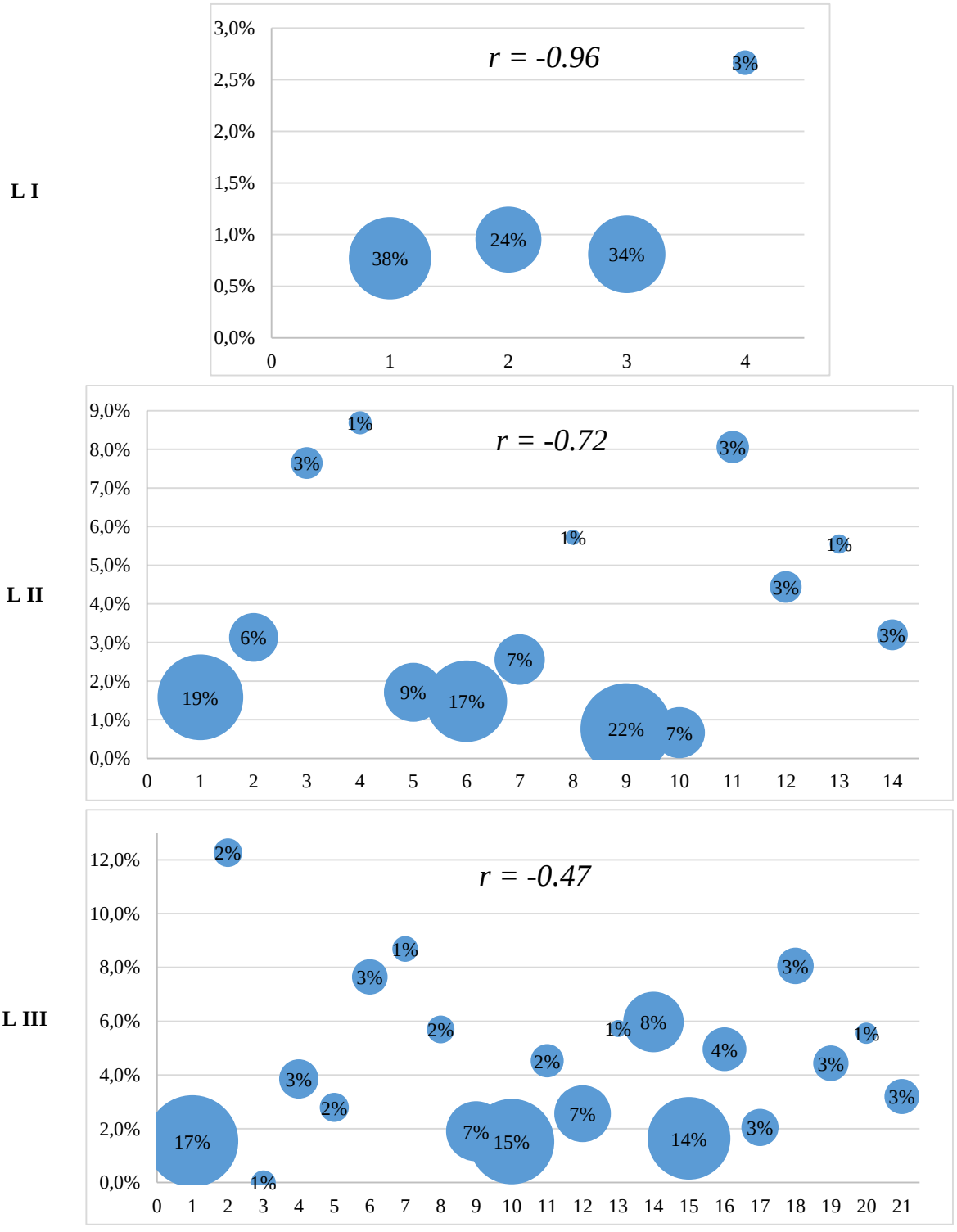


Figure 6.15: The standard deviations of the F1 scores at different semantic levels (L I – L III). X-axis: class IDs (cf. Tab. 5.2). Y-axis: standard deviations of F1 scores; The bubble size describes the percentage of samples of a class in the dataset, and the percentages are shown in the center of the bubbles. The correlations (r) between the percentage of samples belonging to a class and standard deviations of the F1 scores at each semantic level are shown at the top of each chart.

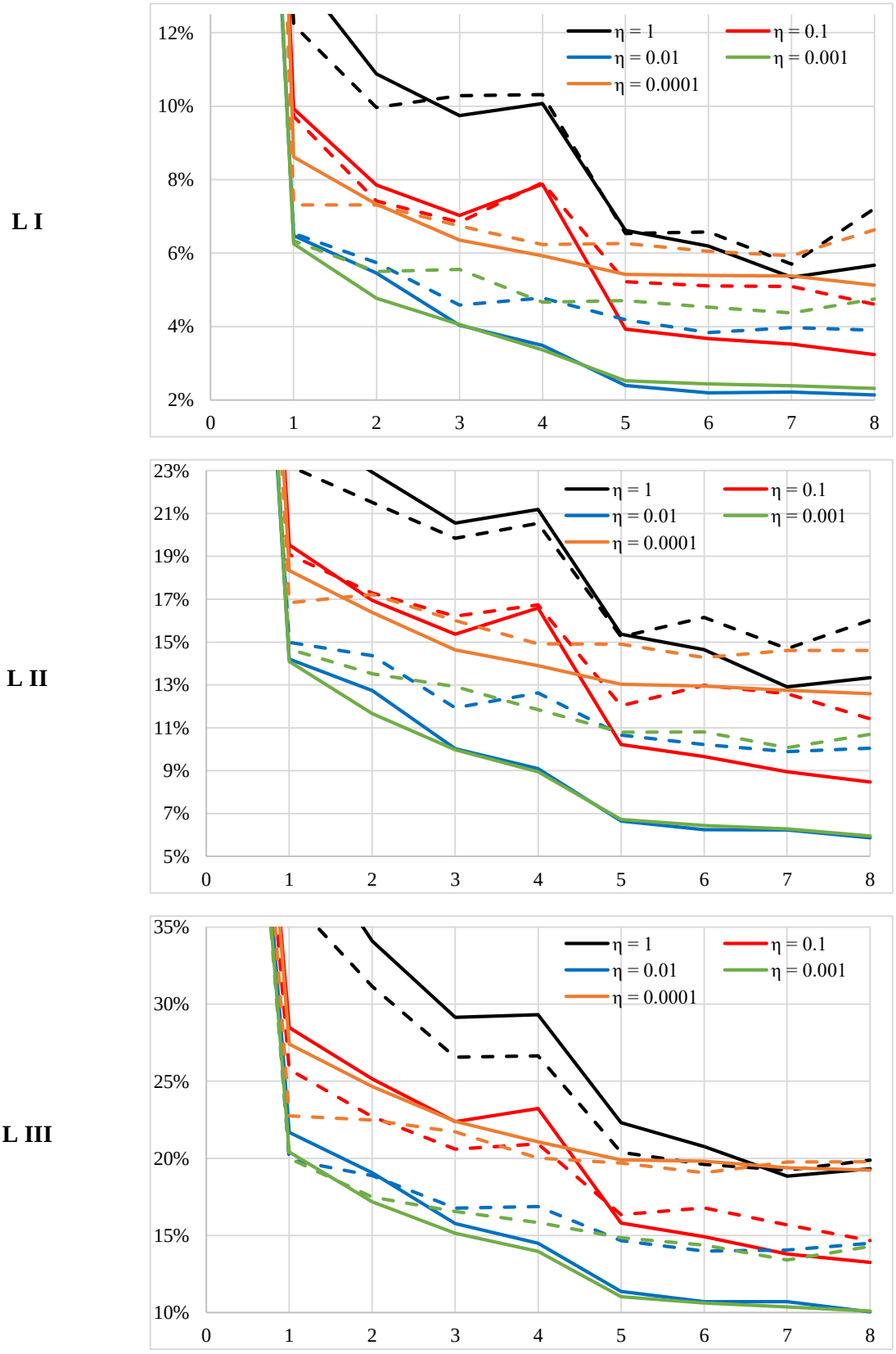


Figure 6.16: Training and validation errors as a function of training epoch for different values of η over three semantic levels using the Schleswig dataset. Solid lines: training errors; dashed lines: validation errors.

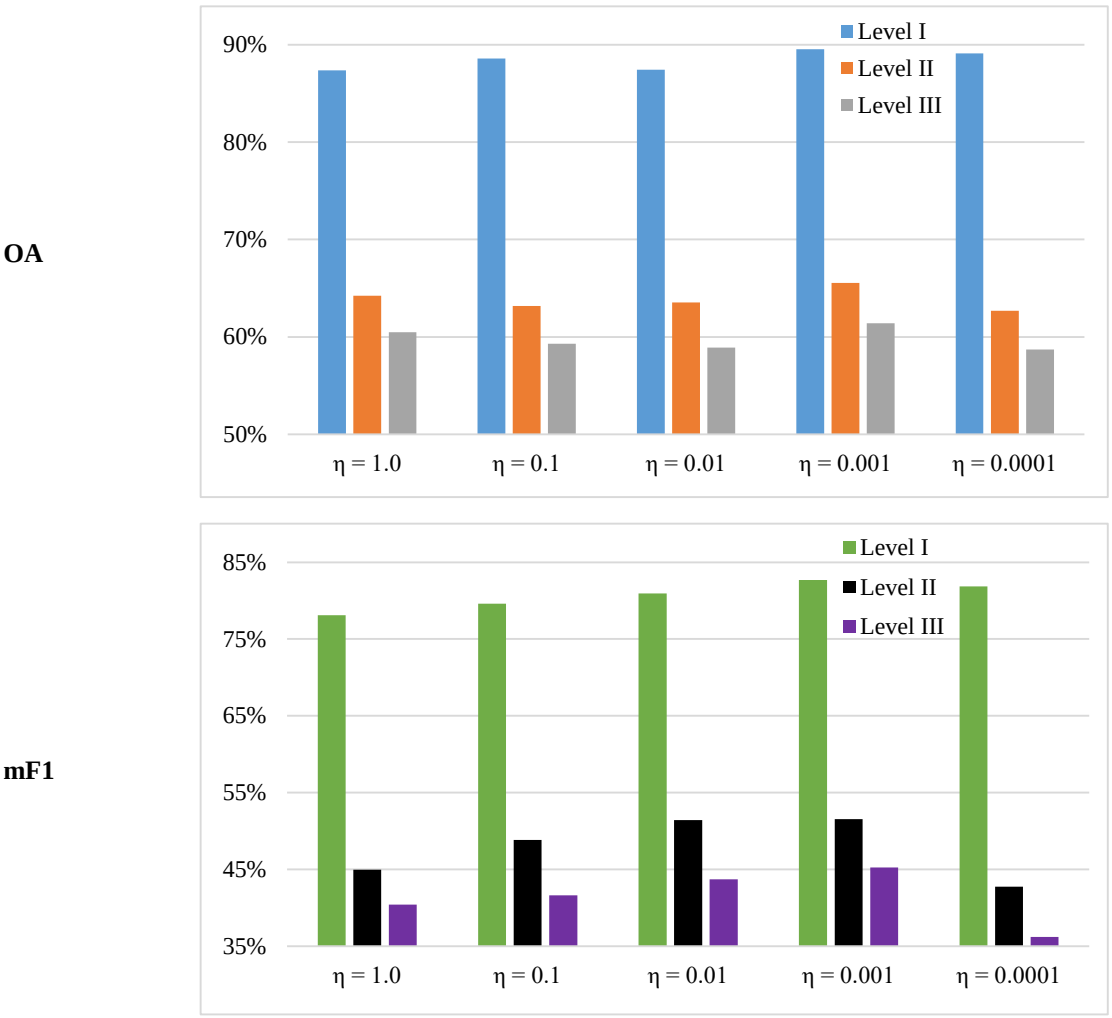


Figure 6.17: OA and mean F1 scores for different values of η over three semantic levels on the Schleswig test data.

Focusing on the training and validation errors, for all values of η, the main tendency of the training errors is to decrease as training epoch increases. At level I, as η decreases from 1 to 0.001, the training error decreases. For instance, after the final epoch 8, the training error is about 5% using $\eta = 1$ and only about 2% for $\eta = 0.001$. A similar behavior can also be observed for the training errors at levels II and III. Another obvious finding is that as the semantic level increases, the training error increases considerably. For example, after the final epoch 8, the training error increases from about 2% at level I to about 10% at level III for $\eta = 0.001$. This large training error is likely due to the fact that there are more classes to be discerned, making it more difficult for the CNN to classify the training data. However, as η decreases further from 0.001 to 0.0001, the training error increases again: at the final epoch 8 it is increased by about 2% at level I, by 7% at level II and by 9% at level III. Regarding to validation errors, one can observe a similar behavior as for the training errors at different values of η over all levels. From the observations mentioned so far it can be concluded that $\eta = 1$ and $\eta = 0.0001$ are improper values for the learning rate, and $\eta = 0.01$ and $\eta = 0.001$ are good choices.

Checking the validation errors is important because it gives information whether the trained model overfits to the training data or not. If the training and validation errors were 0, the trained model would fit the training data perfectly without overfitting. Fig. 6.16 shows training and validation errors over all values of η and over all semantic levels. It is easy to see that the differences between the validation and training errors are quite small for $\eta = 1$ and $\eta = 0.0001$ over all levels, yet large training errors are observed, probably indicating an underfitting problem. For a proper value of η of 0.01 or 0.001, the corresponding differences are a little larger: about 2% at level I, 4% at levels II and III. These numerical values indicate that there is some overfitting, but only to a small degree.

Now switching the focus on the test results at different levels, all values of η lead to promising results in terms of OA of about 89% at level I. Here, $\eta = 0.001$ delivers the best OA of about 90%. However, in terms of mean F1 scores, the differences between different values of η are larger. For instance, $\eta = 1$, for which the CNN is not well trained, delivers the worst mean F1 score of about 78% whereas $\eta = 0.001$ delivers the best mean F1 score of about 83% at level I. As the semantic level increases, there is a drop for OA of about 25% between levels I and II and of about 5% between levels II and III for all values of η. The best results in terms of OA is delivered by $\eta = 0.001$: about 66% at level II and about 61% at level III. At the same time, $\eta = 0.001$ also delivers the best mean F1 scores at levels II and III.

The answer to the question (a) raised in Section 5.3.2.5 is that too large (i.e. $\eta = 1$) or too small (i.e. $\eta = 0.0001$) values for the base learning rate cause the model to not fit the training data well to a certain degree, indicated by the large training errors. Using a value of $\eta = 0.001$ leads to good results in the experiments.

6.2.2.2. Mini Batch Size

While investigating the mini batch size Ω, the values of η and ϵ are set to 0.001 and 1, respectively. As Tab. 5.12 in section 5.3.2 shows, there are five values for Ω to be tested (2, 5, 10, 20 and 30). Fig. 6.18 presents the training and validation errors as a function of training epoch, and Fig. 6.19 presents the OA and the mean F1 scores on the test set for different values of Ω.

Fig. 6.18 shows for $\Omega = 2$ the training errors are much higher than those for other values over all levels. For instance, at level I the training errors achieved using $\Omega = 2$ are about 8% higher than those for $\Omega = 5$. Furthermore, an obvious tendency is that as Ω increases, the training errors decrease at all semantic levels. For $\Omega = 30$, the lowest training errors are observed: the training error at the final epoch 8 is about 2% at level I, about 6% at level II and about 10% at level III. For a fixed Ω, as the semantic level increases, the training errors also increase, which is mainly due to the increased difficulties in differentiating on the increasing number of classes. The results shown in Fig. 6.18 indicate that using a larger mini batch size is important and helpful to train a CNN. For a small value of $\Omega = 2$, it is clear to see that over all levels the validation errors are larger than those achieved for other values of Ω. As the CNN does not fit the training data well, which is indicated by large training errors, it is not a surprise that the model results in larger validation errors. As Ω increases, the validation errors decrease accordingly, but the differences between validation errors and training errors increase over all levels.

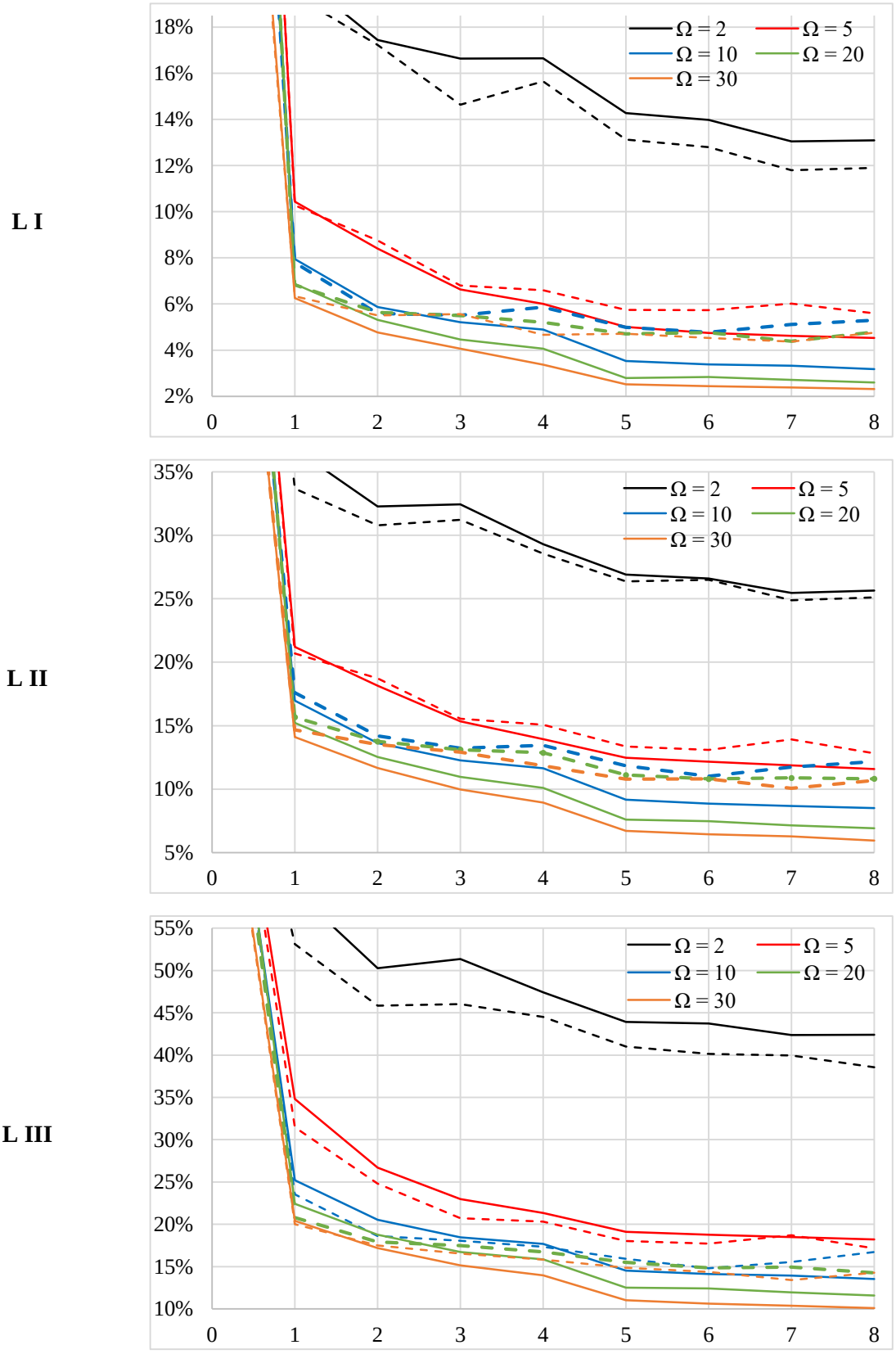


Figure 6.18: Training and validation errors as a function of training epoch for different values of Ω over three semantic levels. Solid lines: training errors; dashed lines: validation errors.

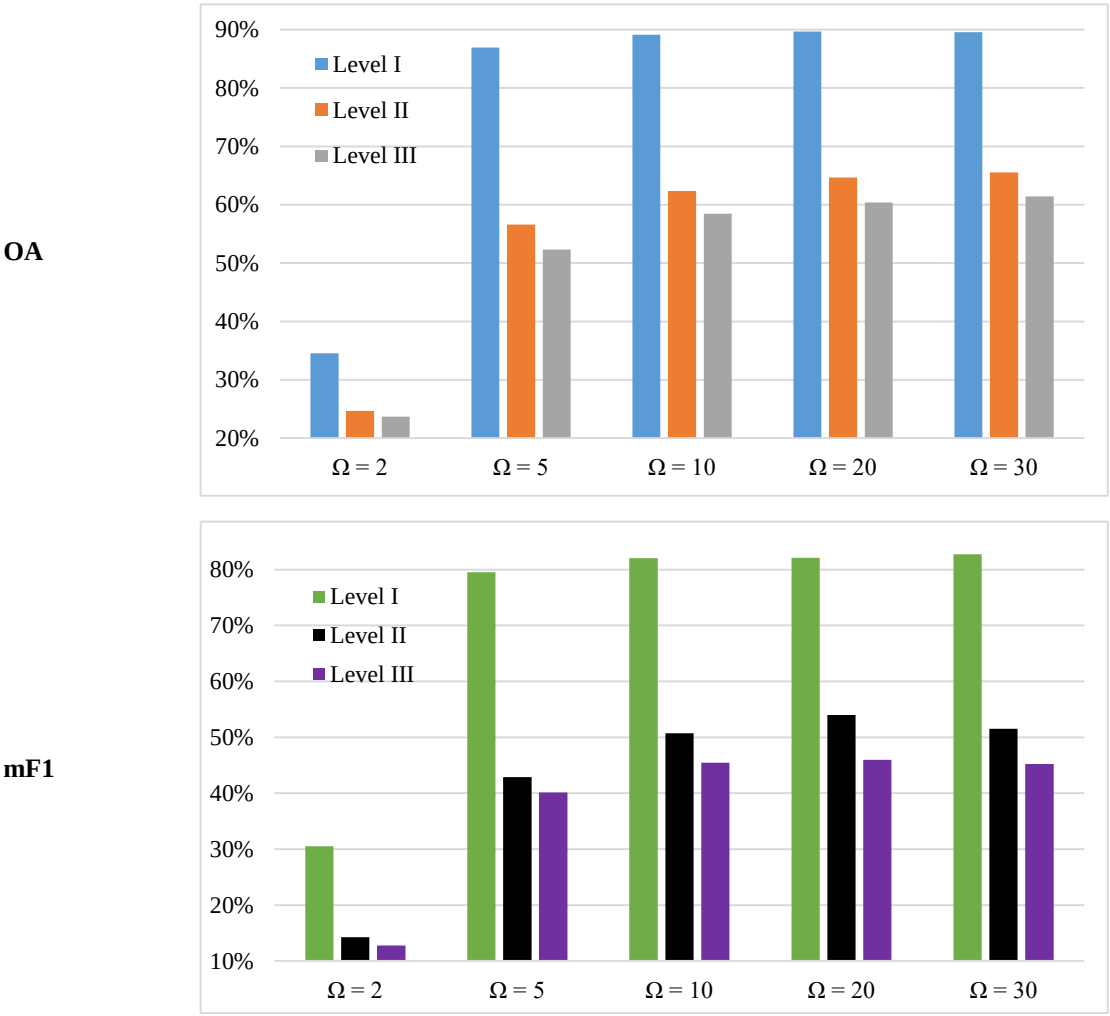


Figure 6.19: OA and the mean F1 scores for different values of Ω over three semantic levels on the Schleswig test data.

Taking $\Omega = 5$ and $\Omega = 30$ for comparison, for $\Omega = 5$ the differences are about 1%, 1.5% and 0% at the three semantic levels, respectively, whereas they are about 2%, 4% and 4% for $\Omega = 30$. Indeed, a large difference indicates that the model overfits to the training data to a certain degree. Switching the focus on the test results in Fig. 6.19, it is clear that $\Omega = 2$ delivers the worst results in terms of both OA and mean F1 scores over all levels, again indicating that the CNN is not well-trained. As Ω increases, the test results get better and better, and the best OAs over all levels are delivered by the largest value of $\Omega = 30$, with OA of about 90% at level I, 66% at level II and 61% at level III. In addition, $\Omega = 30$ also delivers promising mean F1 scores over all levels.

In conclusion, too small a mini batch size (i.e. $\Omega = 2$) results in the worst classification performance, and as the mini batch size increases, smaller training and validation errors could be achieved. This conclusion is the answer to the question (a) raised in Section 5.3.2.5. However, due to a limited capacity of the GPU, the largest mini batch size which could be used is 30. It may still be possible to obtain better results by increasing the mini batch size if the available hardware allows it. It has to be noted that there

might be a dependency between mini batch size and base learning rate, which is not investigated in this thesis.

6.2.2.3. The Weight of the Penalty Term in the Focal Loss

In this section, investigations of the weight of the penalty term ϵ in the loss functions in eqs. 4.16 – 4.17 are performed. While investigating ϵ, the values of the base learning rate η and the mini batch size Ω are set to 0.001 and 30, respectively. As presented in Tab. 5.12 in section 5.3.2, the values for ϵ to be investigated are 0, 0.5, 1, 2, 5. Fig. 6.20 presents the training and validation errors as a function of training epoch, and Fig. 6.21 presents the test OA and mean F1 scores for different values of ϵ.

Looking at the results presented in Fig. 6.20, the training errors for different values of ϵ are very close to each other at levels I and II. The largest training error is achieved for $\epsilon = 5$. Turning the focus on the validation errors, there is no clear relation between increasing the value of ϵ and the validation errors at each level. On average, the validation errors for different values of ϵ do not vary much ($\pm 0.3\%$ at level I and $\pm 0.4\%$ at level II) except the one for $\epsilon = 5$ at level III, which is about 2% higher than others. It seems that the variations of ϵ do not affect the training too much. Regarding to the test results shown in Fig. 6.21, the best results is delivered by using $\epsilon = 1$ with OAs of about 90% at level I, about 66% at level II and about 61% at level III, and with mean F1 scores of about 83% at level I, about 52% at level II and about 45% at level III. Compared to $\epsilon = 0$, which excludes the penalty in eqs. 4.16 and 4.17, the proposed penalty term helps the classification a little. Improvements of up to +2% (at level II) for OA and up to +3% (at level III) for the mean F1 score can be observed. Fig. 6.22 presents an overview of the F1 scores of all individual classes at semantic level I as a function of the values of ϵ. Compared to the F1 score of the unrepresented class *water body* delivered by using the value of $\epsilon = 0$, an improvement by 1.4% is observed when using the value of $\epsilon = 0.5$, but for other values of $\epsilon > 0$, a decrease of this F1 score is observed. A similar behavior is observed for the underrepresented classes when using a value of $\epsilon > 0$ at levels II and III.

The answer for the question (a) raised in Section 5.3.2.5 is that it seems that ϵ does not play an important role on training a CNN. Compared to the loss excluding the penalty term ($\epsilon = 0$), a proper value of ϵ ($\epsilon = 1$ in this thesis) contributes positively to the classification to a certain degree. In this investigation, improvements for OA and mean F1 score are up to +2% (about $2 \cdot \sigma_{OA}^{LIII}$) and +3% (about $2 \cdot \sigma_{mF1}^{LIII}$) have been observed. For underrepresented classes, there are also improvements when using a value of $\epsilon > 0$. This is the answer to the question (b) raised in Section 5.3.2.5.

6.2.3. Impact of Joint Optimization

In this section, an evaluation using the three individual datasets is performed to answer the two scientific questions raised in Section 5.3.2.6. All experiments of the experiment sets called Baseline-JO, Exp-MT and Exp-F2C in Tab. 5.10 are conducted.

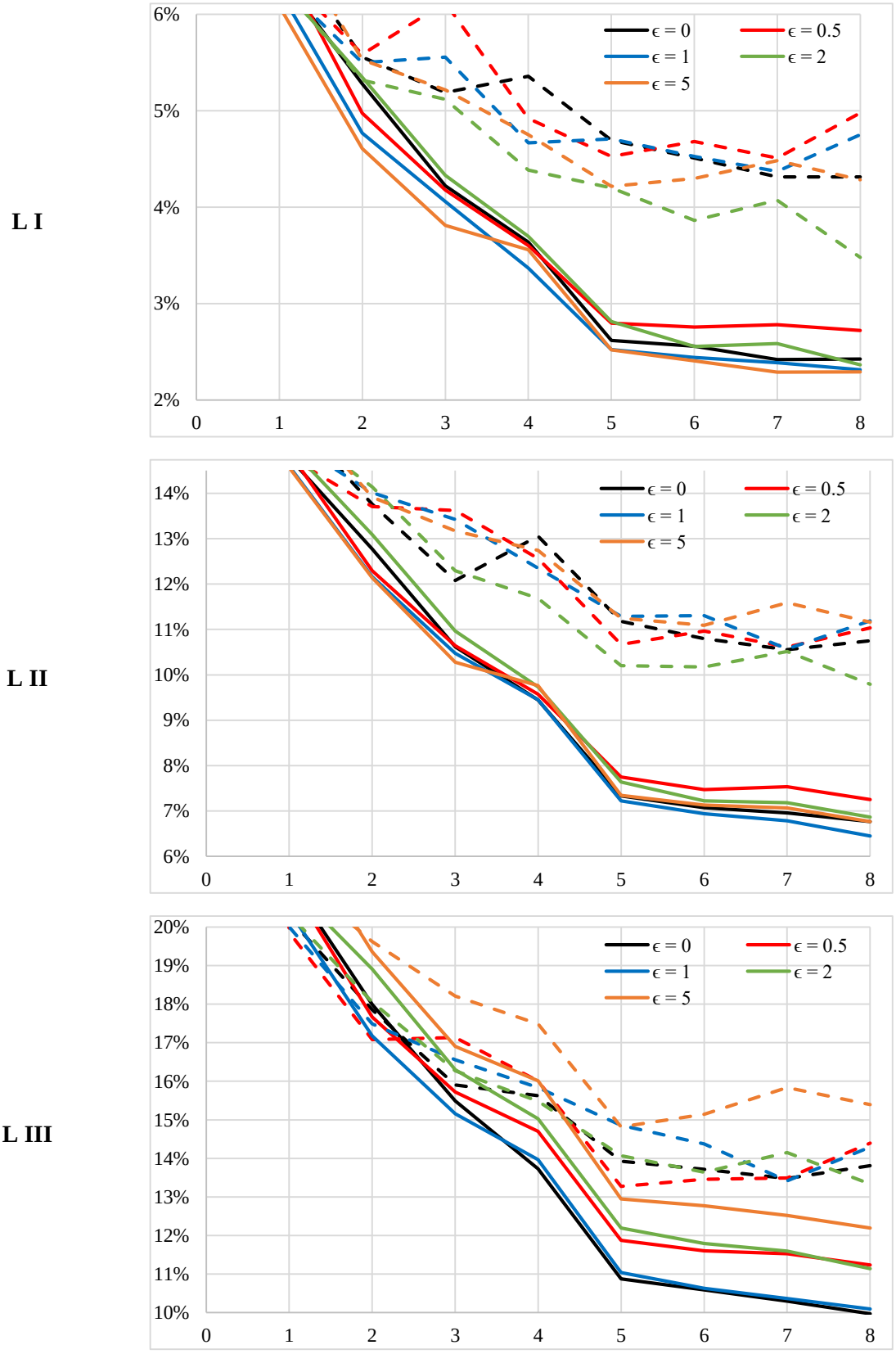


Figure 6.20: Training and validation errors as a function of training epochs for different values of ϵ over three semantic levels on the Schleswig dataset. Solid line: training errors; dashed line: validation errors.

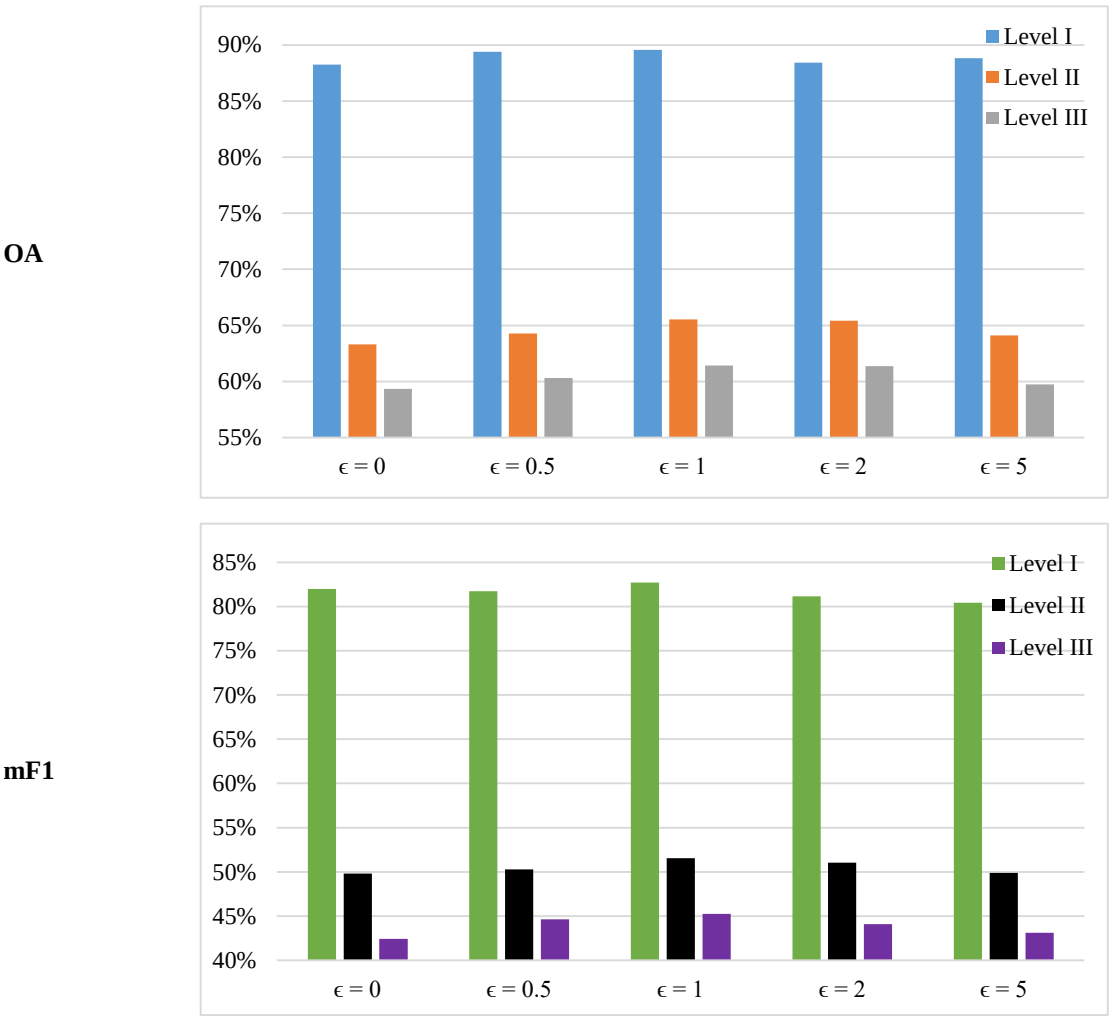


Figure 6.21: OA and mean F1 scores for different values of ϵ over three semantic levels on the Schleswig test data.

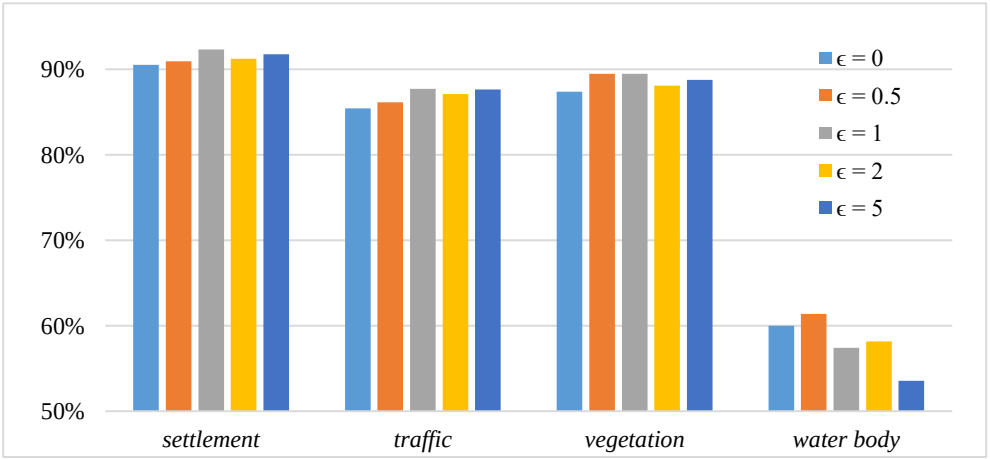


Figure 6.22: F1 scores of the classes at semantic level I as a function of the values of ϵ in the Schleswig test data.

More precisely, experiments on Hameln, Schleswig and MV datasets are conducted based on the network variants *LuNet-lite-JO*, *LuNet-lite-MT* and *LuNet-lite-F2C*, and their inputs are the patches of the input configuration Mask-T-Full in Tab. 5.7. It has to be noted that *LuNet-lite-MT* cannot deliver predictions while guaranteeing the consistency with the class hierarchy, and although both *LuNet-lite-JO* and *LuNet-lite-F2C* deliver consistent hierarchical predictions, the latter one requires a sequential post-processing approach. One main drawback of the post-processing approach is that if the prediction at the finest semantic level is wrong, the entire prediction is wrong and cannot be corrected. Tab. 6.8 presents an overview of the results (OA and the mean F1 scores) of all network variants in the three test sites.

Test Site	**Level**	**Metrics**	Experiment Set		
			Baseline-JO	Exp-F2C	Exp-MT
			LuNet-lite-JO	*LuNet-lite+F2C*	*LuNet-lite-MT*
Hameln	I	OA [%]	91.5	92.3	**92.7**
		mF1 [%]	85.9	86.1	**86.7**
	II	OA [%]	78.5	**78.7**	78.2
		mF1 [%]	59.9	**61.5**	61.1
	III	OA [%]	74.1	**74.6**	73.8
		mF1 [%]	51.8	**54.7**	51.2
Schleswig	I	OA [%]	**91.0**	90.1	90.8
		mF1 [%]	85.4	85.5	**85.7**
	II	OA [%]	**74.8**	74.1	73.7
		mF1 [%]	61.8	63.0	**63.9**
	III	OA [%]	**70.4**	70.0	69.8
		mF1 [%]	55.3	57.1	**57.2**
MV	I	OA [%]	79.7	79.9	**80.9**
		mF1 [%]	79.8	80.0	**81.0**
	II	OA [%]	67.7	**68.8**	68.4
		mF1 [%]	52.0	52.9	**53.4**
	III	OA [%]	64.5	65.3	**65.4**
		mF1 [%]	40.3	**42.2**	41.3

Table 6.8: Overview of the results achieved for Hameln, Schleswig and MV using the three network variants. Best values in each site are printed in bold font. Refer to Tab. 5.10 for the experiment sets.

Comparing *LuNet-lite-JO* and *LuNet-lite-F2C*, in Hameln and MV, the variant requiring post-processing approach outperforms the variant based on joint optimization in all metrics (up to +1.1% in terms of OA at level II in MV and +2.9% in terms of the mean F1 score at level III in Hameln). In Schleswig, *LuNet-lite-JO* delivers better predictions than *LuNet-lite-F2C* in terms of OA at all semantic levels, but again worse results in terms of the mean F1 score. Turning the focus on the comparison between *LuNet-lite-JO* and *LuNet-lite-MT*, the joint optimization delivers better predictions than the pure multi-task learning in most cases in Hameln and Schleswig. For instance, at level II in Schleswig the variant based on joint optimization outperforms the one based on the multi-task learning by more than 1% in terms of OA. Nonetheless, there is another picture for the predictions in MV. The multi-task

learning outperforms the variant based on joint optimization in all cases with increase of OA of up to +0.9% and an increase of the mean F1 score of up to +1.4%. As mentioned earlier, the predictions of *LuNet-lite-MT* do not guarantee consistency with class hierarchy. There are about 7.3%, 9.5% and 13.1% of samples for which the predictions of the three semantic levels are not consistent in Hameln, Schleswig and MV, respectively, when multi-task learning is applied.

In conclusion, the variant based on joint optimization performs better than the variants *LuNet-lite-MT* and *LuNet-lite-F2C* in some cases, and a consistent hierarchy of the predictions is kept, which serves as the answer to the question raised in Section 5.3.2.6. However, F2C performs better in some other cases than the JO approach.

In Tab. 6.8, it is clear to see that the results in MV (OA and the mean F1 scores at all levels) are considerably worse than those in Hameln and Schleswig. For instance, at level I, at which the classes are supposed to be more easily separated, the OA in MV is about 80%, which is about 10% worse than the OAs in Hameln and Schleswig. Similar tendencies are observed at levels II and III. To know the reason of the decrease, a patch-based evaluation is performed for the three datasets, because the final predictions of the polygons rely on the patch-wise predictions which are directly delivered by *LuNet-lite-JO*. Furthermore, the variability of the predictions of all patches of a polygon is investigated (in an ideal case, all predictions of these patches of a polygon shall be identical to support the final prediction of that polygon, in which case the variability would be zero). If the variability is large, the combined prediction of the corresponding polygon is more likely to be wrong.

Patch-based evaluation

Tab. 6.9 presents the evaluation results in terms of OA and the mean F1 scores in the three datasets. The corresponding numbers of test patches are shown as well. As far as the OA is concerned, there is no large difference between these datasets. Having a look at the mean F1 scores, it is easy to see that the numerical values over all levels in Hameln and Schleswig are very close, yet in MV the values are about 5-7% lower than the corresponding ones in Hameln and Schleswig. In conclusion, the network variant *LuNet-lite-JO* performs well in all datasets.

Test Site	**Level I**		**Level II**		**Level III**		**#Test Patches**
	OA [%]	mF1 [%]	OA [%]	mF1 [%]	OA [%]	mF1 [%]	
Hameln	93.5	92.4	83.4	62.9	76.3	52.7	9869
Schleswig	93.6	92.0	83.8	59.7	77.4	52.3	28643
MV	92.2	84.9	83.7	55.7	76.8	45.6	61851

Table 6.9: Patch-based evaluation results delivered by *LuNet-lite-JO* for Hameln, Schleswig and MV: OAs and mean F1 scores. The total number of test patches in each dataset is shown in the rightmost column.

Variability of the predictions for patches of the same object

To obtain the variability of the predictions of all patches of a polygon, four steps are taken. First, for each patch, the matches between the predictions and the reference at each semantic level are counted. Subsequently, the sum of the number of matches is divided by the number of semantic levels, and the

result is called *similarity*, denoted by ϱ. A value of $\varrho = 1$ indicates a perfect match and a value of $\varrho = 0$ indicates that the prediction is totally wrong. After obtaining the similarity values of all patches of a polygon, mean and standard deviation of these similarity values for that polygon are computed. Finally, for all polygons which have at least two patches, mean similarity (μ_ϱ) and the corresponding standard deviation (σ_ϱ) are reported. The predictions of all sub patches of a polygon are expected to be similar, resulting in a low variability which would be indicated by a large mean similarity and a small standard deviation. Fig. 6.23 presents the mean similarity and the corresponding standard deviations in Hameln, Schleswig and MV.

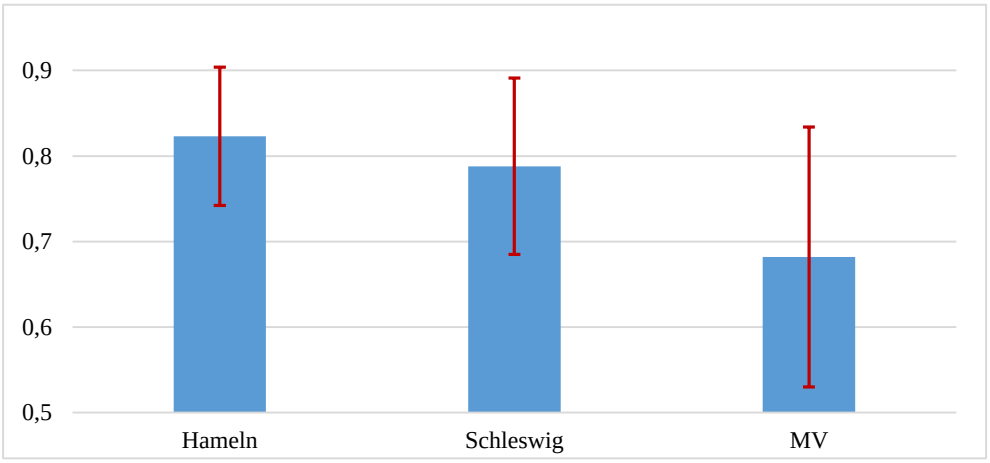


Figure 6.23: μ_ϱ in Hameln, Schleswig and MV. The standard deviations of the similarities (σ_ϱ) are shown in dark red.

Clearly, the mean similarities in Hameln and Schleswig are much higher than that in MV, and the standard deviations are much lower than that in MV. That is to say, the variability of the predictions is much larger in MV than those in Hameln and Schleswig, which may explain the lower performance at the object level in MV. The GSD in MV is higher than that in Hameln and Schleswig (10 cm vs. 20 cm), thus, a fix input size of 256 x 256 pixels results in a smaller context window in MV, which may have a negative influence on the classification performance. One option to enlarge the context window is to use a larger patch size or to downsample the MV dataset by a factor larger than 1. The larger variability in MV is required to be investigated further to know the reason.

6.2.4. Impact of the Polygon Representation

To answer the two questions raised in Section 5.3.2.7, all experiments of the set called Inves-BG in Tab. 5.10 are conducted. The results are compared to those achieved from the experiments of the set Baseline-JO to obtain the answers. Tab. 6.10 presents an overview of all results.

The results obtained in Baseline-JO serve as a baseline for comparison for each test site. Firstly, comparing the results obtained with black background to the baseline. They outperform the latter with

respect to 6 metrics (out of a total of 18 metrics over three levels and three test sites), but, the differences are small: up to +1% (about $1.5 \cdot \sigma_{OA}^{LI}$) in terms of OA (at level I in MV) and +0.6% (about $0.6 \cdot \sigma_{mF1}^{LI}$)in terms of mean F1 score (at level I in Schleswig). Turning the focus on the remaining metrics, the baseline outperforms the results obtained with black background by a slightly larger margin of up to +1.8% (about $2 \cdot \sigma_{OA}^{LIII}$) in terms of OA (at level III in Hameln) and +1.3% (about $1 \cdot \sigma_{mF1}^{LIII}$) in terms of mean F1 score (also at level III in Hameln). It seems that the representation of polygon by binary masks performs slightly better than the implicit polygon representation with black background. Secondly, comparing the results obtained with Gaussian background to the baseline, they outperform the latter with respect to 5 metrics, among which the maximal increase is +0.6% in terms of OA. For all test sites, the baseline outperforms the results obtained with Gaussian background by a large margin of up to +3.1% (about $3 \cdot \sigma_{OA}^{LIII}$) in terms of OA (at level III in Hameln) and +1.7% (about $1.1 \cdot \sigma_{mF1}^{LII}$) in terms of mean F1 score (at level II in Hameln). Based on these analyses, question (a) in Section 5.3.2.7 can be answered: the representation of polygons by a binary mask delivers slightly better results than the implicit polygon representation for land use classification.

Test Site	Experiment Set	Repres.	Level I		Level II		Level III	
			OA [%]	mF1 [%]	OA [%]	mF1 [%]	OA [%]	mF1 [%]
Hameln	Baseline-JO	Mask	91.5	85.9	**78.5**	**59.9**	**74.1**	**51.8**
	Inves-BG	Black	**92.1**	**86.4**	76.8	58.7	72.3	50.5
		Gauss.	**92.1**	85.8	75.6	58.2	71.0	51.2
Schleswig	Baseline-JO	Mask	**91.0**	85.4	**74.8**	61.8	**70.4**	**55.3**
	Inves-BG	Black	90.7	**86.0**	74.0	**62.6**	69.8	54.9
		Gauss.	90.8	85.5	74.2	62.2	70.0	55.2
MV	Baseline-JO	Mask	79.7	79.8	**67.7**	**52.0**	**64.5**	40.3
	Inves-BG	Black	**80.7**	**80.7**	67.5	**52.0**	64.3	40.3
		Gauss.	80.0	80.1	66.9	51.8	63.9	**40.4**

Table 6.10: Results of experiments of the experiment sets Baseline-JO and Inves-BG. Repres.: Polygon Representation, cf. Section 4.3.2. Mask: polygon mask; Black: implicit representation with back background; Gauss.: implicit representation with background affected by Gaussian noise.

To answer question (b) in Section 5.3.2.7, a comparison between the results obtained with black background and Gaussian background is performed. In Hameln and MV, the variant using black background outperforms the one based on Gaussian background with respect to 9 metrics (out of a total of 12 metrics), and the maximum increase is +1.3% for OA (at level III in Hameln) and +0.6% for the mean F1 score (at level I in Hameln). In Schleswig, the results based on black background and Gaussian background are very similar, their maximum absolute differences are +0.2% in terms of OA and +0.5% in terms of mean F1 score over all semantic levels. In conclusion, the implicit polygon representation with black background delivers slightly better predictions than the one with Gaussian background in most cases.

6.2.5. Impact of Land Cover Information

In the proposed two-step framework, land cover information is determined to serve as an additional input for the subsequent land use classification along with image data. In section 5.3.2.8, two scientific questions were raised with respect to the impact of land cover information on the results of land use classification. To answer them, a series of experiments were conducted, in which different types of land cover information (i.e. no land cover, land cover labels and land cover posteriors) are used, resulting in the experiment set called Inves-LC. It has to be mentioned again that an ensemble of the results of the network variant relying on no land cover and the network variant relying on land cover posteriors only is performed, leading to ENS-T-LC. These results are compared to those achieved in Baseline-JO, so as to answer both questions. Tab. 6.11 presents an overview of the results of all involved experiments.

Test Site	Experiment Set	Input Configuration	Level I		Level II		Level III	
			OA [%]	mF1 [%]	OA [%]	mF1 [%]	OA [%]	mF1 [%]
Hameln	Inves-LC	Mask-T-NoLC	92.3	85.4	78.1	60.7	74.0	**53.4**
		Mask-T-LabelLC	92.4	87.0	77.6	60.1	73.0	52.3
	Baseline-JO	Mask-T-Full	91.5	85.9	78.5	59.9	74.1	51.8
	ENS-T-LC	Mask-T-NoLC; Mask-T-ProbLC	**93.4**	**89.7**	**79.4**	**62.3**	**75.4**	53.3
Schleswig	Inves-LC	Mask-T-NoLC	**91.1**	86.2	74.4	61.4	70.5	56.0
		Mask-T-LabelLC	91.0	**86.5**	**75.5**	**64.4**	**71.5**	**57.7**
	Baseline-JO	Mask-T-Full	91.0	85.4	74.8	61.8	70.4	55.3
	ENS-T-LC	Mask-T-NoLC; Mask-T-ProbLC	90.9	85.6	75.2	61.2	71.1	55.2
MV	Inves-LC	Mask-T-NoLC	80.8	80.8	68.2	**52.9**	65.1	**41.5**
		Mask-T-LabelLC	80.2	80.3	67.3	52.1	64.2	40.7
	Baseline-JO	Mask-T-Full	79.7	79.8	67.7	52.0	64.5	40.3
	ENS-T-LC	Mask-T-NoLC; Mask-T-ProbLC	**81.0**	**81.0**	**69.2**	52.1	**66.3**	40.7

Table 6.11: Overview of the results in different experiment sets. Tab. 5.10 presents the experiment sets.

Here, the results obtained in the variant relying on the inputs which exclude land cover information, corresponding to the input configuration Mask-T-NoLC, serve as the baseline for comparison. In general, the classification incorporating land cover information outperform the baseline in most metrics over all semantic levels, except the OA at level I in Schleswig, the mean F1 score at level III in Hameln and the mean F1 scores at levels II and III in MV; the differences of these values to the second best ones are smaller than 1%.

Comparing the results of Baseline-JO, which uses land cover posteriors as input, to the baseline in the three datasets, only 5 metrics (out of 18 metrics) perform better, with a difference of up to 0.4%. Looking at the results of the variant using land cover label as input (i.e. the input configuration Mask-T-LabelLC), only 7 metrics perform better than the baseline. The increases are up to +1.0% (about $1.1 \cdot \sigma_{OA}^{LIII}$) in terms of OA (at level III in Schleswig) and +3.0% (about $2 \cdot \sigma_{mF1}^{LII}$) in terms of the mean F1 score (at level II in Schleswig). Finally, a comparison between the results of ENS-T-LC and the baseline

is performed. In this case, 11 out of a total of 18 metrics over the three datasets are improved: up to +1.4% (about $1.5 \cdot \sigma_{OA}^{LIII}$) in terms of OA (at level III in Schleswig) and +4.5% (about $3 \cdot \sigma_{mF1}^{LIII}$) in terms of the mean F1 score (at level I in Hameln). It seems that this way of incorporating land cover information improves the classification most.

The results of ENS-T-LC outperform the ones based on the input configuration Mask-T-LabelLC in 10 metrics with +0.9% to +2.4% in terms of OA and +0.2% to 2.7% in terms of mean F1 scores over all semantic levels. For those rest metrics, the ensemble method does not outperform the other one: the differences are -0.1% to -0.4% in terms of OA and -0.4% to -3.2% in terms of mean F1 scores. Conclusively speaking, the ensemble method delivers better results than the ones relying on land cover labels in most cases. Comparing the results of ENS-T-LC and Baseline-JO, it can be seen that 15 out of a total of 18 metrics are improved by the ensemble method. The improvements are up to +1.9% (about $3 \cdot \sigma_{OA}^{LI}$) in terms of OA (at level I in Hameln) and +3.8% (about $4 \cdot \sigma_{mF1}^{LI}$) in terms of mean F1 score (also at level I in Hameln).

On the basis of the analysis described so far, the answer to the question (a) in Section 5.3.2.8 is that the land cover information does have positive impact on land use classification. In the pursuit of obtaining a higher accuracy, the optimal way is to perform classification using image data and land cover posteriors separately, and then making a final prediction by combining these results via an ensemble. This is the answer to the question (b) in Section 5.3.2.8. In the following sections, the land cover information is utilized in a way as ENS-T-LC does.

6.2.6. Impact of the Patch Generation Approach

With respect to the scaling approach, two questions have been raised in Section 5.3.2.9. To answer the question (a), all experiments of the experiment set called Inves-Scaling in Tab. 5.10 were conducted, in which the scaled patches, corresponding to the input configurations Mask-S-NoLC and Mask-S-ProbLC, serve as input, resulting in two network variants based on *LuNet-lite-JO*. The obtained results of both network variants are combined with results achieved by the two network variants using the input configurations Mask-T-NoLC and Mask-T-ProbLC (as presented in Section 6.2.5) forming an ensemble and resulting in ENS-TS-LC. To answer the question (b), all experiments of the experiment set called Inves-ComTS were conducted, in which all tiled and multi-scaled patches of small polygons are combined by the COM-TS approach presented in Section 4.2.3.3. Here, the input configurations Mask-Com-NoLC and Mask-Com-ProbLC in Tab. 5.8 are used, resulting in two types of inputs for the CNN. Another two network variants based on *LuNet-lite-JO* using these two types of inputs are trained and tested, and their results are again combined in an ensemble, leading to ENS-ComTS-LC. The results of ENS-T-LC in Section 6.2.5 serve as the baseline for comparison. Tab. 6.12 presents an overview of the results of ENS-TS-LC, ENS-ComTS-LC and the baseline.

Comparing the results between ENS-TS-LC and the baseline, the former one is a winner. The improvement caused by the integration of the scaling approach is between +0.6% (level III in Schleswig) and +2% (level II in MV) in terms of OA, and between +0.6% (level III in Schleswig) and +2.3% (level II in MV) in terms of the mean F1 score. Looking at the results delivered by ENS-ComTS-LC, they

outperform the ones of the baseline in Hameln and MV. Although they are slightly worse than the ones of the baseline in Schleswig, the absolute differences are very small (up to 0.4% in terms of OA and 0.5% in terms of the mean F1 score). The results of ENS-TS-LC outperform the ones of ENS-ComTS-LC in Hameln and Schleswig (up to +1.3% for OA and +2.2% for mean F1 score) and achieve very similar to the ones of ENS-ComTS-LC in MV. In conclusion, the scaling approach does help the land use classification, and the ensemble method ENS-TS-LC delivers better results in most cases, which is the answer to the question (a) in Section 5.3.2.9.

The primary goal of the multi-scale approach is to improve the classification of small polygons. Thus, it is quite interesting to see the changes for small polygons. Fig. 6.24 shows the differences between the results achieved by ENS-TS-LC and ENS-ComTS-LC and those achieved by the baseline as a function of object size. Tab. 6.13 presents the number of objects in each object size bin differentiated in Fig. 6.24. Looking at the OA in all test sites, the classification is improved by ENS-TS-LC in most object size groups (by up to +8%). In Hameln and Schleswig, objects smaller than 0.1*A are better identified, yet in MV there is a negative impact. As far as the mean F1 scores are concerned, ENS-TS-LC delivers also much better results than the baseline with improvements of up to +14% for most object sizes in the three datasets.

A similar behavior can be observed for objects larger than 0.1*A in the three test sites based on the differences between results of ENS-ComTS-LC and the baseline, but with a lower degree of improvement (up to +7% for OA and +12% for mean F1 score). However, for objects smaller than 0.1*A in the three test sites, there is a decrease of OA in Schleswig and MV (up to –4%), and of the mean F1 score in Hameln and Schleswig (up to –6%) over the three semantic levels. In conclusion, the classification of small polygons profits from the scaling approach, especially for objects having a size between 0.1*A and A, which is the answer to the question (b).

6.2.7. Evaluation on all Datasets

To answer the two questions raised in Section 5.3.2.10, the results achieved by ENS-TS-LC are used for evaluation, because they outperform the results delivered by other methods in most cases. To obtain further insights into the separability of classes as a function of the semantic level, an analysis of the results for all three datasets are carried out. Tab. 6.14 presents the F1 scores for all categories at all semantic levels.

Level I: at this level, the four classes can be separated easily in the three datasets, and the results in Hameln and Schleswig are superior than those in MV. In Hameln and Schleswig, class *water body* has the lowest F1 score compared to other classes, indicating a problem with that class. This may partly be due to the fact that there are very few samples of that class (about 2% and 3.3% in Hameln and Schleswig, respectively). Furthermore, the confusion matrix shows that about 14.4% of the samples of *water body* in Hameln are confused with *traffic*; the corresponding percentage for Schleswig is 21.7%. The reason for this confusion could be that both object types are very similar in shape and some land cover components (e.g. both are surrounded by *grass* and *tree* pixels). In addition, the main material (*water* vs. *sealed*) appears similar, and both types of objects may be occluded or affected by cast shadow.

In combination with the relatively small number of training samples for water, these problems apparently prevent the CNN from properly learning to differentiate these classes. In MV, class *traffic* has the lowest F1 score, and the corresponding objects are mainly misclassified as *water body*.

Level II: the results at this level are analysed according to the ancestor categories at level I. In Hameln, there are three level II sub-categories of ***settlement*** achieving F1 scores over 60% (*residential*, *industry* and *recreation*), whereas the corresponding number is two in MV and Schleswig (*residential* and *recreation*); obviously, it is more difficult to differentiate the other categories correctly. The main source of errors is a confusion between *mixed usage*, *industry* and *residential*. This may be due to their similar appearance and composition of land cover elements (cf. Fig. 6.25 a, 6.25 b, 6.25 c). Among the sub-categories of ***traffic***, *road traffic* and *path* are differentiated best (F1 scores ≥ 60% in all sites). *Parking lot* is frequently confused with *road traffic* and *industry*. It can be differentiated much better in Hameln than in Schleswig, where about 50% of the *parking lot* objects are erroneously assigned to *road traffic*.

Test Site	Method	Level I		Level II		Level III	
		OA [%]	mF1 [%]	OA [%]	mF1 [%]	OA [%]	mF1 [%]
Hameln	ENS-T-LC	93.4	89.7	79.4	62.3	75.4	53.3
	ENS-ComTS-LC	94.1	90.3	80.2	63.7	76.1	54.5
	ENS-TS-LC	**94.4**	**91.0**	**80.8**	**64.7**	**76.8**	**55.8**
Schleswig	ENS-T-LC	90.9	85.6	75.2	61.2	71.1	55.2
	ENS-ComTS-LC	90.7	85.2	74.9	60.7	70.7	54.9
	ENS-TS-LC	**91.9**	**87.4**	**76.1**	**62.4**	**71.7**	**55.8**
MV	ENS-T-LC	81.0	81.0	69.2	52.1	66.3	40.7
	ENS-ComTS-LC	**82.7**	**82.9**	**71.2**	**56.3**	**68.1**	**44.0**
	ENS-TS-LC	**82.7**	82.8	**71.2**	54.4	**68.1**	42.6

Table 6.12: Test results from different ensemble methods in Hameln, Schleswig and MV. Best results are shown in bold.

Test Site	Object Size					#Small Polygons	Pertentage in the whole dataset [%]
	< 0.1*A	0.1*A – 0.3*A	0.3*A – 0.5*A	0.5*A – 0.7*A	0.7*A – A		
Hameln	423	449	354	236	295	1757	59.7
Schleswig	261	447	367	298	381	1754	40.4
MV	170	215	172	136	188	881	31.0

Table 6.13: Number of small polygons as a function of the object size bin differentiated in Fig. 6.24 in Hameln, Schleswig and MV, and the corresponding percentage in the whole dataset. A is the area of a patch of 256 x 256 pixels ($A = 2621\ m^2$at GSD of 20 cm in Hameln and Schleswig and $A = 533\ m^2$ at GSD of 10 cm in MV).

Figure 6.24: Differences of OA (left column) and mean F1 score (right column) between the results of ENS-TS-LC and the baseline, and between the results of ENS-ComTS-LC and the baseline, achieved by *LuNet-lite-JO*, as a function of object size with a focus on polygons smaller than size A, where A is the area of a patch of 256 x 256 pixels ($A = 2621\ m^2$at GSD of 20 cm in Hameln and Schleswig and $A = 533\ m^2$ at GSD of 10 cm in MV). **Solid bars**: differences between ENS-TS-LC and the baseline; **checked bars**: differences between ENS-ComTS-LC and the baseline. The number of objects per bin is shown in Tab. 6.13.

Level I				Level II				Level III			
category	**Hameln**	**Schleswig**	**MV**	**category**	**Hameln**	**Schleswig**	**MV**	**category**	**Hameln**	**Schleswig**	**MV**
settlement	**95.1**	**93.9**	**88.0**	*res.*	**87.3**	**87.2**	**80.1**	*res.use*	**89.8**	**88.4**	**81.6**
								ext. res.	39.6	**60.3**	0
				ind.	**76.7**	**59.2**	47.5	*fact.*	39.1	0	41.5
								busi.	**61.7**	**53.9**	0
								infra.	**61.1**	32.6	23.3
				mix	0	17.1	41.9	*mix*	0	17.1	41.9
				special	**55.0**	35.7	0	*special*	**55.0**	35.7	0
				rec.	**79.1**	**71.7**	**65.0**	*leis.*	0	**56.4**	0
								park	**78.3**	**71.9**	**66.8**
traffic	**95.5**	**92.1**	**79.9**	*ro.traf.*	**85.4**	**87.1**	**73.2**	*mo. road*	**87.4**	**90.9**	**73.7**
								traf. area	**72.9**	44.9	**50.3**
				path	**85.5**	**76.2**	**64.0**	*path*	**85.5**	**76.2**	**64.0**
				park.lot	**59.8**	42.4	0	*park.lot*	**59.8**	42.4	0
vegetation	**88.9**	**94.3**	**80.4**	*agr.*	**88.1**	**92.7**	**76.3**	*farm*	**85.1**	**93.1**	**83.0**
								garden	**71.8**	**86.6**	**62.3**
				forest	**83.0**	**89.3**	**79.7**	*h/s.wood*	**56.8**	47.9	**71.3**
								h&s.wood	21.7	**58.5**	0
				grove	**65.3**	48.2	46.4	*grove*	**65.3**	48.2	46.4
				moor	48.2	**61.3**	30.9	*moor*	48.2	**61.3**	30.9
water	**84.3**	**69.2**	**82.9**	*flow.wat.*	**82.5**	26.5	**84.5**	*flow.wat.bo*	**82.5**	26.5	**84.5**
				stag.wat.	10.0	**79.3**	**72.5**	*stag.wat.bo.*	10.0	**79.3**	**72.5**

Table 6.14: F1 scores in [%] of individual categories at all semantic levels achieved in ENS-TS-LC. Refer to Tab. 5.2 for the abbreviations of the categories. F1 scores of ≥ 50% are printed in bold font.

Figs. 6.25 b and 6.25 d show examples for two objects belonging to the classes *industry* and *parking lot* that have a very similar appearance; this degree of similarity between these object types may be the reason why they are confused frequently. Among the sub-categories of ***vegetation***, *agriculture* and *forest* are particularly well classified (F1 ≥ 75%) in all cases. The other sub-categories are more problematic. *Grove* is most frequently confused with *recreation* and *forest*, i.e. again with classes that have a similar appearance (cf. Figs. 6.26 a and 6.26 b). The category *moor* is mainly confused with *agriculture*.

Level III: at this level, some classes can be differentiated very well, e.g. *residential in use* or *motor road*, both with F1 scores larger than 70% in all cases. Although more than half of the categories achieve F1 scores larger than 50%, it is still more difficult to separate all categories than it is at other levels. To a certain degree this is due to the comparably small number of training samples for some classes, e.g. classes *stagnant water body* and *special usage*.

In summary, with an increasing number of categories from level to level, it becomes more and more difficult for a classifier to differentiate them, which is the answer to the question (a) in Section 5.3.2.10. At the finer levels, some categories are very similar in appearance and land cover composition (e.g. *industry* vs. *mixed usage*; *grove* vs. *forest*), which may make them difficult to be discerned. On the other hand, the scarcity of training samples also has a negative impact on some classes. The situation could be improved by increasing the number of training samples further, e.g. by enlarging the size of the area to be processed, or combining different data sources despite a domain gap. This would certainly be required if the classification at level III should achieve the level of accuracy required for operational use.

Influence of the object size

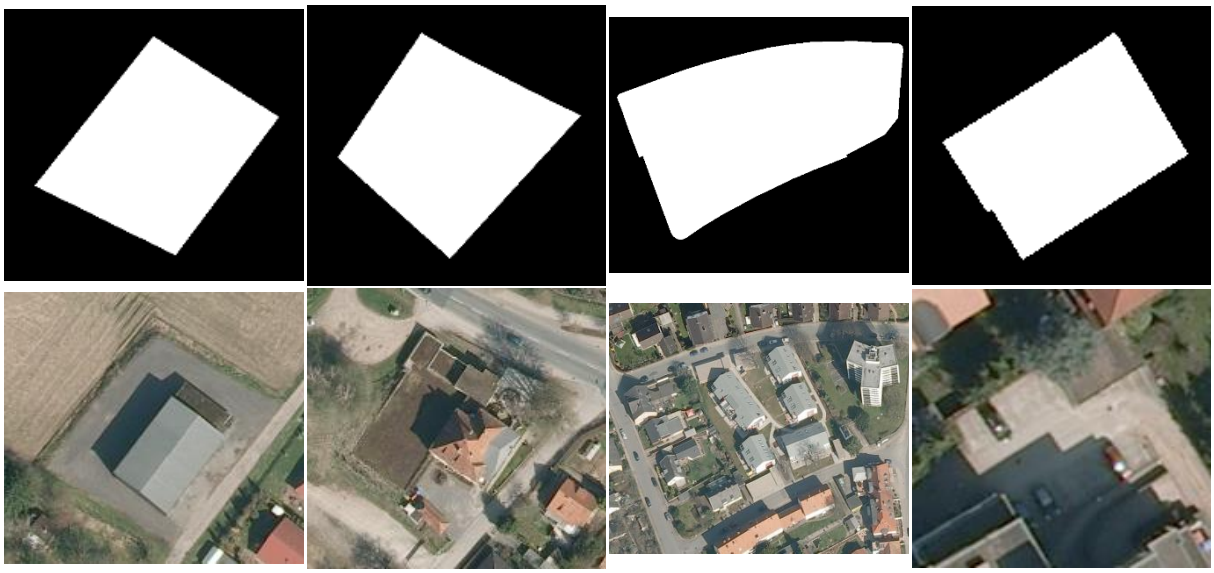
In (Yang et al., 2019; Yang et al., 2020; Yang et al., 2021), it has been noted that the object size has a large influence on the classification accuracy and mean F1 scores. Thus, in this thesis an analysis of that influence is carried out again on the basis of the results delivered by ENS-TS-LC in Hameln, Schleswig and MV. Fig. 6.27 shows the overall accuracy and mean F1 scores as a function of object size. The corresponding number of objects and average number of tiles per object, generated by the tiling approach in each object size group are presented in Tab. 6.15.

As far as the OA is concerned, in all datasets, a clear trend is visible: larger objects are better classified than smaller ones at each semantic level. In Hameln, even small objects can be detected with an OA better than 90% at level I, but the larger the object, the better the prediction becomes. The differences of OA between objects of different size are more pronounced at levels II and III. The most significant improvement of OA occurs between objects having a size between $A = 2621\ m^2$ (the area of a patch of 256 x 256 pixels at a GSD of 20 cm) and 2A and objects of size 2A to 3A (+ 15% at level II and +13% at level III). In Schleswig, there is a similar trend. It seems to be more difficult to classify objects smaller than $A = 2621\ m^2$ at level I. Over all levels, the biggest increase occurs between objects smaller than A and objects having an area between A and 2A (+8% at level I, +16% at level II and +14% at level III). In MV, due to a GSD of 10 cm, the area of a patch of 256 x 256 pixels is $A = 655\ m^2$. Clearly, the trend of an increasing OA for larger objects is observed again. The most prominent increases occur between objects having size between 2A and 3A and objects having an area more than 3A, which are +7% at level I, +13% at level II and +10% at level III.

Considering the mean F1 scores in all sites, the tendency to increase with the area can be observed at level I, similar to the case of OA. However, at level II the mean F1 scores do not change too much for different object sizes in Hameln and Schleswig, which is mainly due to the unbalanced class distribution, which affects both large and small objects. In MV, there is again a large improvement of mean F1 scores between objects having size between 2A and 3A and objects having an area more than 3A (+16%). At level III, the largest increase of mean F1 scores occurs between objects having a size between A and 2A and objects having size between 2A and 3A in Hameln (+9%) and in Schleswig (+10%), whereas it occurs between objects having size between 2A and 3A, and objects having an area more than 3A in MV (+14%).

Object Size	**Hameln**		**Schleswig**		**MV**	
	#Polygon	#avg. Tiles	#Polygon	#avg. Tiles	#Polygon	#avg. Tiles
< A	1757	1.4	1754	1.4	881	1.5
A – 2A	513	3.0	768	2.8	444	3.2
2A – 3A	222	4.6	393	4.5	283	5.2
> 3A	453	10.1	1430	15.5	1232	46.8

Table 6.15: Number of polygons (#Polygons) and average number of tiles per object (#avg. Tiles) generated by the tiling approach as a function of object size in Hameln, Schleswig and MV. $A = 2621\ m^2$at a GSD of 20 cm (Hameln and Schleswig) and $A = 533\ m^2$ at a GSD of 10 cm (MV).



a) *mixed usage* b) *industry* c) *residential* d) *parking lot*

Figure 6.25: Similar land use objects in category level II corresponding to *settlement* and *traffic* at level I. Upper row: binary images indicating the object shapes, bottom row: RGB orthophoto.

a) *recreation* b) *grove* c) *forest*

Figure 6.26: Similar land use objects in category level II corresponding to *settlement* and *vegetation* at level I. Upper row: binary images indicating the object shapes, bottom row: RGB orthophoto.

Figure 6.27: OA (left column) and the mean F1 score (right column) of land use classification achieved by ENS-TS-LC in each site as a function of object size given in units of A, which is the area of a patch of 256 x 256 pixels. $A = 2621\,m^2$at a GSD of 20 cm (Hameln and Schleswig) and $A = 533\,m^2$ at a GSD of 10 cm (MV). The number of objects and the corresponding average number of tiles per object in each object size bin are shown in Tab. 6.15.

This analysis makes clear that in addition to the number of training samples per class, the object size is another limiting factor for the success of the classification at all semantic levels. Larger objects are better classified in general, which is the answer to the question (b) in Section 5.3.2.10. At level I, the accuracies achieved for small objects (~85% - 95%) may still be in an acceptable range in Hameln and Schleswig, at higher semantic levels this problem becomes more crucial, e.g. leading to about 40% of false classifications for small objects in Schleswig. However, in MV the accuracies for objects smaller than 2A are lower than 80% at all levels, which poses a problem in a real application. It can be expected that enlarging the test areas or adding more training data will improve the performance in some degree.

6.2.8. Training on combined Datasets

Similarly to land cover classification, datasets are combined to investigate whether there is a benefit for land use classification. For that purpose, the Hameln and Schleswig datasets are combined. The four types of inputs, corresponding to the input configurations Mask-T-NoLC, Mask-T-ProbLC, Mask-S-NoLC, Mask-S-ProbLC of both datasets, are combined respectively, resulting in four network variants based on *LuNet-lite-JO*. These network variants are trained and tested, corresponding to the experiments of the experiment set ComHS-LU in Tab. 5.10. All classification results achieved by the four network variants are combined in an ensemble, leading to ENS-TS-LC-ComHS. To answer the question raised in Section 5.3.2.11, comparisons between ENS-TS-LC-ComHS and ENS-TS-LC are performed for Hameln and Schleswig. Tab. 6.16 presents an overview of the results achieved by ENS-TS-LC-ComHS and ENS-TS-LC.

Test site		**Category level**					
		Level I		**Level II**		**Level III**	
		OA [%]	mF1 [%]	OA [%]	mF1 [%]	OA [%]	mF1 [%]
Hameln	IND	94.4	91.0	80.8	64.7	76.8	55.8
	COM	94.2	89.7	81.2	64.5	77.1	55.7
	Δ	-0.2	-1.3	+0.4	-0.2	+0.3	-0.1
Schleswig	IND	91.9	87.4	76.1	62.4	71.7	55.8
	COM	92.3	87.0	76.2	61.7	72.1	56.1
	Δ	+0.4	-0.4	+0.1	-0.7	+0.4	+0.3

Table 6.16: Classification results in each test site. COM: results achieved by ENS-TS-LC-ComHS; IND: results achieved by ENS-TS-LC; Δ: difference between the results of COM and IND. Green color denotes an increase and red color denotes a decrease of metrics.

The difference between Hameln and Schleswig is mainly caused by different capturing seasons (spring in Hameln, summer in Schleswig), which should mainly affect vegetation objects, e.g. deciduous trees and agriculture. In Hameln, the OAs of the two training variants are close to each other for all three semantic levels, with an absolute difference smaller than 0.5%, and the results relying on the training on the combined dataset are slightly better. Nonetheless, the mean F1 score achieved by the training on the combined datasets is worse than the one achieved by individual training (up to –1.3% at level I). In Schleswig, the training on the combined dataset delivers slightly better classification results than the

individual training in terms of OA over all semantic levels (up to +0.4%). Similar to Hameln, there is also decrease of the mean F1 scores at levels I and II (up to –0.7%). However, a slight improvement of the mean F1 scores of +0.3% is observed at level III. In conclusion, the answer to the question raised in Section 5.3.2.11 is that adding more data seems to not help the land use classification too much.

6.2.9. Discussion

In the context of land use classification, a workflow of classifying land use objects, which are stored in the form of polygons in geospatial databases, in multiple semantic levels based on CNN is proposed. Due to the variations of polygons in size and shape, they are required to be converted to CNN patches of regular size. For that purpose, two approaches are proposed, namely tiling and scaling. The prediction of a polygon relies on the predictions for its patches. To guarantee a consistent prediction with the class hierarchy, two strategies are proposed. The first one is starting the predictions at the finest level and using a post-processing strategy called *fine-to-coarse* (F2C) to control the predictions at the coarser levels, whereas the other one uses the so-called *joint optimization* (JO) strategy to select the hierarchical tuple having the maximum joint class scores over all semantic levels as prediction. Due to the fact that the class distributions are imbalanced in the datasets, a focal-loss-style penalty term has been applied in the loss designed for the JO, with the expectation that the problem of the imbalance of distribution can be mitigated to a certain degree.

The results of experiments in three datasets have shown that the F2C approach delivers better predictions than the JO approach in the Hameln and MV datasets with a difference of up to +1.1% in terms of OA and of up to +2.9% in terms of the mean F1 score. In Schleswig, the JO approach outperforms the F2C approach only in terms of OA by up to 1%. This fact indicates that the JO strategy still has some limitations on the hierarchical land use classification, which is required to be further investigated. Turning the focus on the loss for JO, it has been found that a value larger than 0 to the weight of the penalty term can improve the classification performance, in particular the F1 score of the underrepresented classes.

The investigations of the impact of land cover information have shown that the best way to utilize the land cover information is by ensemble. Precisely speaking, image data and land cover posteriors of a polygon serve as individual input for two CNNs. The final prediction for that polygon shall be made by combining the predictions delivered by the two CNNs in an ensemble. Regarding to the scaling approach, another two CNNs using the scaled inputs based on image data and land cover posteriors need to be created. The final prediction of a polygon is made by combining the results delivered by the four CNNs in an ensemble.

To investigate whether adding more data from different domains help the classification or not, the Hameln and Schleswig datasets are combined. Four CNNs using different input configurations of the combined datasets are trained and tested. Their achieved classification results are combined in an ensemble to predict the land use of the polygons. The results achieved based on training on the combined dataset have shown that there is a slight improvement compared to the results based on individual training, but not too much, although there is domain gap between the two datasets. Further research is

required to investigate whether there is still more potential to improve the classification by combining datasets.

7. Conclusion and Outlook

7.1. Conclusion

In this thesis, a hierarchical deep learning framework based on CNN with the goal of the verification of geospatial databases with hierarchical object catalogue is proposed. Using high-resolution aerial images and derived products as input, a two-step classification procedure is applied: first, land cover information is determined at pixel level; afterwards, that information along with the aerial images and additional information such as height data is combined in a second classification process to determine the land use for every database object. This strategy is chosen because land use objects of a given class (e.g. *residential, agricultural*) are often composed of different land cover elements, so that the prediction of land cover delivers valuable information for determining land use.

In the context of land cover classification, the main contribution is the proposed two-branch encoder-decoder network *FuseNet-lite,* which utilizes the proposed learnable *skip-connections* to fuse the features in the encoder stage to those in the corresponding decoder stage. In *FuseNet-lite,* the first branch requires RGB image data and the second branch requires RID image data. Compared to the commonly applied elementwise addition skip-connections, the proposed one is able to learn the combination of features. Experiments using five different datasets show that CNNs with the learnable skip-connections outperform those with elementwise addition skip-connections in most cases in terms of OA and the mean F1 score. Particularly, underrepresented classes profit from the learnable skip-connections to a certain degree, for instance, class *clutter* in Hameln, Vaihingen and MV, *car* in Schleswig and *unpaved road* in MV. Their F1 scores are increased by 3% to 9%. Compared to CNN without skip-connections, the CNN with skip-connections improve the OA by 1% to 2.7% and the mean F1 scores by 1% to 8.2% in those five datasets.

Another contribution is the proposed extended focal loss to mitigate the problem of imbalanced class distribution. However, the results of the experiments using the Hameln and Vaihingen datasets have shown that the variant using the standard cross entropy performs better than the one using the proposed focal loss in some cases. This is contrast to the findings in (Lin et al. 2017). The reason may be due to the fact that the imbalance degree of the class distributions is much smaller than the one in (Lin et al. 2017). In comparison with state-of-the-art methods using the benchmark datasets of Vaihingen and Potsdam, the proposed *FuseNet-lite* delivers results on par with them: in terms of OA, the difference between the best results and the predictions of *FuseNet-lite* is within 1%. However, *FuseNet-lite* requires only about 1% or even smaller percentage of the parameters required by ResNet-based methods, for which the best performance has been reported on these datasets.

In the context of land use classification, the main contribution is the development of a CNN-based method for hierarchical land use classification. Due to the fact that many topographic databases contain land use information in different semantic levels of abstraction, from the view of the application, it is beneficial for NMAs to obtain classification results in multiple semantic levels. To achieve that goal, two network variants are proposed. The first one (*LuNet-lite-JO*), on the basis of a multi-task network *LuNet-lite-MT*, predicts land use at multiple semantic levels while guaranteeing the consistency with the

class hierarchy. In that CNN, a so-called joint optimization (JO) strategy is proposed to optimize the joint class scores of the hierarchical tuples. Another network variant called *LuNet-lite-F2C* starts with predictions at the finest level, and uses them to control the predictions at coarser levels by a post processing strategy called *fine-to-coarse* (F2C).

Because land use objects are represented as polygon in the geospatial databases, they need to be pre-processed to regular-sized patches which fit the input size of CNN. A polygon is represented in the form of combination of a binary mask, showing the polygon's shape, and the corresponding image data (e.g. RGB). Two approaches of patch generation are proposed. One is to split polygon into tiles whereas the other one is scaling. In the tiling approach, large polygons will have multiple representative patches whereas small polygons will have exactly one patch. In contrast to tiling, small polygons will have patches of multiple scales in the scaling approach and large polygons will have exactly one patch. Patches generated from tiling and scaling approaches serve as input for the CNN.

Experiments using three datasets show that the variant relying on JO strategy delivers better predictions in most cases than the multi-task network, but does not outperform the one relying on the F2C strategy. It has to be noted that one main drawback of *LuNet-lite-MT* is that its prediction may not be consistent with the class hierarchy, and in the F2C-based variant, if the prediction at the finest semantic level is wrong, the entire prediction is wrong and cannot be corrected. The experiments of investigating the impact of land cover information on land use classification show that the results relying on an ensemble of the predictions of the CNN variants using only land cover information and only image data are the best in most cases. The assessment of the impact of the scaling approach on land use classification shows that the results achieved by an ensemble of the predictions of the variants relying on the patches generated by scaling and tiling are the best. Compared to results only relying on the patches generated by the tiling approach, the classification of small polygons is improved by the ensemble predictions relying, by up to 10% in terms of OA and 14% in terms of the mean F1 score in the three datasets.

On the basis of the results of this ensemble, one can see that the object size is also an important factor influencing the classification. Generally speaking, the classes of large polygons are better identified than those of smaller ones. As large polygons have more patches than small polygons, the improved quality for large polygons may be due to the combination of more predictions. In addition, it is found that as the number of classes increases, the classification performance decreases, which poses a problem in a real application. One reason for that decrease is the low number of training samples for some classes. Therefore, one possibility to increase the performance is to increase the training data set by enlarging test areas. In this thesis, research on combining datasets so as to increase training data was performed. It has to be noted that these datasets have differences caused e.g. by different capturing seasons. The results relying on training on combined datasets show that there is slight improvement for the classification of the classes at the finest semantic level.

7.2. Outlook

CNN commonly have a large number of unknown parameters (weights in filter matric and in FC layers), which are learned from training data. To obtain reliable results, a sufficient number of correct and representative training samples is a pre-requisite (Kaiser et al., 2017). However, it is time-consuming and expensive to get a large number of training samples via manual labelling for a classification task. To reduce the effort of the manual labelling, yet still guarantee a high classification quality, there are some possibilities to consider:

- *Transfer learning*: this is a common strategy applied in deep learning, e.g. in (Yosinski et al., 2014). One can use a CNN with pre-trained weights, which are learned from a large dataset, to initialize the CNN and fine-tune it on a small dataset. Training from scratch on a small dataset may prevent training from converging to a good optimum in parameter space (corresponding to overfitting). Pre-training on a large dataset has already adapted the CNN well to the original task, and the classifiers seems to "remember" previously seen samples even if taken from a different domain. Therefore, transfer learning will start the training from a good point to adapt the CNN to fit the new small dataset.
- *Domain adaption*: another possibility is using multi-source data to train the CNN despite a domain gap. In land cover classification, the experiments in section 6.1.5 have shown that combining datasets contributes to the individual classification only to a very limited degree. However, it is still expected that the combination would give a larger contribution, if an appropriate strategy for domain adaption were designed. As these data come from different domains, it may be useful to design strategy to learn a more common representation for the same category from all available data, for instance, by adding regularization on the intermediate feature maps belonging the same category, e.g. (Wittich et al., 2019).
- *Label noise*: it is also possible to consider the existing geospatial databases (land cover and land use), which can be used to derive class labels. However, a severe problem of such databases is that the available labels may be wrong (e.g. due to temporal changes or errors in the original acquisition). Kaiser et al. (2017) combined a huge amount of data from Google aerial images, Open Street Map and the Potsdam dataset from ISPRS 2D Labeling Challenge to obtain class labels for buildings and roads. They found that using large amount of data from multiple cities with inaccurate labels can nevertheless boost the performance of classification for both objects if a small "clean" data is available. However, they did not explicitly design strategy to tackle the problem of label noise in their CNN. Maas et al. (2019) proposed a label noise robust classifier based on random forests to mitigate the impact of wrong class labels of training samples. Therefore, it is very interesting to transfer the principles for dealing with label noise to CNN-based land cover and land use classification.

Currently, land cover is firstly determined by an independent CNN, and then it is combined with image data to serve as input for the subsequent land use classification. The experiments have shown that land cover classification results clearly support the classification of land use. However, in the current

framework the support is unidirectional (from land cover to land use). It is also expected that land use classification results could support the performance of land cover classification as well. Albert et al. (2017) proposed a higher-order CRF to classify land cover and land use simultaneously, and their results show that the accuracy of land cover is improved by the simultaneous classification. In the context of deep learning, Zhang et al., (2019) proposed a joint deep learning framework for land cover and land use classification. They use a multi-layer perceptrons (MLP) for land cover classification and a CNN for land use classification. The training of the MLP and the CNN is carried out simultaneously. The results show that there is a dependency between land cover and land use. In the case of simultaneous classification relying on CNN, an input image patch will have two types of reference: one single land use label and pixel-level land cover labels. To classify them simultaneously, a series of convolution and pooling layers could be shared to encode the input image, and then the CNN is divided into two branches, where one branch is input for a symmetric decoder to classify land cover at pixel level, and the other one consists of further convolution and pooling layers to obtain a 1D feature vector for land use classification. In that model, the strategy of exchanging information between land cover and land use should be investigated, so that a bidirectional support can be achieved.

The proposed deep learning framework relies on the object boundaries stored in a database. These boundaries are supposed to be correct and unchanged. However, the change of land use can lead to a change of the object boundaries. Thus, it is required to check the object boundaries in the verification process and possibly update the boundaries. For that purpose, one possibility is to use the land cover results, which can indicate a possible boundary change. This might be achievable by training a CNN to predict object boundaries, extending the works of (Marmanis et al., 2018) and (Volpi & Tuia, 2018) to the boundaries of land use objects.

References

Abadi, et al., 2015. Large-scale machine learning on heterogeneous systems. Available online (accessed 24/02/2021): https://www.tensorflow.org.

AdV, 2008. Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland. ALKIS®-Objektartenkatalog 6.0. Available online (accessed 24/02/2021): http://www.adv-online.de/GeoInfoDok/GeoInfoDok-6.0/Dokumente/

Albert, L., Rottensteiner, F., Heipke, C., 2014. Land use classification using conditional random fields for the verification of geospatial databases. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. II-4. pp. 1-7.

Albert, L., Rottensteiner, F., Heipke, C., 2015. An iterative inference procedure applying conditional random fields for simultaneous classification of land cover and land use. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. II-3/W5. pp. 369-376.

Albert, L., Rottensteiner, F., Heipke, C., 2016. Contextual land use classification: how detailed can the class structure be? International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XLI-B4, pp. 11-18.

Albert, L., Rottensteiner, F., Heipke, C., 2017. A higher order conditional random field model for simultaneous classification of land cover and land use. ISPRS Journal of Photogrammetry and Remote Sensing 130: 63-80.

Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaria, J., Fadhel, M., Al-Amidie, M., Farhan, L. 2021. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data 8 (53): 1-74

Audebert, N., Saux, B. L., Lefevre, S., 2016. Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. In: Asian Conference on Computer Vision, pp. 180-196.

Audebert, N., Saux, B. L., Lefevre, S., 2018. Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks. ISPRS Journal of Photogrammetry and Remote Sensing 140: 20-32

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): 2481-2495.

Banzhaf, E., Hoefer, R., 2008. Monitoring urban structure types as spatial indicators with cir aerial photographs for a more effective urban environmental management. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 1 (2): 129-138.

Barnsley, M. J. & Barr, S. L., 1996. Inferring urban land use from satellite sensor images using kernel-based spatial reclassification. Photogrammetric Engineering & Remote Sensing 62 (8): 949–958.

Barnsley, M. J. & Barr, S. L., 2000. Monitoring urban land use by earth observation. Surveys in Geophysics 21 (2): 269-289.

Bishop, C., 2006: Pattern Recognition and Machine Learning. Springer, Singapore, 2006. ISBN: 9780387310732.

Brostow, G.J., Fauqueur, J., Cipolla, R., 2009. Semantic object classes in video: a high-definition groundtruth database. Pattern Recognition Letters 30 (2): 88-97

Campos-Taberner, M., Romero-Soriano, A., Gatta, C., Camps-Valls, G., Lagrange, A.,Saux, B. L., Beaupere, A., Boulch, A., Chan-Hon-Tong, A., Herbin, S., Randria-narivo, H., Ferecatu, M., Shimoni, M., Moser, G., Tuia, D., 2016. Processing of extremely high resolution LiDAR and RGB data: Outcome of the 2015 IEEE GRSS Data Fusion Contest. Part A: 2D contest. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 9 (12): 5547–5559

Chaurasia, A. and Culurciello, E., 2017. Linknet: Exploiting encoder representations for

efficient semantic segmentation. IEEE Visual Communications and Image Processing (VCIP), pp. 1–4.

Cicek, O., Abdulkadir, A., Lienkamp, S. S., Brox, T., Ronneberger, O., 2016. 3d U-Net: learning dense volumetric segmentation from sparse annotation. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 424–432.

Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L., 2017. DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 40, pp. 834-848.

Cheng, D., Meng, G., Xiang, S., Pan, C., 2017. FusionNet: Edge aware deep convolutional networks for semantic segmentation of remote sensing harbor images. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 10 (12): 5769-5783.

Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.-T., 2009. Nus-wide: A real-world web image database from national university of singapore. In: Conference on Image and Video Retrieval.

Comaniciu, D., Meer, P., 2002: Mean shift: a robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (5): 603-619.

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. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3213–3223.

Dai, J., Li, Y., He, K., Sun, J., 2016. R-FCN: object detection via region-based fully convolution networks. In: International Conference on Neural Information Processing Systems (NIPS'16) 30, pp. 379-387.

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

Deng, J., Ding, N., Jia., Y., Frome, A., Murphy, K., Bengio, S., Li., Y., Neven, H., Adam, H., 2014: Large-scale object classification using label relation graphs. In: European conference on Computer Vision (ECCV), Lecture Notes in Computer Science, Vol. 8689, Springer, Cham, pp. 48-64.

Diakogiannis, F., 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: 94-114.

Eigen, D., Fergus, R., 2015. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: IEEE International Conference on Computer Vision, pp. 2650-2658.

Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A., 2012. The PASCAL visual object classes challenge 2012 (VOC2012) results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. Accessed on 10.09.2021.

Freeman, W., Roth, M., 1995. Orientation Histograms for Hand Gesture Recognition. In: IEEE International Workshop on Automatic Face and Gesture Recognition. Zurich.

Fu, J., Liu, J., Wang, Y., Zhou, J., Wang, C., Lu, H., 2019. Stacked deconvolution network for semantic segmentation. IEEE Transactions on Image Processing (99): 1-13.

Gerke, M., Heipke, C., 2008: Image-based quality assessment of road databases. International Journal of Geographical Information Science 22 (8): 871-894.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. Journal of Machine Learning Research 9: 249-256.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press, ISBN: 9780262035613, Online available: http://www.deeplearningbook.org.

Gujrathi, A., Yang, C., Rottensteiner, F., Buddhiraju, K.M., Heipke, C., 2020: Improving the classification of land use objects using densing connectivity of convolutional neural networks. ISPRS Archives of Photogrammetry,

Remote Sensing and Spatial Information Sciences. Vol. XLIII-B2-2020, pp. 667-663.

Guo, Y., Liu, Y., Bakker, E.M. et al., 2018: CNN-RNN: a large scale hierarchical image classification framework. Multimedia Tools and Applications 77: 10251-10271.

Hazirbas, C., Ma, L., Domokos, C., Cremers, D., 2016. FuseNet: incorporating depth into semantic segmentation via fusion-based CNN architecture. In: Proceedings of the Asian Conference on Computer Vision, Vol. 2.

Haralick, R., Shanmugam, K., Dinstein, I., 1973. Textural features for image classification. IEEE Transactions on Systems, Man and Cybernetics. Vol. SMC-3 (6), pp. 610-621.

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

He, K., Gkioxari, G., Dollar, P., Girshick, R., 2017. Mask R-CNN. In: IEEE International Conference on Computer Vision (ICCV), pp. 2961-2969.

Helmholz, P., Rottensteiner, F., Heipke, C., 2014. Semi-automatic verification of cropland and grassland using very high resolution mono-temporal satellite images. ISPRS Journal of Photogrammetry and Remote Sensing 97: 204-218.

Hermosilla, T., Ruiz, L. A., Recio, J. A., Cambra-López, M., 2012. Assessing contextual descriptive features for plot-based classification of urban areas. Landscape and Urban Planning, 106 (1): 124-137.

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): 14780-14707.

Hu, H., Zhou, G.T., Dong Z., Liao Z., Mori G., 2016: Learning structured inference neural networks with label relationships. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2960-2968.

Hu, A., Cotter, F., Mohan, N., Gurau, C., Kendall, A., 2020. Probabilistic future prediction for video scene understanding. In European conference on Computer Vision (ECCV), pp. 767-785.

Hua, Y., Mou, L., Zhu, X.X., 2019: Recurrently exploring class-wise attention in a hybrid convolutional and bidirectional LSTM network for multi-label aerial image classification. ISPRS Journal of Photogrammetry and Remote Sensing 149: 188-199.

Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q., 2016: Deep networks with stochastic depth. In: Leib2 B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. Lecture Notes in Computer Science, Vol. 9908, Springer, Cham.

Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017: Densely connected convolutional networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4700-4708.

Ioffe, S., Szegedy, C., 2015. Batch Normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (ICML), pp. 448-456.

Ji, S., Wei, S., Lu, M., 2018. Fully convolutional networks for multisource building extraction from an open aerial and satellite imagery data set. IEEE Transactions on Geoscience and Remote Sensing 57 (1), 574–586.

Jiang, J., Zhang, Z., Huang, Y., Zheng, L., 2017. Incorporating depth into both CNN and CRF for indoor semantic segmentation. 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS).

Johnson, J.M., Khoshgoftarr, T.M., 2019. Survey on deep learning with class imbalance. Journal of Big Data 6 (27): 1-54.

Kaiser, P., Wegner, J.D., Lucchi, A., Jaggi, M., Hofmann, T., Schindler, K., 2017: Learning aeiral image segmentation from online maps. IEEE Transactions on Geoscience and Remote Sensing 55 (11): 6054-6068.

Kampffmeyer, M., Salberg, A., Jenssen, R., 2016. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. IEEE Conf. CVPR, pp. 680-688.

Krizhevsky, A., Sutskever, I., Hinton, G. E.,

2012. ImageNet classification with deep convolutional neural networks. In: International Conference on Neural Information Processing Systems (NIPS'12) 25 Vol. 1, pp. 1097-1105.

Kumar, S., Hebert, M., 2006. Discriminative random fields. International Journal of Computer Vision 68 (2): 179-201.

Längkvist, M., Kiselev, A., Alirezaie, M., Loutfi, A., 2016. Classification and segmentation of satellite orthoimagery using convolutional neural networks. Remote Sensing 8 (4): 329-350.

Lampert, C. H., Nickisch, H., Harmeling, S., 2014. Attribute-based classification for zero-shot visual object categorization. IEEE Transactions on Pattern Analysis and Machine Intelligence 36(3): 453–465.

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

Leiva-Murillo, J.M., Gomez-Chova, L., Camps-Valls, G., 2013. Multi-task remote sensing classification. IEEE Transactions on Geoscience and Remote Sensing 51 (1): 151-161.

Lin, M., Chen, Q., & Yan, S. (2013). Network in network. arXiv preprint arXiv:1312.4400.

Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P., 2017a. Focal loss for dense object detection. In: IEEE International Conference on Computer Vision (ICCV), pp. 2999-3007.

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017b. Feature pyramid networks for object detection. In: IEEE conference on computer vision and pattern recognition (CVPR), pp. 2117–2125.

Liu, S., Ding, W., Liu, C., Liu, Y., Wang, Y., Li, H., 2018. ERN: edge loss reinforced semantic segmentation network for remote sensing images. Remote Sensing 10 (9),1339.

Liu, Q., Kampffmeyer, M., Jenssen, R., Salberg, A.B., 2020. Dense dilated convolutions merging network for land cover classification. IEEE Transactions on Geoscience and Remote Sensing 58 (9): 6309-6320

Liu, W., Rabinovich, A., Berg, A.C., 2015. "Parsenet: Looking wider to see better," arXiv preprint arXiv:1506.04579.

Luus, F.P.S., Salmon, B.P., Bergh, F. Van Den, Maharaj, B.T.J., 2015. Multiview deep learning for land-use classification. IEEE Geoscience and Remote Sensing Letters 12 (12): 2448–2452.

Long, J., Shelhamer, E., Darrell, T., 2014. Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3431-3440.

Maas, A., Rottensteiner, F., Heipke, C., 2019. A label noise tolerant random forest for the classification of remote sensing data based on outdated maps for training. Computer Vision and Image Understanding188: 102782.

Maggiori, E., Tarabalka, Y., Charpiat, G., 2017a. Convolutional neural networks for large-scale remote-sensing image classification. IEEE Transactions on Geoscience and Remote Sensing 55 (2): 645-657.

Maggiori, E., Tarabalka, Y., Charpiat, G., Alliez, P., 2017b. High-resolution semantic labelling with convolutional neural networks. IEEE Transactions on Geosciences and Remote Sensing 55 (12): 7092-7103

Marmanis, D., Schindler, K., Wegner, J. D., Galliani, S., Datcu, M., Stilla, U., 2018. Classification with an edge: Improving semantic image segmentation with boundary detection. ISPRS Journal of Photogrammetry and Remote Sensing 135: 158–172.

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., Terzopoulos, D., 2021. Image segmentation using deep learning: a survey. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Milletari, F., Navab, N., Ahmadi, S.-A., 2016. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: Fourth International Conference on 3D Vision (3DV), pp. 565–571.

Montanges, A.P., Moser, G., Taubenböck, H., Wurm, M., Tuia, D., 2015. Classification of urban structural types with multisource data and structured models. In: IEEE Joint Urban Remote Sensing Event (JURSE), pp. 1–4.

Nair, V., Hinton, G. E., 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In: International Conference on Machine Learning, pp. 807-814.

Nguyen, U., Heipke, C., 2020. 3D pedestrian tracking using local structure constraints. ISPRS Journal of Photogrammetry and Remote Sensing 166: 347-358.

Noh, H., Hong, S., Han, B., 2015. Learning deconvolution network for semantic segmentation. In: IEEE International Conference on Computer Vision (ICCV), pp. 1520-1528.

Nogueira, K., Penatti, O.A.B., dos Santos, J.A., 2017. Towards better exploiting convolutional neural networks for remote sensing scene classification. Pattern Recognition 61 (12): 539–556.

Novack, T., Stilla, U., 2015. Discrimination of urban settlement types based on space-borne SAR datasets and a conditional random fields model. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences II-3/W4, pp. 143–148.

Othman, E., Bazi, Y., Alajlan, N., Alhichri, H., Melgani, F., 2016. Using convolutional features and a sparse autoencoder for land-use scene classification. International Journal of Remote Sensing 37 (10): 2149–2167.

Paisitkriangkrai, S., Sherrah, J., Janney, P., Hengel, A., 2016. Semantic labeling of aerial and satellite imagery. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 7 (9): 1-14.

Pan X., Shi, J., Luo, P., Wang, X., Tang, X., 2018. Spatial as deep: spatial CNN for traffic scene understanding. Proceedings of the AAAI Conference on Artificial Intelligence 32 (1): 7276-7283.

Postadjian, T., Bris, A.L., Sahbi, H., Mallet, C., 2017. Investigating the potential of deep neural networks for large-scale classification of very high resolution satellite images. ISPRS Annals of Phot., Rem. Sens. and Spat. Info. Sc. Vol. IV-1/W1, pp. 183-190.

Ren S., He, K., Girshick, R., Sun, J. 2015. Faster R-CNN: towards real-time object detection with region proposal networks. In: International Conference on Neural Information Processing Systems (NIPS'15) 28 Vol. 1, pp. 91-99.

Ren, Y., Zhang, X., Ma, Y., Yang, Q., Wang, C., Liu, H., Qi, Q.,2020. Full convolutional neural network based on multi-scale feature fusion for the class imbalance remote sensing image classification. Remote Sensing 12 (21). 3547.

Russakovesky et al., 2015. ImageNet large scale visual recognition challenge. International Journal of Computer Vision, Vol. 115, pp. 211-252

Sherrah, J., 2016. Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. ArXiv:1606.02585.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: International Conference for Learning Representations.

Song, S., Lichtenberg, S., Xiao, J., 2015. SUN RGB-D: A RGB-D scene understanding benchmark suite. In: Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

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): 1929-1958.

Sun, Y., Tian, Y., Xu, Y., 2019. Problems of encoder-decoder frameworks for high-resolution remote sensing image segmentation: Structural stereotype and insufficient learning. Neurocomputing 330 (10): 297-304.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp: 1-9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlen, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp: 2818-2826.

Voelsen, M., Bostelmann, J., Maas, A., Rottensteiner, F., Heipke, C., 2020. Automatically generated training data for land cover classification with CNNs using Sentinel-

2 Images. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XLIII-B3-2020, pp. 767-774.

Voelsen, M., Lobo Torres, D., Feitosa, R.Q., Rottensteiner, F., Heipke, C., 2021. Investigations on feature similarity and the impact of training data for land cover classification. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. V-3-2021, pp. 181-189.

Volpi, M., Tuia, D., 2017. Dense semantic labeling of sub-decimeter resolution images with convolutional neural networks. IEEE Transactions on Geoscience and Remote Sensing 55 (2): 881-893.

Volpi, M., Tuia, D., 2018. Deep multi-task learning for a geographically-regularized semantic segmentation of aerial images. ISPRS Journal of Photogrammetry and Remote Sensing 144: 48-60.

Walde, I., Hese, S., Berger, C., Schmullius, C., 2014. From land cover-graphs to urban structure types. International Journal of Geographical Information Science 28 (3): 584–609.

Wang, Y., Ding, W., Zhang, R., Li, H., 2021. Boundary-Aware Multitask Learning for Remote Sensing Imagery. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 14: 951-963

Wang, H., Wang, Y., Zhang, Q., Xiang, S., Pan, C., 2017. Gated convolutional neural network for semantic segmentation in high-resolution images. Remote Sensing 9 (5), 446.

Wegner, J.D., Rottensteiner, F., Gerke, M., Sohn, Gunho, 2017. The ISPRS labelling challenge. Online Available: http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html (accessed 11/03/2021).

Wittich, D., Rottensteiner, F., 2019. Adversarial domain adaption for the classification of aerial images and height data using convolutional neural networks. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. IV-2/W7, pp. 591-598.

Wittich, D., 2020. Deep domain adaptation by weighted entropy minimization for the classification of aerial images. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. V-2-2020, pp. 591-598.

Xie, S.N., Tu, Z.W., 2017. Holistically-Nested Edge Detection. International Journal of Computer Vision, Vol. 125 (3), pp. 3-18.

Yang, C., Rottensteiner, F., Heipke, C., 2018: Classification of land cover and land use based on convolutional nerual networks. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. IV-3, pp. 251-258.

Yang, C., Rottensteiner, F., Heipke, C., 2019: Towards better classification of land cover and land use based on convolutional nerual networks. In: ISPRS Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XLII-2/W13, pp. 139-146.

Yang, C., Rottensteiner, F., Heipke, C., 2020a. Investigations on skip-connections with an additional cosine similarity loss for land cover classification. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. V-3-2020, pp. 339-346.

Yang, C., Rottensteiner, F., Heipke, C., 2020b. Exploring semantic relationships for hierarchical land use classification based on convolutional neural networks. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. V-2-2020, pp. 599-607.

Yang, C., Rottensteiner, F., Heipke, C., 2021: A hierarchical deep learning framework for the consistent classification of land use objects in geospatial databases. ISPRS Journal of Photogrammetry and Remote Sensing 177: 38-56.

Yang, Y., Newsam, S., 2010. Bag-of-visual-words and spatial extensions for land-use classification. In: 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 270–279.

Yosinski, J., Clune, J., Bengio, Y., Lipson, H., 2014. How transferable are features in deep neural networks? In: International Conference

on Neural Information Processing Systems (NIPS'14) 27, Vol. 2, pp. 3320-3328.

Zhang, C., Atkinson, P.M., 2016. Novel shape indices for vector landscape pattern analysis. International Journal of Geographical Information Science 30: 2442–2461.

Zhang, C., Sargent, I., Pan, X., Li, H., Gardiner, A., Hare, J., Atkinson, P.M., 2018. An object-based convolutional neural networks (OCNN) for urban land use classification. Remote Sensing of Environment 216: 57-70.

Zhang, C., Sargent, I., Pan, X., Li, H., Gardiner, A., Hare, J., Atkinson, P.M., 2019. Joint deep learning for land cover and land use classification. Remote Sensing of Environment 221: 173-187.

Zhang, Z., Liu, Q., Wang, Y., 2018. Road extraction by deep residual u-net. IEEE Geoscience and Remote Sensing Letters 15 (5), pp. 749–753.

Zheng, X., Huan, L., Xia, G.S., Gong, J., 2020. Parsing very high resolution urban scene images by learning deep ConvNets with edge-aware loss. ISPRS Journal of Photogrammetry and Remote Sensing 170: 15–28.

Zhou, Y., Chellappa, R., 1988. Computation of optical flow using a neural network. In: IEEE International Conference on Neural Networks, Vol. 2, pp. 71 – 78.

Zhu, X.X., Tuia, D., Mou, L., Xia, G.S., Zhang, L., Xu, F., Fraundorfer, F., 2017. Deep learning in remote sensing: a comprehensive review and list of resources. IEEE Geoscience and Remote Sensing Magazine 5 (4): 8-36.

Curriculum Vitae

Personal Information

Name	Chun Yang
Date and place of birth	21.10.1987 in Sichuan, China

Work Experience

Nov. 2016 – Dec. 2021	Leibniz University Hanover Institute of Photogrammetry and GeoInformation *Research Associate*
May. 2015 – Sep. 2016	Viscoda GmbH, Hannover, Germany *Software Engineer*
Dec. 2011 – Mar. 2015	EdgeWave GmbH, Aachen, Germany *Electronics Engineer*

Education

2009 – 2011	Ruhr University Bochum, Germany Communication Technology *Master of Science*
2008 – 2009	Aachen University of Applied Science, Germany Mechatronics *Bachelor of Engineering*
2005 – 2009	Tongji University, Shanghai, China Mechatronics *Bachelor of Engineering*
2003 – 2005	Nanchong High School, Sichuan, China

Acknowledgements

I want to express grateful thanks to all persons who has helped me and contributed to the success of this dissertation.

First and foremost, my deep gratitude goes to my supervisor Prof. Dr. techn. Franz Rottensteiner for giving me the trust and the freedom to develop my research ideas and for his guidance, advise and expertise throughout my studies and my doctorate. As chair of the Institute of Photogrammetry and GeoInformation (IPI), I thank Prof. Dr.-Ing. habil. Christian Heipke for giving me the opportunity to conduct my research and to do my PhD at the IPI. During my entire PhD career, he has offered me valuable and professional guidance and advise to my research. Besides, I thank Prof. Dr.-Ing. habil. Monika Sester and Prof. Dr.-Ing. habil. Michael Ying Yang for acting as referees of this dissertation and for providing valuable remarks.

I also want to express my gratitude to my friends. In the first place, I want to thank Dr.-Ing. Liao W.T. for his supervising and guidance during the time in TNT, which allowed me to start my PhD, and Dr.-Ing. Chen L. for letting me know and join IPI. Back to 2014, I thank Ms. Cheng for the values and discussions she has shared with me. I thank my dearest friends known since 2012. Our group has been enlarged by extending each single man to a family, which is now called SixFamily. My deep thanks go to their supports in terms of finance, mentality, friendship and love. In particular, they are Guo X.Y., Hu W., Li C.B., Li J.H., Dr.-Ing. Lin G., Dr.-Ing. Song J., Sun T.H., Tang X.J., Wen X.H., Wu J.Q. Furthermore, many friends have offered me the encouragement during my tough time, particularly they are Cao W., Feng A.G., Prof. Dr. Li J.S., Li M., Shi H.Y., Yuan J.Y. etc. I also want to thank my startup cooperation partner Zhou B.W. for his trust and understanding. Many friends in Hanover have helped me in the last few years and offered me a very nice social living environment. Some of them are Chen Y.J., Dr.-Ing. Cheng H., Dr.-Ing. Feng Y., Gu Y., Han S., Dr. Kang J.H., Lin Q.W., Dr.-Ing. Ren L., Su J.Y., Dr.-Ing. Wang X., Wu F.F., Yang J., Zhao X.R., Dr. Zeng Z. etc. (The list will be very long if I put all of them here. Each of them deserves my thanks from my bottom heart!).

I also want to express my gratitude to colleagues from the IPI. They have offered me a very nice working environment with their trust and tolerance. In particular, I thank Mirjana for the kind cooperation w.r.t the project seminars. I thank Claudia for her solutions related to organizations at IPI. I thank Amadeus, Uwe Bolte, Uwe Breitkopf for their IT supports. I also thank the boulder group for offering me fun, and particularly, they are Andreas, Anne, Christian, K., Dennis, Mareike, Mirjana etc. Of course, the breakfast group also deserves my deep thanks and they are Max, Philipp, Sönke etc.

Finally, I cordially thank my parents for their loving support and faith throughout my life. I thank my grandmother for sharing me many stories about her generation, which enriched my adolescent life. I thank my dear younger sister Yang R. for her caring of my parents during the time when I am in Germany. I also thank my dear cousins for their help and caring of my family, and particularly, they are Huang L., Yuan X.X, Yuan J.T., Wu, Z.L., Wu Y.L.