

Improving the Classification of Land use Objects using Dense Connectivity of Convolutional Neural Networks

A Dissertation
Submitted in partial fulfillment of
the requirements for the degree of
Master of Technology
by

Aishwarya Gujrathi
(183310004)

Supervisors:

Prof. Krishna Mohan Buddhiraju

and

Prof. Christian Heipke, M. Sc. Chun Yang (Leibniz Universität Hannover)



Centre of Studies in Resources Engineering
Indian Institute of Technology Bombay
Mumbai 400076 (India)

31 May 2020

Dedicated to my parents for their unconditional love and support...

Approval Sheet

This dissertation entitled “Improving the Classification of Land use Objects using Dense Connectivity of Convolutional Neural Networks ” by Aishwarya Gujrathi is approved for the degree of Master of Technology.

Examiners

Supervisor (s)

Chairman

Date: _____

Place: _____

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I declare that I have properly and accurately acknowledged all sources used in the production of this report. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: 31 May 2020

Aishwarya Gujrathi
(183310004)

Abstract

Land use is an important variable in remote sensing which describes the functions carried out on a piece of land in order to obtain benefits and is especially useful to the personnel working in the fields of urban management and planning. The land use information is maintained by national mapping agencies in geo-spatial databases. Commonly, land use data is stored in the form of polygon objects; the label of the object indicates land use. The main goal of classification of land use objects is to update an existing database in an automatic process. Recently, Convolutional Neural Networks (CNN) have been widely used to tackle this task utilizing high resolution aerial images (and derived data such as digital surface model). One big challenge classifying polygons is to deal with the large variation in their geometrical extent. To overcome this challenge, we adopt the method of cropping proposed in [19] to decompose polygons into regular patches of fixed size. The decomposition leads to two sets of polygons: small and large, where the former suffers from a lower identification rate. This thesis proposes CNN methods which incorporate dense connectivity and integrate it with intermediate information via global average pooling to improve land use classification, mainly focusing on small polygons. Different network variants are presented by incorporating intermediate information via global average pooling from different stages of the network. The proposed methods are experimented on two sites, Hameln and Schleswig (Germany); experiments show that the dense connectivity and integration of intermediate information has a positive effect not only on the classification accuracy on the whole but also on the identification of small polygons.

Table of Contents

Abstract	ix
List of Figures	xiii
List of Tables	xv
1 Introduction	1
1.1 Motivation	2
1.2 Contribution	3
1.3 Outline	3
2 Literature Review	5
2.1 Traditional Methods	5
2.1.1 Kernel Based Techniques	6
2.1.2 Region Based Techniques	7
2.1.3 Object Based Techniques	9
2.2 Land-use classification based on Conditional Random Fields	11
2.3 Land-use classification based on Convolutional Neural Networks	14
2.3.1 LiteNet	15
2.3.2 LuNet	16
3 Theoretical Foundation	19
3.1 Background	19
3.1.1 Perceptron	20
3.1.2 Multi layer Perceptron	21
3.1.3 Forward and Back propagation	22
3.2 Convolutional Neural Networks	23
3.2.1 Convolution Layer	24
3.2.2 Pooling Layer	25
3.2.3 Fully connected layer	25

3.2.4	Non-linearity	26
3.2.5	Batch Normalization	27
3.2.6	Loss function	28
3.2.7	Optimization algorithms	29
3.2.8	Regularization	30
3.3	Network Architectures	31
3.3.1	LeNet	31
3.3.2	AlexNet	32
3.3.3	ResNet	33
4	Problem Statement	35
4.1	Problem statement	35
5	Methodology	37
5.1	Patch preparation	37
5.2	Dense connectivity	38
5.3	DenseLuNet	39
5.4	Network Variants	40
5.5	Inference of polygons	41
6	Results and Discussion	43
6.1	Datasets and Test Setup	43
6.1.1	Datasets	43
6.1.2	Test setup	43
6.2	Evaluation of Land Use Classification	45
6.2.1	Evaluation and comparison of network variants	45
6.2.2	Effectiveness of using global average pooling	46
6.2.3	Influence of the object size	46
6.2.4	Analysis of confusion matrix	47
7	Conclusion and Outlook	49
A	Appendix	51
A.1	Programming	51
A.2	Confusion Matrices	52
	References	55
	Acknowledgements	65

List of Figures

2.1	Adjacency events within a 3×3 kernel window [4]	6
2.2	Land cover map of Graphtown [4]	8
2.3	Exploded representation of Graphtown [3]	8
2.4	Graph Visualisation of Graphtown for spatial relation 'Adjacency'[4]	9
2.5	Location of the city of Sagunto [9]	10
2.6	CRF consisting of a land cover layer (c) and land use layer (u) [2]	12
2.7	Image representing super-pixels for land cover classification (left) and land use objects for land use classification (right) [2]	13
2.8	Architecture of LiteNet-O [18]	15
2.9	Architecture of LuNet [19]	17
2.10	ROI location and feature extraction [19]	18
3.1	The illustration of a perceptron computing weighted sum of inputs and applying an activation function on it	20
3.2	Neural network processing	21
3.3	Convolution layer illustrating one convolution filter outputs one feature map	24
3.4	The illustration of a spatial pooling operation with a filter size of 2×2 by a stride of 2 in the high direction, and 2 in the width direction	25
3.5	Nonlinear functions (a) Sigmoid function (b) Tanh function (c) ReLU function (d) Leaky ReLU function	26
3.6	Batch Normalization Algorithm	28
3.7	Architecture of LeNet-5 [14]	32
3.8	Architecture of AlexNet [13]	33
3.9	Architecture of a residual block [8]	33
3.10	The architecture of ResNet-18	34
5.1	The figure represents two database objects, the left image represents binary object mask and right image represents the corresponding RGB image patch from digital orthophoto	38

5.2	A 3-layer dense block with n input channels and k growth rate. Please refer to texts for the abbreviations.	39
5.3	Structure of Two-Branch-Convolution	40
5.4	The architecture of DenseLuNet. TL: Transition layer, DenseBlock: cf. Figure 5.2, Two-Branch-Convolution: cf. Figure 5.3	41
5.5	The architecture of DenseLuNet-2	42
6.1	Confusion Matrix of DenseLuNet-12 on Hameln dataset	47
6.2	Confusion Matrix of DenseLuNet-12 on Schleswig dataset	48
A.1	Confusion Matrix of LuNet on Hameln dataset	52
A.2	Confusion Matrix of DenseLuNet on Hameln dataset	52
A.3	Confusion Matrix of DenseLuNet-1 on Hameln dataset	52
A.4	Confusion Matrix of DenseLuNet-2 on Hameln dataset	53
A.5	Confusion Matrix of LuNet on Schleswig dataset	53
A.6	Confusion Matrix of DenseLuNet on Schleswig dataset	53
A.7	Confusion Matrix of DenseLuNet-1 on Schleswig dataset	53
A.8	Confusion Matrix of DenseLuNet-2 on Schleswig dataset	54

List of Tables

2.1	List of descriptive features extracted at different object aggregation levels	11
6.1	Class distribution of Hameln and Schleswig. The table lists number of small polygons, large polygons and total number of polygons.	44
6.2	Results of land use classification. Network variants (cf. section 5.4). F1: F1 score, OA: Overall Accuracy, both evaluated on the basis of objects. Best scores are printed in bold.	45
6.3	Results of land use classification represented separately for large, small and all polygons (cf. Table 6.2). The results are provided for all the network variants on Hameln and Schleswig dataset. The number of polygons in each set is given in parenthesis.	47

Chapter 1

Introduction

Land use is an important variable in remote sensing which describes the functions carried out on a piece of land in order to obtain benefits. Land cover refers to the physical material on the surface of a given parcel of land. Land use classification assigns labels to larger functional units or spatial entities [18, 19, 20, 21]. Land cover can be derived from spectral characteristics of remote sensing data [4, 12, 5], but it is not true for land use. Land use is an abstract concept [4], a combination of social, economical and cultural factors.

In the region of central Europe, the government surveying authorities maintain geospatial database of property boundaries. The semantic information in such a database become outdated quickly as the property owners are not obliged to inform the government of change in land use. Thus, a system is required to analyse the change in land use of the property object stored in geospatial database. This can be done by automatic derivation of semantic information from aerial images [2]. The derived information is checked against the current information stored in the database and thus a database update is performed.

Recent works on land use classification are based on Convolutional Neural Networks (CNN) [18, 19, 20, 21]. CNN provide better results than the traditional classifiers that take hand-crafted features as input. Traditional approaches for land use classification require hand-crafted features derived from image data, and then apply a supervised classifier such as Random Forests to deal with these features. However, the methods incorporating hand-crafted features are strenuous and time consuming. CNN provide a framework in which features can be learned from training data, which explains much of the success of CNN in classification.

1.1 Motivation

The rapid progress in remote sensing technology has resulted in a bulk of images of the earth surface taken by satellites, airplanes or drones, with different imaging modalities. With the large availability of data, the focus shifts to the automatic extraction of valuable information. Deep learning is known to provide impressive results when large amount of data is available, and is currently being used in many remote sensing applications [22]. Also, with the rapid growth of Graphic Processing Units (GPUs), the training and testing time has been largely improved.

Land use classification is an important task in remote sensing, especially useful to the personnel working in the fields of urban management and planning. Geospatial land use database contain important information with high benefits for the above mentioned users, but fast changes in land use due to urban growth and land use conversion makes the geospatial databases outdated quickly. The objects stored in geospatial database are represented in the form of polygon objects, with the object label indicating land use. The polygon object can be of any size, but CNN takes a fixed size input. Therefore, this leads to two types of polygons: the polygon which fits into the window size of CNN is called a small polygon, the polygon which is larger than the window size of the CNN is called a large polygon.

One of the challenges faced while classifying polygon objects is because of their large variation in terms of geometric extent. For example, road objects are typically thin and long, but residential objects may be very large or small in size and of arbitrary shape. As mentioned earlier, CNN requires a fixed size input patch. Therefore, the large polygons must be decomposed into small patches. Two strategies proposed for patch preparation in [18, 19] are: cropping and rescaling. The cropping strategy is differentiated into two scenarios: *Small polygons*: If the polygon object fits into the window size of CNN, it is placed in the window such that it's centre coincides with polygon's centre. *Large polygons*: If the polygon object does not fit into the window size of CNN, it is further split into tiles of size equal to window size of CNN. In the rescaling strategy, the large polygons are rescaled using bilinear interpolation so that they fit into the window size of polygon. The rescaling approach causes loss of information, therefore, cropping strategy is used for patch preparation in this thesis. The object shape is represented in the form of a binary mask, where the area inside the polygon object is represented by 255 and the area outside the object is represented by 0.

Analysing the result of land use classification of polygons presented in [19], it is observed that the small polygons are hard to be classified. This can be caused because of the following reasons: (i) One problem of CNN is that as input passes through many layers of a neural network, it can vanish by the time it reaches the end of the network. (ii) The final 1-D feature vector before classification may not capture valid information of the small polygons due to many convolution and pooling operations. DenseNet [10] solves the problem of data loss by creating short paths from early layers to later layers, maximizing the data flow through the network. Also, adding intermediate information from different stages of the just before classification could be helpful in improving the results.

1.2 Contribution

The goal of this thesis is to build on the methods proposed in [19] with the aim of improving the classification of land use objects, mainly focusing on small polygons. In particular, the scientific contributions of this thesis can be summarized as follows:

1. To propose a network architecture incorporating dense connectivity [10] that strengthens information flow to improve the land use classification. The key is to create short paths from early layers to later layers, maximizing the data flow through the network.
2. Applying global average pooling (GAP) [15] at different stages of the network, resulting in many network variants, and utilize it as intermediate information in the classification process, to compensate the data loss caused by the many pooling operations in the network.
3. To conduct an extensive set of experiments to compare these network variants, and to highlight the benefits and drawbacks of the proposed methods.

All the networks were implemented using tensorflow framework [1]. GPU with specifications (Nvidia GeForce GTX 1080 TI, 11GB) was used to accelerate training and inference.

1.3 Outline

The content of the thesis has been split into the following sections.

- Chapter 2 is dedicated to literature review, presenting traditional land use classification methods such as Kernel based, Region based and Object based techniques. More recent Conditional Random Field (CRF) based and CNN based land-use classification methods are also described.
- Chapter 3 presents a short primer on the concepts of Deep Learning, particularly Convolutional Neural Networks, that will come handy in understanding the methodology of this thesis. The chapter starts with a discussion on CNN, different layers of CNN and concepts related to training of the CNN such as loss functions, Optimization algorithms and Regularization.
- Chapter 4 presents the drawbacks of the state-of-the-art in land use polygon classification and presents the problem that we set out to solve in this thesis.
- Chapter 5 presents our neural network architecture called "DenseLuNet" for land use classification, along with its three variants.
- Chapter 6 compares the result of the proposed CNN with the state of the art. Also, a comparison among the network variants is provided.
- Chapter 7 presents a conclusion of work presented and also provides an outlook for further research.

Chapter 2

Literature Review

Land use classification has been an important area in remote sensing. Earlier, the spatial resolution of the satellite images was coarse (eg., Multispectral Scanning Systems (MSS) aboard Landsat 1-5 having spatial resolution 79 x 57 meters), and was inadequate for performing land cover/land use classification. In the current times, it is possible to obtain satellite images having spatial resolution better than 1 meter (eg., GeoEye-1 having 0.41meter panchromatic, 1.65-meter multispectral resolution). It is also possible to obtain images using UAV's with pixel dimension in the order of centimeters. Earlier, image processing techniques based on separate feature extraction and classification pipelines were used for Land use/land cover classification. With the availability of large quantity of data (though data annotation is an issue), techniques based on neural networks are being used for Land cover/Land use classification.

2.1 Traditional Methods

Land cover is the physical material present on a piece of land (eg., water, grass, concrete etc.). Land use is the human activity that takes place on that land (eg., commercial, industrial, residential etc.). Classification of land cover is simple because there is a direct relationship between land cover and exitant spectral reflectance, but land use is an abstract concept. Land use is a combination of economic and cultural factors and is usually of great interest. Usually, the categories of land use have a spectrally-distinct land cover type arranged in a characteristic spatial pattern that enables it's recognition using remotely-sensed images.

The technique suggested in [4] for land use classification is to divide the classification procedure into two stages: the first being semantic segmentation of the image for

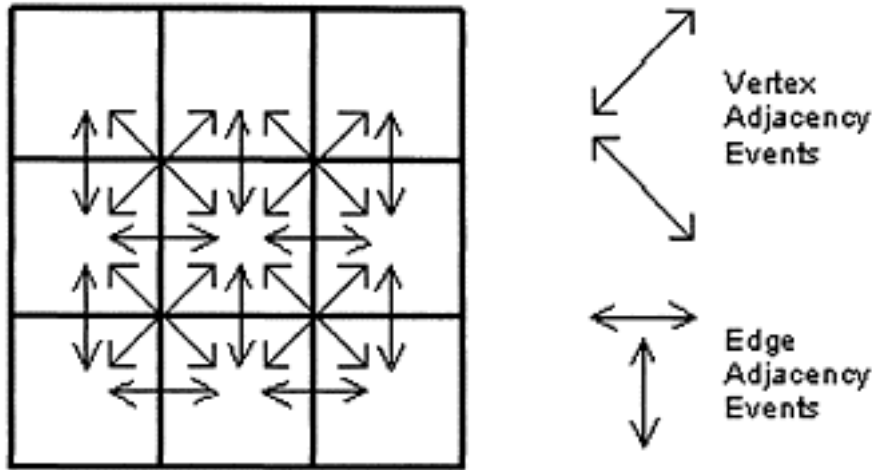


Figure 2.1: Adjacency events within a 3×3 kernel window [4]

land cover classification; the second being land use classification based on the spatial pattern of land cover. The first stage can be performed by a number of techniques ranging from supervised or unsupervised per-pixel classification algorithms to artificial neural networks. The disadvantage of such a two stage process is that the accuracy of the land use classification depends on the accuracy of land cover classification, i.e., error in the first stage is propagated through the second stage. [12] and [5] investigated segment-based land use classification. Segments are obtained by spectral classification. Spatial information of segments such as size, neighbours etc., are used for rule-based classification of image segments into land use categories. An interesting work on land use object classification combining high spatial resolution imagery, LiDAR data and cadastral plots is given in [9]. Land use objects are characterised by image based, geometric and contextual hand crafted features.

The traditional methods of inferring land use from characteristic spatial arrangement of land cover types can be broadly divided into three methods: (i) Kernel based techniques, (ii) Region based techniques and (iii) Object based techniques.

2.1.1 Kernel Based Techniques

In kernel-based techniques, an $n \times m$ pixel window or kernel is used to convolve the image to be classified, to find the relative frequency and spatial arrangement of land cover labels. In [4], their method SPARK (SPATial Re-classification Kernel) determines an adjacency matrix M , where every element M_{ij} of the matrix represents the frequency of pixels belonging to land cover class i that are adjacent to pixels belonging to land cover class j . The adjacency events in a kernel of size 3×3 is shown in figure 2.1.

The matrix M is compared to the matrix T_k , where T_k is the template matrix for every land use class k . The equation 2.1 is used to calculate the similarity index between the current position of the kernel and the land use categories.

$$\Delta_k = 1 - \sqrt{\frac{1}{2N^2} \sum_{i=1}^C \sum_{j=1}^C (M_{ij} - T_{kij})^2} \quad (2.1)$$

where N is the total number of adjacency events (for a 3×3 kernel in Fig. 2.1, $N = 20$), C is the number of land cover classes, M_{ij} is an element of the adjacency event matrix and T_{kij} is the corresponding element in the template matrix for land use class k . The central pixel of the window is assigned the land use label k for which Δ_k is maximum. Kernel based techniques are easy to implement but suffers from following disadvantages: smoothing of boundaries between land use/land cover parcels and determining the size of the kernel beforehand.

2.1.2 Region Based Techniques

The region based techniques work on a group of pixels, representing a land cover parcel. Since the focus is on parcels, additional information such as morphological properties (size and shape of the parcels) and spatial and structural relationships (distance, direction, adjacency, containment, etc.,) can be determined. Region based techniques are of greater theoretical and computational complexity when compared to kernel-based techniques.

In [3], SAMS (Structural Analysis and Mapping System), which is a structural pattern-recognition system has been described. From the thematically-labeled regions (land cover parcels) present in the categorical raster-format, structural information is derived. Morphological data (e.g., area, perimeter, shapes etc.,) and spatial and structural relations are populated in a graph data model, known as XRAG (eXtended Relational Attribute Graph), represented by the tuple:

$$XRAG = \{N, E, EP, I, L, G, C\} \quad (2.2)$$

where N is a set of regions ($N \neq \emptyset$), E is the set of relations (edges) between $n \in N$ (e.g., adjacency), EP is set of properties associated with E (e.g., distance and direction), I is the set of properties associated with $n \in N$ (e.g., area, shape, perimeter etc.,), L is the set of labels assigned to $n \in N$, G is the set of group bindings (i.e., the label $l \in L$ is assigned to the group land cover or land use), C is the set stating the confidence with which a label $l \in L$ belongs to a region $n \in N$.

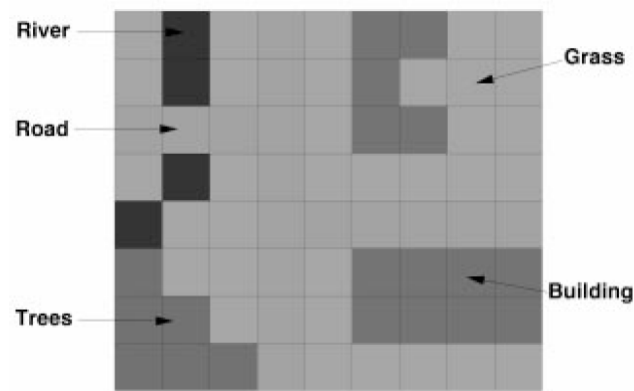


Figure 2.2: Land cover map of Graphtown [4]

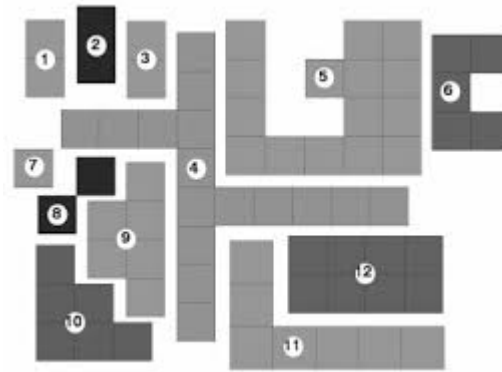


Figure 2.3: Exploded representation of Graphtown [3]

In an XRAG model, each region ($n \in N$) is represented by a node and the relationship between a pair of regions is represented by edges. The attributes of the nodes (I) represent non-relational properties of the regions whereas the attributes of the edges represent relational properties between two regions. In [4], the preceding concepts are explained using an example of raster format land cover map of an imaginary urban area called "Graphtown" as shown in Fig. 2.2. It is assumed that the land cover map of graph town is produced from a high spatial resolution remotely-sensed image.

Twelve discrete land cover regions are recognized by SAMS in the image of Graphtown and it stores location and boundaries of these regions in Region Search Map (RSM) (Fig. 2.3). The XRAG data model is populated with the spatial and structural relationships between regions determined from the RSM. The graph visualization of Graphtown for the adjacency spatial relation is shown in Fig. 2.4. From such a graph representation, distinct clusters of nodes and edges can be that exhibit a particular pattern can be identified. The XRAG data model allows computation of simple quantitative measures that can be used to distinguish urban land use classes.

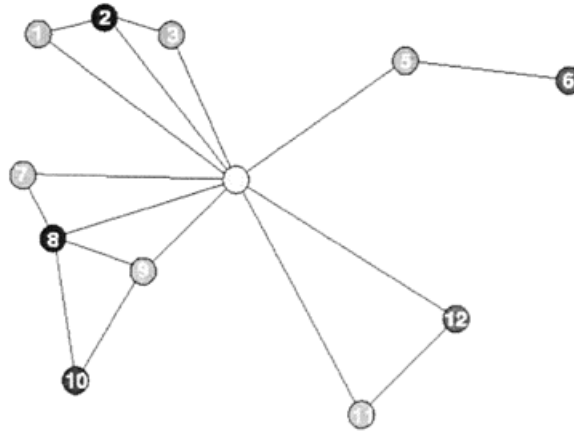


Figure 2.4: Graph Visualisation of Graphtown for spatial relation 'Adjacency'[4]

In general, the spatial structure of the remotely-sensed images is much more complex than the example of Graphtown. Also, the output of the land cover classification is subject to the effects of mixed pixels, shadowing, misclassification and occlusion. The above factors affect the morphological properties and spatial relations between land cover parcels. This in turn affects the land use classification results.

2.1.3 Object Based Techniques

A disadvantage of remote sensing for land use classification is that, there is no direct relationship between the land-use and spectral response. Pixel-based classification of high resolution multi-spectral imagery is unsuitable for land-use classification because of spatial heterogeneity in the spectral response of different cover types. An analysis of contextual object based land-use classification is given in [9]. The descriptive features used for classification are derived from high spatial resolution imagery and airborne LiDAR data. The main aim here is to emulate human cognition by numerical quantification of discriminant properties of images.

In an object-based approach, image analysis is performed on objects; an object is a group of pixels created by a segmentation algorithm. In [9], cartographic limits from the cadastral geospatial database were used to create objects. Two-stage classification approach was used: land-cover types were detected in the first stage and in the second stage, spatial context of land cover was used to determine land-use. Internal and external level analysis of the objects provides the context. At an internal level, analysis of land cover types inside the object is performed. At an external level, the neighbouring objects are analysed.

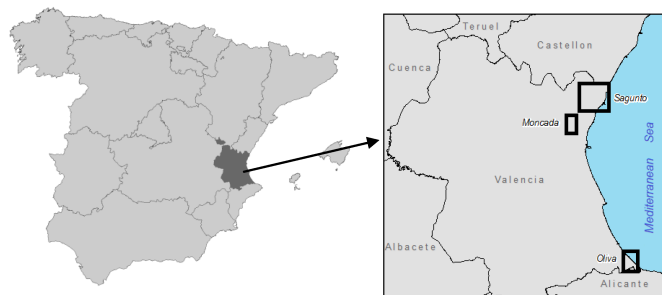


Figure 2.5: Location of the city of Sagunto [9]

The study was carried out in the city of Sagunto in Spain as shown in the figure 2.5. The land-use classes to be discerned were: historical, urban, open-urban, detached housing, terraced housing and industrial. An nDSM, which is the difference between Digital Surface Model (DSM) and Digital Terrain Model (DTM) was derived from LiDAR data.

Descriptive features at three object aggregation levels were defined: object-based, internal context, and external context. Object-based features are divided into two feature groups: image-based features (group I), and geometrical and three-dimensional features (group II). Description of an object with respect to land-cover types within the object was given in internal context features (group III). The characteristics of each object with respect to adjacent objects was given in external context features (group IV). The description of features extracted in different groups is shown in table 2.1.

The classification algorithm used was decision trees. The decision tree predicts the land-use class based on several conditions on extracted features. To analyse the effects of extracted features as listed in table 2.1, four classification tests were applied. In the first test, classification was performed based on the image based features (group I). In the second test, classification was performed based on image, geometrical and 3D features (group 1 + group II). In the third test, classification was performed on image, geometrical, 3D and internal context features (group I + group II + group III). Finally, the fourth test was performed on the features extracted in all the groups. It was observed that the overall classification accuracy consistently increased from Test-1 to Test-4. This indicates the complementary nature of feature groups.

Object based techniques facilitate the usage of information from different data sources and enables multi-scale analysis. Image objects are described in a greater depth than in the pixel-wise approach. The only disadvantage of this approach is that the object-based technique is highly dependent on the algorithm used to create the objects and its selected parameters.

Group description	Extracted features
Group I: image-based features	Spectral: Mean, Standard deviation, Minimum, Maximum Texture: Skewness, Kurtosis, Entropy, Contrast etc.,
Group II: geometrical and three-dimensional features	Compactness, Area, Perimeter, Hight mean, Height maximum etc.,
Group III: internal context features	Building Covered Ratio (BCR), Vegetation Covered Ratio (VCR) etc.,
Group IV: external context features	Number of adjacencies, Mean distance etc.,

Table 2.1: List of descriptive features extracted at different object aggregation levels

2.2 Land-use classification based on Conditional Random Fields

An approach for simultaneous classification of land cover and land use using higher order Conditional Random Field (CRF) is given in [2]. The method uses aerial images, digital surface models (DSM) and polygon objects from land use database as input. CRF are discriminative classifiers, represented by undirected graphical models consisting of nodes n and edges e . The nodes represent image pixels or segments and the edges represent the statistical dependencies between the class labels and data at the associated nodes. CRF directly model the posterior probability $P(y/x)$ of the label vector $y = [y_1, y_2, \dots, y_i, \dots, y_n]$ ($i \in S$ is the index of the image site and S is the set of all image sites) given observations x . The goal of CRF is to assign the most probable label to all the image sites simultaneously (label vector y) from a set of classes $L = [l_1, \dots, l_m]$ given observations x .

$$P(y/x) = \frac{1}{Z(x)} \left(\prod_{i \in S} \phi(y_i, x) \cdot \prod_{h \in H} \psi_h(y_h, x)^{w^h} \right) \quad (2.3)$$

In equation 2.3, $\phi(y_i, x)$ is association potential which models the relation between class label y_i at the site i and the observations x and $\psi_h(y_h, x)$ is clique potential which models the relation between the labels y_j of all nodes n_j that belong to a clique $h \in H$ where h is the index of the clique and H is the set of all cliques. $Z(x)$ is the partition function that acts as a normalization constant and w^h represents the influence of clique potential in the classification process.

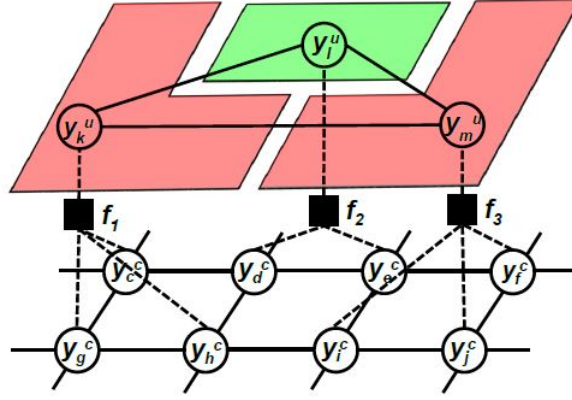


Figure 2.6: CRF consisting of a land cover layer (c) and land use layer (u) [2]

The CRF in [2] consists of a land cover layer and a land use layer, that performs simultaneous classification of land cover and land use with both the classification tasks mutually supporting each other. The nodes in the land cover layer represent super-pixels, the nodes in the land use layer correspond to objects in geospatial database. The intra-layer edges represent spatial dependencies between neighbouring sites in the image, whereas spatially overlapping sites are connected by inter-layer edges as shown in figure 2.6. This leads to higher order cliques (f_1, f_2, f_3) modelling the semantic relation between the land cover and land use sites in the clique. In figure 2.6, nodes are represented by circles, intra-layer edges as solid lines and inter-layer edges as dashed lines. The land cover and land use class labels to be determined are represented by y_i^c and y_i^u respectively. CRF is applied according to equation 2.4 for land cover and land use classification:

$$\begin{aligned}
 P(y/x) = \frac{1}{Z(x)} & \left(\prod_{i \in S^c} \phi^c(y_i^c, x)^{w^1} \cdot \prod_{i \in S^c} \prod_{j \in N_i^c} \psi^c(y_i^c, y_j^c, x)^{w^2} \cdot \prod_{k \in S^u} \phi^u(y_k^u, x)^{w^3} \right. \\
 & \left. \cdot \prod_{k \in S^u} \prod_{l \in N_k^u} \psi^u(y_k^u, y_l^u, x)^{w^4} \cdot \prod_{h \in H} \psi^{cu}(y_h^c, y_h^u)^{w^5} \right) \quad (2.4)
 \end{aligned}$$

The label vector y represents the label configuration of nodes in both land cover and land use layer, $y = \{y^c, y^u\}$. $\phi^c(y_i^c, x)$ and $\phi^u(y_k^u, x)$ are association potentials which model the relationship between class labels y_i^c and y_k^u and the observations x , where $i \in S^c$ and $k \in S^u$ are indexes belonging to land cover super-pixels S^c and land use objects S^u . $\psi^c(y_i^c, y_j^c, x)$ and $\psi^u(y_k^u, y_l^u, x)$ represent the intra-layer interaction potentials which model the spatial dependencies between neighbouring sites in each layer. N_i^c represents the neighbourhood of the land cover super-pixel and N_k^u represents the neighbourhood of the land use object in consideration. $\psi^{cu}(y_h^c, y_h^u)$ represents the inter-layer higher order potential that models the relations between the labels of all nodes belonging to the clique $h \in H$. The parameters

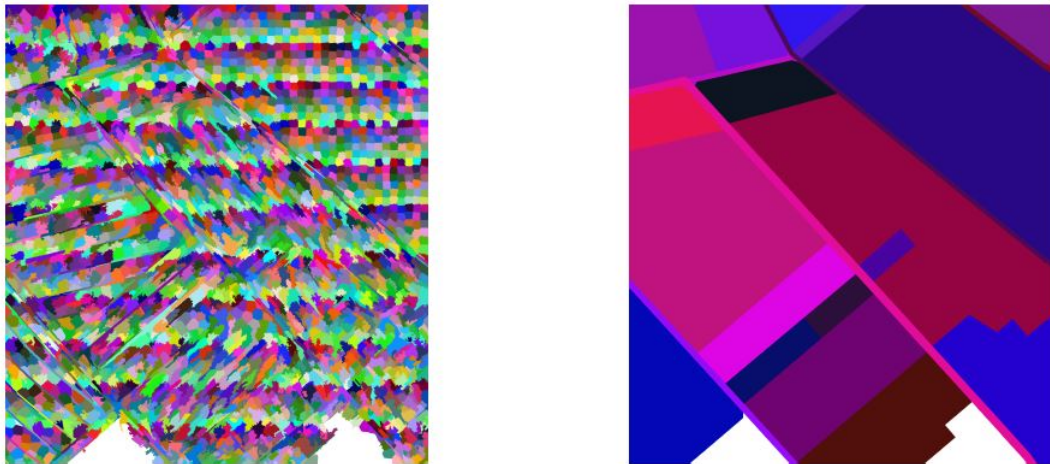


Figure 2.7: Image representing super-pixels for land cover classification (left) and land use objects for land use classification (right) [2]

$(w^1, w^2, w^3, w^4, w^5)$ give the impact of each potential relative to the first potential term w^1 ($w^1 \equiv 1$).

Given the observations x , site-wise feature vectors are generated corresponding to the nodes in the land cover and land use layers. The site-wise feature vectors contain image-based and geometric features. Association potential at both the layers is computed by employing the Random Forest classifier [6]. The pair-wise intra-layer potential is also computed using the Random Forest classifier. A feature vector is generated for each edge in the land cover layer and land use layer. For the land cover layer, the feature vector for each edge is computed by performing element-wise difference of the feature vectors of the nodes joined by the edge. For land use layer, feature vector for each edge is computed by concatenating the feature vectors of both the nodes joined by the edge.

The inter-layer higher order potential is computed using a joint iterative inference procedure. In each iteration, the most probable label configuration is determined at each layer separately, thus obtaining partial solution in the form of class labels for nodes in land cover and land use layers. For both the layers, unary terms are obtained by applying the following two assumptions: the higher order potential in each layer depends on the class labels of the other layer, the second is that the class labels of other layer remain constant during each iteration (the partial solution obtained previously). Not only labelling obtained in the partial solution, but also the beliefs for all the labels is considered. The inference procedure is repeated until no change is observed in the classification result.

In the iterative inference procedure, the two classification tasks mutually influence each other. Contextual relations between land cover and land use are integrated in the classification process by using contextual features that describe complex dependencies of all the nodes in a clique. A discriminative Random Forest classifier then approximates the higher order potentials during the inference procedure. Although introducing context in the classification process, this method is computation intensive because of the hand-crafted feature extraction and iterative approach.

2.3 Land-use classification based on Convolutional Neural Networks

With the emergence of classifiers that work on both spatial and spectral dimensions, e.g., neural network classifier, it is possible to perform land use classification in one step. With the availability of large quantity of data, it has become possible to use techniques based on Convolutional Neural Networks to perform land-use classification. Another variant of classifiers called Support Vector Machines (SVMs) are frequently used for solving image classification problems. SVMs are independent of the dimensionality of feature space, therefore provide better classification results with limited training samples. Neural networks and SVMs show comparable results for land use classification [7]. However, neural network based classification is more robust to training site heterogeneity; and such heterogeneity is common in remote sensing images [16]. The classification problem is mainly tackled using supervised learning methods. The land use information is maintained by national mapping agencies in geo-spatial databases. Commonly, land use data is stored in the form of polygon objects; the label of the object indicates land use.

As mentioned in Chapter 1, the first challenge in the classification of land use polygons using CNN is the variation in geometric extent of polygons. To the best of our knowledge, LiteNet [18] is the first architecture to perform classification of land use polygons using CNN. The network was trained separately using RGB data and a label image encoding land cover. The input patches for CNN were generated by decomposing the polygons. In the input patch, the area inside the polygon is represented by RGB data or land cover encoding and the area outside the polygon is set to 0. However, this under-utilization of data leads to a loss of context information. [19] represent a polygon using a combination of its shape in the form of a binary mask and the image data (e.g. RGB), finally decomposing it to form patches of a fixed size. We adopt this methodology for patch generation from polygons. LuNet [19], which is based on

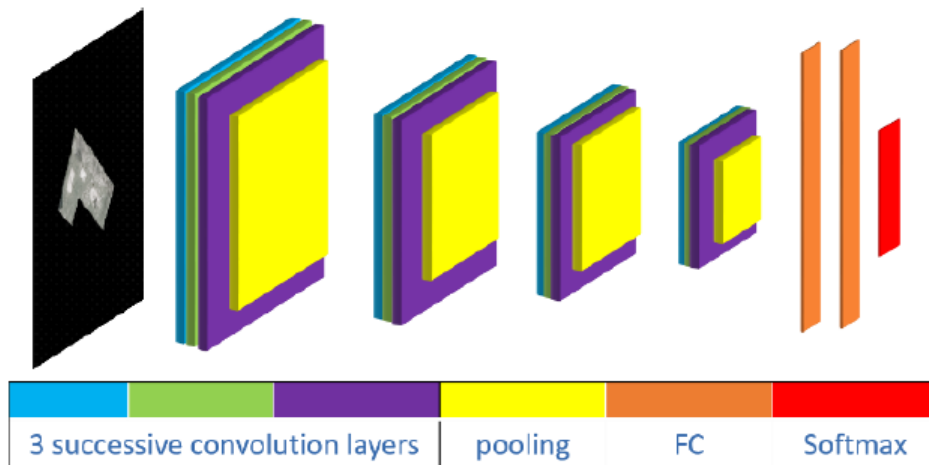


Figure 2.8: Architecture of LiteNet-O [18]

LiteNet, consists of four convolutional blocks and two branches towards the end called two-branch-convolution. The upper branch of the two-branch-convolution extracts global features that are representative of the complete image. The lower branch uses a region of interest (ROI) to focus on the most relevant regions in the image, which helps in the classification of polygons. We also adopt this two-branch convolution in our architectures in this thesis, as it was demonstrated to enhance the classification of land use polygons. Another work on urban land use classification using object based CNN is presented in [20]. The objects generated using mean shift clustering algorithm are classified into two types: linearly and non-linearly shaped objects. Two CNNs with different model structures and window sizes predict the labels for linearly and non-linearly shaped objects and a rule based decision fusion is performed to combine the results. However, such two-scale feature representation might be insufficient to characterize complex geometric polygons. A joint deep learning framework for land cover and land use classification that involves Multi Layer Perceptron (MLP) and CNN classification models was proposed in [21]. The intrinsically hierarchical relationships between land cover and land use were modelled via an iterative Markov process. However, their method focuses solely on urban and suburban areas, leading to an insufficient model transferability. This section briefly describes two network architectures LiteNet and LuNet proposed by Yang et al., in [18] and [19].

2.3.1 LiteNet

In [18], CNN takes an input image of 256×256 pixels, consisting of high resolution aerial images, DSM and DTM and image encoding land cover classes and returns a land-use label. As the sizes of polygons is variable and input size of CNN is fixed, patch

preparation is required. When the polygon is bigger than 256×256 , the polygon was split into tiles of 256×256 with 25% overlap between the tiles. All the tiles whose proportion of area inside the polygon object less than 99.995% are rejected. The thresholding still results in a large number of patches to be classified. Of the remaining tiles, 30% of tiles were then selected at random for classification. A patch is then initialized with a black background and the RGB values from the polygon object are copied to the patch. When the polygon was small enough to fit into the window size of CNN, the patch was initialized with a black background and the polygon was centered to coincide with center of the window.

LiteNet-B architecture [18] consists of four convolution layers, each followed by a ReLU operation and a max-pooling layer with a window of size 2×2 and stride 2. The first three convolutional layers have 32, 64, and 96 filters of size 5×5 , respectively. The fourth convolutional layer has 128 filters with a size of 3×3 . There are two fully connected layers after the last pooling layer, having 128 neurons each. Each of the fully connected layer is followed by a dropout layer with dropout ratio of 0.5. The last fully connected layer gives as output, a vector of class scores. Soft-max classifier is applied to obtain a probabilistic class score.

LiteNet-O [18], as shown in Fig. 2.8 is an extension of LiteNet-B architecture. All convolutional layers were replaced by three successive 3×3 convolutional layers, each followed by a ReLU. This introduces an implicit regularization due to more non-linearity operations. The last convolution in LiteNet-B of size 3×3 is also replaced by three successive 3×3 convolutional layers, so the receptive field of the combined convolution is larger than the one of the last LiteNet-B layer. This architecture performs poorly on the classification of small polygons and using black background for patch generation leads to loss of context information. Also, uncertainties of land cover classification is neglected by using an image encoding land cover classes.

2.3.2 LuNet

LuNet [19] takes a fixed size input of 256×256 , but the land use objects vary considerably in size. Therefore, two patch preparation techniques were introduced in [19]: cropping and re-scaling.

Cropping preserves the original resolution of the image. If the polygon object fits into the window size 256×256 of the CNN, the patch preparation is straight forward.

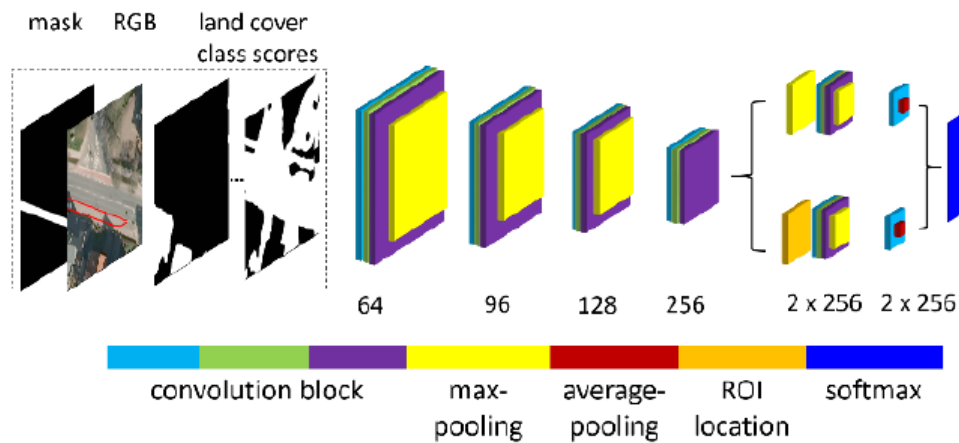


Figure 2.9: Architecture of LuNet [19]

The polygon object is placed in the window such that its center coincides with the window's center. For the polygons that do not fit into the window size of 256×256 , they are split into tiles of 256×256 with 50% overlap between the tiles. The tiles whose proportion of its area inside the polygon object is less than 0.005% are not considered for creating patches. Of the remaining tiles, 40% are selected at random for patch preparation. The object shape is represented in the form of a mask, where the area inside the object is represented by 255 and area outside the object is represented by 0. The patch to be classified consists of $4 + N_c$ where the first three bands are RGB data of polygon objects, the fourth band is binary object mask, the remaining N_c bands are the pixel-wise land-cover classification scores from SkipNet [19].

Using the technique of cropping leads to patches where the overall shape of the objects is not preserved well. On the contrary, the polygon object, the mask and the land cover classification scores can be scaled to fit the window size 256×256 of the CNN. The resulting patches also have $4 + N_c$ bands. The RGB values and the land-cover class scores are scaled using bilinear interpolation and the mask is scaled using nearest neighbourhood interpolation.

LuNet consists of four convolutional blocks at the beginning and two branches towards the end. Each convolutional block consists of three convolution layers of size 3×3 followed by BN and ReLU. Each block in the two branches starts with a 3×3 convolution and a max-pooling layer, followed by a second 3×3 convolution layer with ReLU and a final average pooling layer. After the convolutional block 1, 2 and 3, there is a max-pool layer of size 2×2 with stride 2. Of the two sub-branches after convolutional block 4, the upper branch extracts features representative of the complete image. The lower branch uses ROI (Region of Interest) to focus on the most important regions in the image. ROI is

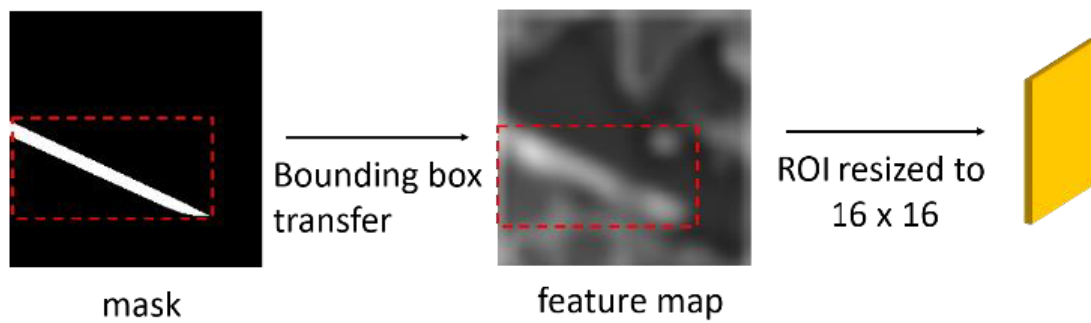


Figure 2.10: ROI location and feature extraction [19]

nothing but a rectangular image grid that tightly encloses the polygon. As the size of the polygons vary, the ROI size also varies. Thus, the output of the fourth convolutional block is resized to the size of 16×16 using bilinear interpolation. The feature vectors of the two subbranches are concatenated and the combined feature vector is given as an input to the fully connected layer that delivers a vector of class scores. Softmax function is then applied to get the probabilistic class scores. This architecture performs better overall and on small polygons when compared to LiteNet (cf. section 2.3.1) because of ROI in the lower branch. The reason for this being the ability to zoom into the object region, which is particularly helpful for small polygons which occupy a small part of the 256×256 input image. However, this architecture takes land-cover classification scores from another network SkipNet [19] as input. The land cover classification scores are derived from the same RGB data that is used in land use classification. Therefore, the input of land cover classification scores is redundant and the errors in the land cover classification are propagated to the land use classification using LuNet.

Chapter 3

Theoretical Foundation

This chapter gives a brief introduction to the concepts of Deep Learning, particularly Convolutional Neural Networks, that will come handy in understanding the methodology of this thesis. The chapter starts with a discussion on CNN, different layers and concepts related to training of the CNN such as loss functions, Optimization algorithms and Regularization.

3.1 Background

The task of recognizing a concept from an image is very challenging for a computer algorithm, although it seems trivial from the perspective of a human. Some of the challenges faced by computer algorithm include deformation, occlusion, illumination, scale variation, background clutter etc. An ideal computer algorithm for image classification would be the one which is less sensitive to intra-class variation and highly sensitive to inter-class variation. One way to go about designing such an algorithm is to provide a large number of examples of each class and let the algorithm learn the visual appearance of objects belonging to each class. Such approach is called data-driven approach, and this approach is the base line of Machine Learning, for that matter, Deep Learning. The two major components of a data-driven approach are: Score function and Loss function. The score function assigns a score vector to each image, this vector contains the confidence value of the image belonging to each class. The loss function checks if the predicted scores and the ground truth value agree with each other. The loss value is high if the classifier is doing a bad job in classifying the training data and the loss value is low if the classification algorithm is working well. Optimization algorithms are used to minimize the loss function (w.r.t., parameters of score function).

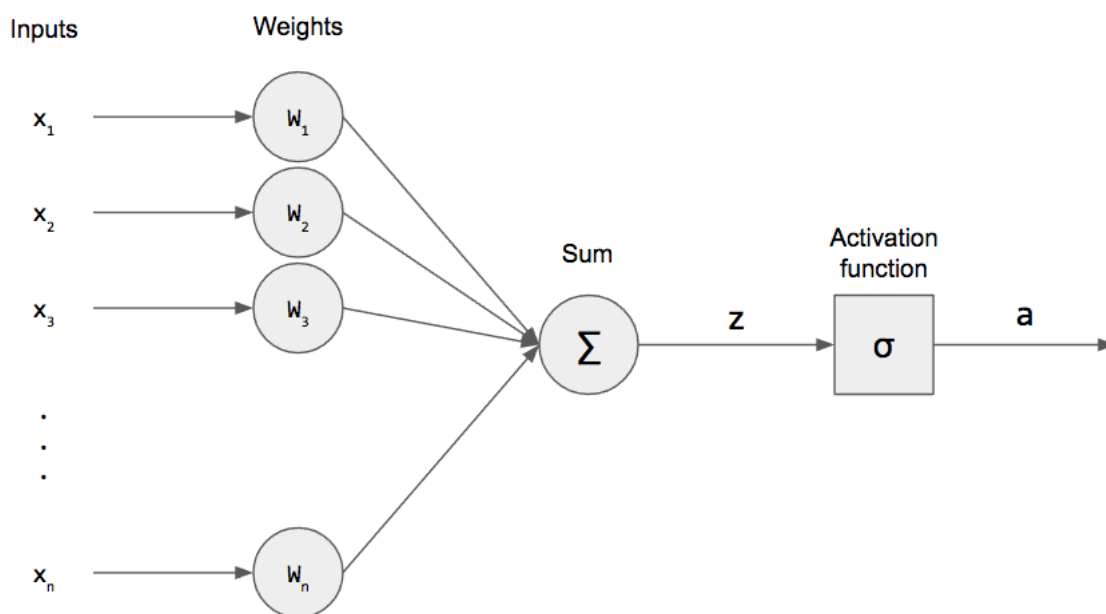


Figure 3.1: The illustration of a perceptron computing weighted sum of inputs and applying an activation function on it

<https://pythonmachinelearning.pro/perceptrons-the-first-neural-networks/>

Originally, the goal of Neural Network was to model the biological neural systems, but today, it has become more oriented towards the field of engineering and research for obtaining good results in Machine Learning tasks. The next sections give a primer on the building blocks of neural network called perceptron, including the process of backpropagation through which it is possible to train the neural network classifier.

3.1.1 Perceptron

A perceptron or neuron is a basic unit of computation in a neural network. It receives input from external sources or other neurons, for example (x_1, x_2, \dots, x_n) in figure 3.1. Each input has an associated weight (w_1, w_2, \dots, w_n) corresponding to its importance when compared to other inputs. The perceptron computes a weighted sum of its input and applies an activation function on it. The activation function applied here is a step function, which produces an output of 1 if the input is greater than or equal to 0, 0 otherwise. The mathematical description of step function is given in equation 3.1. More on its importance and the list of frequently used activation functions is discussed in section 3.2.4.

$$\sigma(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases} \quad (3.1)$$

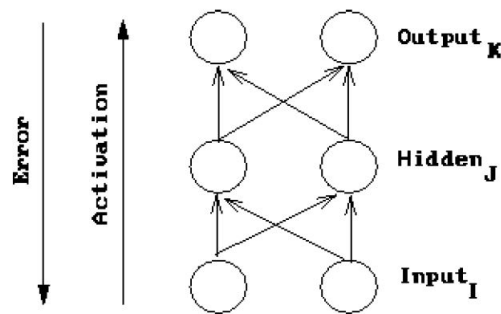


Figure 3.2: Neural network processing

<https://www.cs.swarthmore.edu/~meeden/cs81/s10/BackPropDeriv.pdf>

However, perceptrons are limited to solving two-class problems because of the step function. Also, perceptrons can only classify linearly separable sets of inputs. It is fair to point out that networks consisting of more than one perceptron can be used to solve more difficult problems, as they give rise to complex hyperplanes in the n -dimensional space.

3.1.2 Multi layer Perceptron

A Multi-Layer Perceptron (MLP) is a collection of perceptrons connected in an acyclic graph. MLP consists of distinct layers of perceptrons, and this layer-wise organization of perceptrons is important because it simplifies the evaluation of the network using matrix operations. There are mainly three types of layers: a single input layer, any number of hidden layers and a single output layer. The nodes in the input layer are not perceptrons i.e., they do not perform any computation, they only pass the input to the next layer. The nodes in the hidden and output layer are similar to perceptron except that they use a non-linear activation function instead of a step function. The advantage of the hidden layer lies in the fact that the complexity of the classifier can be further increased by allowing the network to learn higher dimensional features from combinations of the input. Thus, it is capable of distinguishing data that is not linearly separable. Also, MLPs with one hidden layer are capable of approximating any continuous function.

MLPs are usually applied in a supervised learning setting, where the MLP tries to model a correlation between the training data and the ground truth by adjusting the values of weights and biases, by minimizing the error. The adjustment of weights and biases is done by applying back propagation algorithm, which is described in next section.

3.1.3 Forward and Back propagation

Back propagation is a methodology of computing the gradients of expressions by recursive application of chain rule. In simple terms, suppose there is a function $f(x)$, where x is a vector of inputs, the main idea is to find the gradient of f at x . In terms of MLP, the function f corresponds to the loss function and input x corresponds to the training data and network weights. First, the model is fed with input data and the result is obtained. Error is calculated on the obtained result and is back-propagated to update the parameters (weights).

The figure 3.2 contains an input layer i , hidden layer j and output layer k . Let w_{kj} denote weight from hidden layer to output layer, w_{ji} denote weight from input layer to hidden layer. Let a denote output of the activation function, t denote the target value or ground truth and net denote the net input. Let the error on the obtained value after forward propagation be calculated using mean squared error.

$$E = \frac{1}{2} \sum_k (t_k - a_k)^2 \quad (3.2)$$

The main goal of backpropagation is to update the weights of the neural network to reduce the error E .

$$\Delta w_{kj} \propto -\frac{\partial E}{\partial w_{kj}} \quad (3.3)$$

Using chain rule equation 3.3 can be written as:

$$\Delta w_{kj} = -\varepsilon \frac{\partial E}{\partial a_k} \frac{\partial a_k}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}} \quad (3.4)$$

Considering the first term of equation 3.4 in which the error equation 3.2 is differentiated with respect to the output obtained after forward propagation, we get equation 3.5

$$\frac{\partial E}{\partial a_k} = \frac{\partial(\frac{1}{2}(t_k - a_k)^2)}{\partial a_k} = -(t_k - a_k) \quad (3.5)$$

Now computing the second term of equation 3.4 where the activation applied is sigmoid, we get:

$$\frac{\partial a_k}{\partial net_k} = \frac{\partial(1 + e^{-net_k})^{-1}}{\partial net_k} = \frac{e^{-net_k}}{(1 + e^{net_k})^2} = a_k(1 - a_k) \quad (3.6)$$

Finally, differentiating net input with respect to weight, we get:

$$\frac{\partial net_k}{\partial w_{kj}} = \frac{\partial(w_{kj}a_j)}{\partial w_{kj}} = a_j \quad (3.7)$$

Substituting equations 3.5, 3.6 and 3.7 in equation 3.4, we get:

$$\Delta w_{kj} = \varepsilon(t_k - a_k)a_k(1 - a_k)a_j \quad (3.8)$$

By substituting δ_k for the product of derivative of error and activation function in equation 3.8, we get equation 3.9 which gives us a weight updation for hidden to output layer weight.

$$\Delta w_{kj} = \varepsilon\delta_k a_j \quad (3.9)$$

Similarly, weight updation equation for input to hidden layer can be derived as given in equation 3.10, which depends on error at all the nodes which it is connected to.

$$\Delta w_{ji} = \varepsilon\delta_j a_i \quad (3.10)$$

Inspite of being a very powerful algorithm, back propagation suffers from several limitations. Back propagation algorithm can get stuck in a local minimum. This issue can be addressed by using a momentum term α which brings in a factor of previous weight update in the equation. Hence, it gets a small boost from the previous updation which may help in pushing it out of the local minima. However, a large momentum value may push the algorithm beyond global minima. Similarly, the learning rate or step size may push the algorithm beyond global minima. Therefore, right setting of momentum and learning rate parameters is very important for the model to function well which can be achieved by tuning the model using validation set.

3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are similar to the Neural Networks described in the previous section. They consist of neurons and their goal is to learn weights and biases. They are similar in terms of computation inside a single neuron such as dot product and summation, applying non-linearity and computing loss function at the output layer. The difference is that the CNN explicitly assumes that the input to the network are images, hence by encoding certain properties into the architecture, it is possible to reduce the number of parameters in the network. The layers in a CNN have neurons arranged in 3 dimensions which represent height, width and depth of the image. Also, a neuron in a particular layer of CNN is only connected to a small number of neurons in the previous layer unlike neural networks. At the end of the CNN, the whole image is reduced to a vector of dimension equal to the number of classes, which represent class scores.

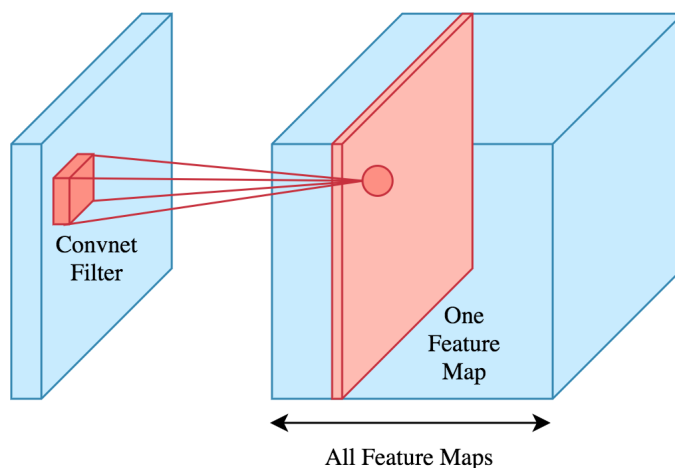


Figure 3.3: Convolution layer illustrating one convolution filter outputs one feature map
<https://datascience.stackexchange.com/questions/67318/convolution-layer-dimensions-in-deeper-layers>

Four types of layers are mainly used to build CNN, namely Convolutional layer, pooling layer, non-linearity and Fully connected layer. The next sections describe these layers, the details of their parameters, hyperparameters and their connectivity.

3.2.1 Convolution Layer

Regular Neural Networks do not scale well to the images as they result in a large number of parameters to be learnt. Convolution Layer is the main building block of CNN and it consists of a set of learnable filters. The filter dimensions are small along width and height dimension (typically 3×3 or 5×5), but extend to the full depth of input volume. This is called filter size or receptive field of neuron and it is a hyperparameter. The input image is convolved with each filter (represents shared parameters) during the forward pass, which results in a two dimensional feature map. The response in the corresponding feature map location is strong when the filter correlates well with a region of the input image. Intuitively, low level features are learned by the filters in initial layers and high level features are learned by filters in the later convolution layers of the network. The two dimensional feature maps obtained as output of a particular convolution layer are stacked together along the depth dimension to form 3 dimensional input volume to the next layer. The other hyperparameters in CNN are number of filters at each layer, stride and padding. Stride corresponds to the number of pixels to jump when we slide the filter to convolve with the input image. The higher the stride, the smaller the output volume. Padding pads the input volume with zeros at the borders, it is typically used to maintain the size of the

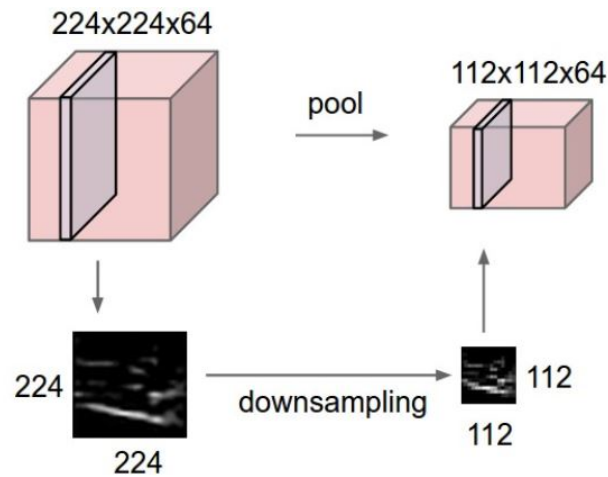


Figure 3.4: The illustration of a spatial pooling operation with a filter size of 2×2 by a stride of 2 in the high direction, and 2 in the width direction

<https://cs231n.github.io/convolutional-networks/>

output volume, so that width and height of input volume equals width and height of output volume.

3.2.2 Pooling Layer

A pooling layer is generally placed between convolution layers. The main idea of pooling layer is to reduce the size of the output volume along width and height dimensions in order to reduce the number of parameters to be learnt thereby decreasing the computation time. It provides invariance to slightly different input images. It also helps in controlling overfitting. The hyperparameters in this layer are filter size and stride. In the pooling layer, there are no parameters to be learnt since it computes a fixed function of the input. The two most commonly used pooling operations are max pooling and average pooling. The max pooling selects the maximum of the value in the coverage area of the pooling filter, whereas average pooling computes an average over all the values in the coverage area of the pooling filter.

3.2.3 Fully connected layer

A fully connected layer typically appears towards the end of the CNN after all the convolution and pooling layers and just before the output layer. Also called as a linear layer, it flattens the output from the previous layers into a 1-dimensional vector. A linear transformation using weights and biases is then applied on this vector along with a non-linearity function. The output layer typically contains nodes equal to the number of classification

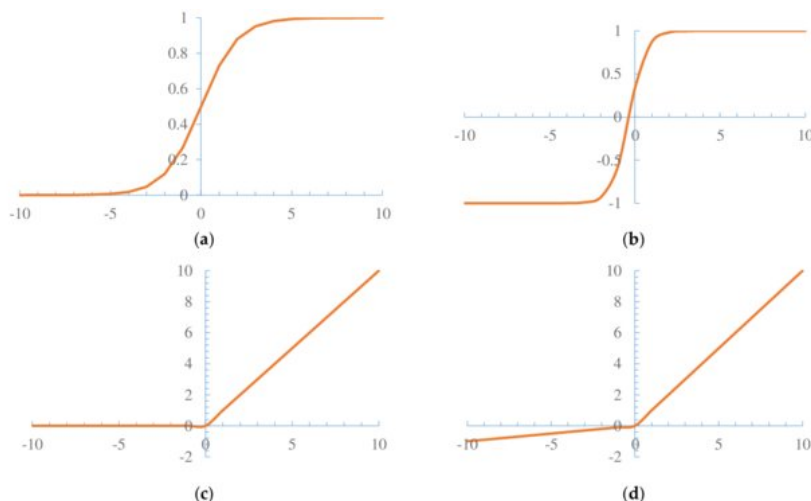


Figure 3.5: Nonlinear functions (a) Sigmoid function (b) Tanh function (c) ReLU function (d) Leaky ReLU function

<https://www.researchgate.net/publication/323617663/>

classes. Typically, softmax activation function is applied to the output of output layer to obtain the probabilities of the input to belong to each class.

$$f(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (3.11)$$

3.2.4 Non-linearity

A Non-linearity or activation function is a mathematical function to introduce non-linearity in the model. It's importance lies in the fact that the CNN typically approximates a non-convex function and it is possible because of non-linear activations. The commonly used activation functions are:

Sigmoid

The mathematical form of sigmoid non-linearity is $\sigma(x) = 1/(1 + e^{-x})$. This function squashes the input into a range of 0 to 1 as shown in figure 3.5 (a). Even though it was frequently used historically, it is not being used in the current times because of its drawbacks. The first drawback of sigmoid function is that it causes the gradient to become zero when it saturates at 0 or 1. Also, the output of the sigmoid function is not zero-centered. This causes the gradients of weights during back propagation to become all positive or negative.

Tanh

The mathematical form of Tanh non-linearity is $Tanh(x) = (e^x - e^{-x}) / (e^x + e^{-x})$. This function squashes the input into a range of -1 to 1 as shown in figure 3.5 (b). An advantage of Tanh non-linearity is that its output is zero centered, therefore it is always preferred over sigmoid non-linearity.

ReLU

The mathematical form of Rectified Linear Unit (ReLU) non-linearity is $f(x) = \max(0, x)$. In simple terms, this function thresholds its input value at 0 , and this non-linearity is most commonly used. Because of its piece-wise linear non-saturating form, it accelerates the convergence of Stochastic Gradient Descent compared to sigmoid and Tanh. The disadvantage with ReLU non-linearity is that the ReLU units can die (never activate again) when they have 0 gradient. With ReLU non-linearity, we may end up with a large number of dead neurons in the beginning of the training, which results in poor learning of the network. It is caused mainly due to bad initialization of parameters.

Leaky ReLU

The mathematical form of Leaky ReLU non-linearity is $f(x) = \max(\alpha x, x)$ where α is a small constant. It overcomes the dying problem of ReLU, as the function always has a small fractional gradient. It is mainly important to overcome problems caused by bad initialization of parameters.

3.2.5 Batch Normalization

Usually, before training a neural network, we preprocess the data to bring it to a normal distribution so as to prevent early saturation of non-linear activations in the network. But the distribution of activations is constantly changing in the intermediate layers during training. This leads to a slow down in the training process because of a phenomena called covariate shift, in which each layer has to adapt itself to a new distribution at every training step. We can normalize the inputs of each layer using Batch Normalization (BN) [11].

Networks can be made more robust to bad initialization using Batch Normalization (BN). BN performs pre-processing at every layer in the network, and it is integrated into the network in a differentiable manner. BN layer typically appears after the Convolution layer. BN accelerates the training of deep networks and also makes the procedure more stable. There are four steps in BN. The first step calculates mean over the mini-batch

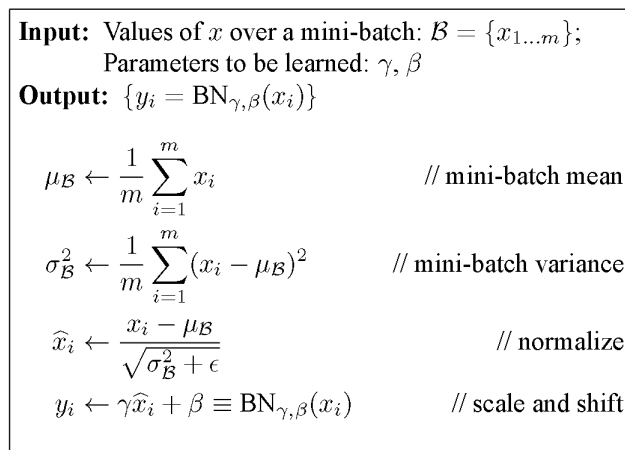


Figure 3.6: Batch Normalization Algorithm

https://kratzert.github.io/images/bn_backpass/bn_algorithm.PNG

and the second step computes variance. The third step normalizes the mini-batch based on calculated mean and variance. The last step allows the batch normalization transform to represent identity, ie., to restore the distribution of data that is learned in the previous layer. It is important to note that the parameters γ and β are learned along with other parameters of the network. BN also acts as a weak regularizer by injecting noise while operating on each mini-batch of data thereby preventing overfitting.

3.2.6 Loss function

A loss function is used to compute the deviation of the predicted output from the ground truth and loss is nothing but an error in prediction. In other words, loss function estimates the error of a set of weights in a neural network. A function with a smooth high-dimensional landscape is preferable as a loss function because it is easy for the optimization algorithm to navigate such a search space via iterative updations to model parameters. The main aim is to obtain a set of parameters that minimize the difference between the model's predicted probability distribution and the distribution of probabilities in the training dataset. This is termed as cross-entropy. The term cross-entropy comes from the field of information theory that computes the difference between an estimated distribution and true distribution. An important thing to note is that the loss function to be used is directly dependent on the activation function that is used in the output layer of neural network.

In the case of a regression problem, where it is required to predict the value of a variable, the output layer has one node and a linear activation is applied in the output

layer. The best suitable loss function for this case is mean squared error (MSE). However, multi-class variant of MSE was also used formerly to train a neural network.

$$MSE = \frac{1}{n} \sum_i (t_i - x_i)^2 \quad (3.12)$$

where x is a vector containing n predictions and t_i is the one-hot encoded ground truth vector correspond to prediction x_i .

In a binary classification problem, the output layer has one node and a sigmoid activation function is used in the output layer. In this case, the binary cross entropy or logarithmic loss function is most suitable. In a multiple-class classification problem, the output layer has nodes equal to the number of classes and the activation function used in the output layer is softmax. In this case, cross-entropy loss is best loss function to use. Generally speaking, the softmax function normalizes the class scores provided by the output layer to probabilistic class scores. Let X is the vector containing class scores obtained from the output layer and let x_i is the i^{th} element of the vector, then the probabilistic class score of i^{th} element is computed as in equation 3.11. Cross entropy loss is computed on this probabilistic class score according to the equation 3.13. Here, t_i and $f(x_i)$ represent the one-hot encoded ground truth and output of the softmax function respectively.

$$CE = - \sum_i t_i \log(f(x_i)) \quad (3.13)$$

3.2.7 Optimization algorithms

The main task of the neural network is to map a set of inputs to a set of outputs by iteratively adjusting the parameters. It is not possible to find the perfect set of parameters, since there are usually tens of millions of parameters to solve for, therefore the main focus here is to find the best set of parameters by using optimization algorithms. In terms of Neural Networks, the process of finding the set of parameters that minimize the loss function is called optimization. The loss functions are non-convex but fully derivable. Gradient is a generalization of slope. While slope is the rate of change in a 1-dimensional function at a particular point, the gradient is a vector of slopes corresponding to each dimension of function. The gradient gives us the direction of steepest rate of increase in the function. Therefore, one needs to make an update in the negative direction of the gradient. Because of large number of parameters, only first order derivatives are applied as second order derivatives are costly in terms of computation and time. However, one still needs to know how far along this direction we should step. This step size or learning rate is one of the most important hyperparameter in Neural Networks.

Gradient Descent is the procedure of iteratively computing gradients and updating the parameters in order to minimize the loss function. It is the most common and standard method to perform optimization of loss functions in Neural Networks. Current day neural networks are trained on millions of training examples. Computing the loss function over complete set of training examples to perform a single parameter update leads to very high time and space complexity. One way to mitigate this issue is to compute the gradient over batches of training examples. This approach is called Mini-Batch gradient descent or stochastic gradient descent (SGD). The reason why this methodology works is because the training examples are correlated.

$$w_{t+1} = w_t - \eta \Delta_{w_t} L(f_{w_t}(x_i), y_i) \quad (3.14)$$

There is no proof of good convergence for SGD. In practice, this algorithm reaches a good local minima even when parameters are randomly initialized. One reason for this might be because of the stochastic property of this algorithm, in which it optimizes a new loss function during every iteration, thus getting out from a bad minima. This topic is still under active research.

3.2.8 Regularization

It is a methodology of preventing overfitting in Neural Networks. Overfitting is said to occur when an algorithm performs well on the training data set, but does not perform well on the test data set. One way to reduce overfitting is to train the neural network with more data, but more training samples may not be available always. The most commonly used regularization strategy is L2 Regularization, also called as weight decay. It penalizes squared magnitude of all weights in the network. The intuitive interpretation of this regularization is that, it chooses the diffused weight vectors over concentrated weight vectors.

$$L = L_0 + \frac{\lambda}{2} \sum_w w^2 \quad (3.15)$$

where L_0 is the original loss function, w is the weight vector and λ is the regularization coefficient. Here $\frac{\lambda}{2} \sum_w w^2$ is the regularization term. The regularization term does not let the weights grow too large when $\lambda > 0$. Taking first derivative of equation 3.15, we get:

$$\frac{\partial L}{\partial w} = \frac{\partial L_0}{\partial w} + \lambda w \quad (3.16)$$

The SGD update then looks like:

$$w_{t+1} = w_t - \eta \frac{\partial L_0}{\partial w_t} - \eta \lambda w \quad (3.17)$$

Since η and λ are positive, the value of w decreases during every iteration, hence the name weight decay.

Data augmentation is another way of preventing overfitting. It artificially increases the size of the training set so that the model does not memorize all the data. It is usually performed by flipping the original images in horizontal or vertical directions or by rotating the images at different angles.

3.3 Network Architectures

As computers became more powerful and processing speed increased, computationally intensive but flexible neural network based classification has become more attractive. The LeNet-5 architecture [14] is one of the first successful applications of CNN and is the origin of most of the recent architectures. The building blocks of LeNet-5 are convolution, pooling and non-linearity layers. Then, Alexnet [13], a deep neural network architecture provided a seismic shift in the field of image classification. This section provides a brief introduction to these two networks. Also, ResNet [8] is introduced, which will deepen the concepts of very deep network with skip connections, which will help in laying the foundation for this thesis.

3.3.1 LeNet

This network was developed by Yann LeCun in the 1990's for identifying digits and zip-codes. This architecture differs from the modern architecture in many ways, nevertheless, it is the origin of much of the recent architectures. The LeNet-5 architecture consists of two sets of convolution and average pooling layers, a flattening convolutional layer, two fully connected layers and a softmax classifier.

The architecture of LeNet-5 can be summarized as:

- To accelerate the training, gray scale input images are normalized using mean and standard deviation.
- Activation functions used are hyperbolic tangent and sigmoid.
- Sparse connection matrices (convolution) between adjacent layers, leading to low computational costs. 5×5 convolutional filters with stride 1 and no padding.

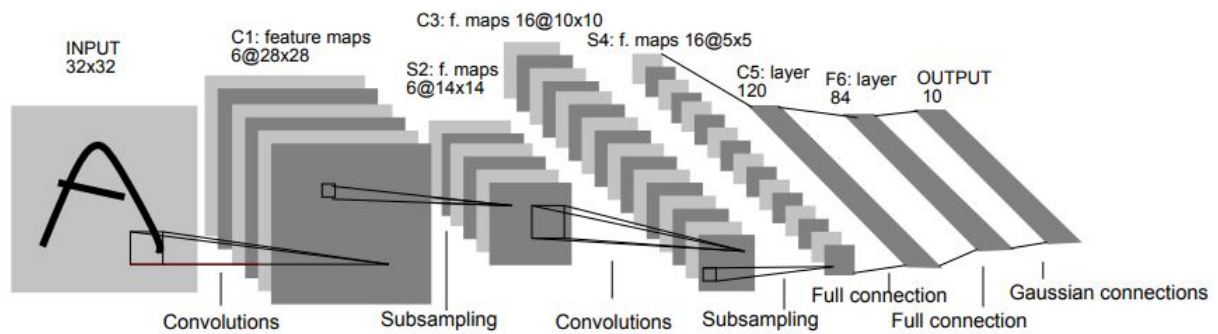


Figure 3.7: Architecture of LeNet-5 [14]

- Average pooling used as pooling function. 2×2 convolution with stride 2.
- Two fully connected layers as final classifier.
- Loss function used is Mean Squared Error.

The main drawback of this architecture is that it lacks built-in mechanisms to avoid overfitting. The saturating activation functions cause the network to stop learning after first few epochs of training. This network architecture is also sensitive to the initial weights.

3.3.2 AlexNet

Following the introduction of LeNet, CNNs were well known in the computer vision community but did not dominate the field owing to the large demands of CNNs in terms of training data and computational resources. AlexNet [13] is a breakthrough paper in deep learning literature. It is a deep convolutional neural network with five convolutional layers (some followed by max-pooling layers), three fully connected layers and a softmax classifier at the end. AlexNet employs ReLU as non-linearity, as opposed to tanh non-linearity employed by the neural network architectures previous to AlexNet. CNN with ReLU units converge several times faster than their counterparts with saturating non-linearities. Also, AlexNet employs overlapping pooling, the authors found that models with overlapping pooling find it more difficult to overfit.

The model learned feature extractors in the lower layers are similar to traditional filters. In the higher layers, high level feature extractors are learned over the ones in the previous layers, which are capable of identifying larger structures. AlexNet for the first time proved that features obtained by learning can transcend the manually designed features. AlexNet controls overfitting by employing dropout regularization. In this technique, the output of each neuron is set to 0 with a probability of 0.5. In this way, during every iteration, the network samples a different architecture. But all the architectures

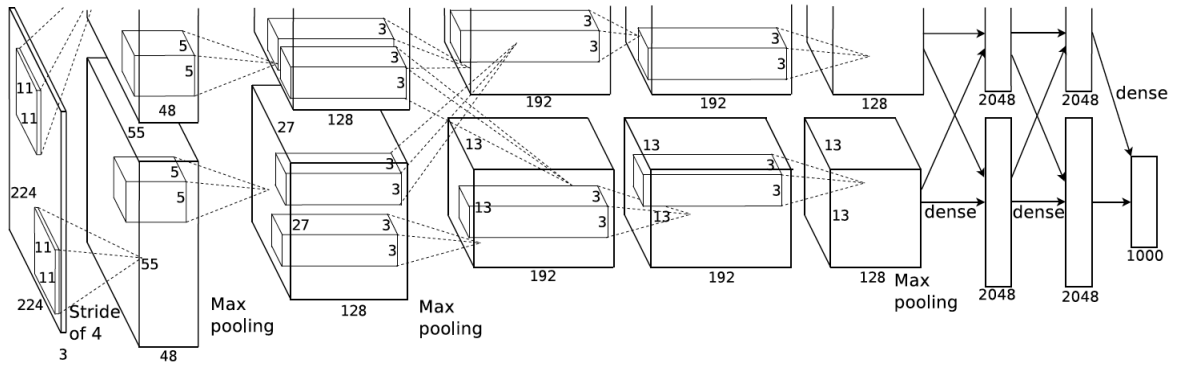


Figure 3.8: Architecture of AlexNet [13]

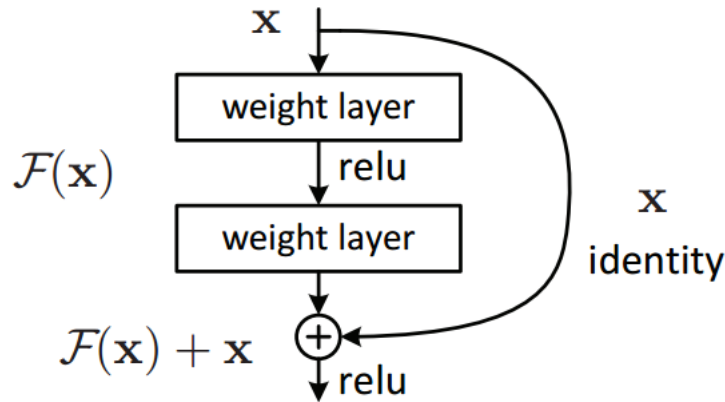


Figure 3.9: Architecture of a residual block [8]

share the weights. In AlexNet, dropout was applied on the two fully connected layers. Also, AlexNet employed image augmentation techniques such as flipping, clipping and color changes. AlexNet showed that Deep Convolutional Neural Networks are capable of near-human performance on large and challenging datasets using supervised learning. However, the operations on overlapping blocks of pixels results in larger memory requirements. The use of large convolution filters (5×5) is not encouraged as it leads to poor weight sharing. Also, due to the higher number of weights, it is computationally expensive. The use of normal distribution to initialize the weights in the neural networks cannot effectively solve the problem of vanishing gradient.

3.3.3 ResNet

ResNet [8] provides a residual learning framework to address the degradation problem (addition of more layers leading to higher training error) faced while training deep neural networks. The vanishing gradient problem was addressed using the normalized

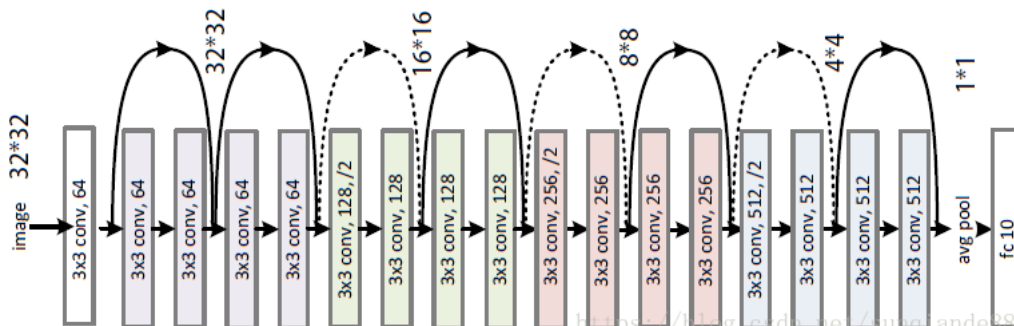


Figure 3.10: The architecture of ResNet-18

<https://www.bitcoininsider.org/article/53615/traffic-sign-classification>

initialization of weights in the intermediate layers [11], but the deeper network accuracy was saturated or degrading. In the residual learning framework, the neural network layers are made to fit a residual mapping. The residual function $\mathcal{F}(x)$ is the difference between the input and output of the residual block. The idea here is to push the residual to zero, which is better than fitting an identity mapping using a stack of non-linear layers. The mapping $\mathcal{F}(x) + x$ is realized using shortcut connection and element-wise addition, performing identity mapping. These connections do not add to the complexity of the model as they do not require additional parameters (a slight increase in computational cost because of the extra element-wise additions).

In order to perform $\mathcal{F}(x) + x$, x and \mathcal{F} must have same dimensions. If this is originally not the case, a linear projection is performed through the shortcut connection to match the dimensions (another easy solution can be to perform zero padding to increase dimension). The residual function \mathcal{F} can take a flexible form (can contain arbitrary number of layers). The figure 3.10 represents the architecture of ResNet-18, which has 18 parameter layers. The network layers are connected in series, along with some skip connections. The final layer is fully connected layer which provides the classification result. Dashed skip connections represent resizing. An obvious drawback of residual networks is that the number of layers have to be increased significantly to achieve a small improvement in accuracy. Nevertheless, because of the skip connections, ResNet acts as an ensemble of multiple shallow networks. Moreover, ResNet solve the problem of vanishing gradient not by preserving the gradient through the depth of the network, but by using an ensemble of exponential number of shallow networks.

Chapter 4

Problem Statement

In this chapter, we begin with a brief discussion of the current state-of-the-art in land use classification of geospatial objects to set the platform for the problem that we set out to solve in this thesis.

In chapter 2, the current state-of-the-art network architecture LuNet, for land use classification of geospatial objects is introduced (cf. Section 2.3.2). In addition to Digital Orthophotos, normalized Digital Surface Model (nDSM), land use objects from geospatial database, this architecture takes land-cover classification scores from another network SkipNet [19] as input. The land cover classification scores are derived from the same RGB data that is used in land use classification. Therefore, the input of land cover classification scores is redundant and the errors in the land cover classification are propagated to the land use classification in this network. Small polygons are difficult to be classified using LuNet. Small polygons occupy a small part of the input image. As this image passes through many layers of the LuNet, it can vanish by the time it reaches the end of the network. The final 1-D feature vector corresponding to the small polygon may not be representative because of the data loss. To overcome the problem of data loss, maximization of dataflow through the network can be performed. To make the final 1-D feature vector representative of the input image, intermediate information from various stages of the network can be utilized. The next section presents the problem statement and builds on these ideas to find solutions for the problems.

4.1 Problem statement

The goal of this thesis is to build on the methods proposed in [19] with the aim of improving the classification of land use objects, mainly focusing on small polygons. The LuNet

architecture [19] is considered as baseline for comparison for the networks proposed in this thesis. In particular, the problems to be solved as part of this thesis and a brief picture of the adopted approach can be summarized as follows:

1. Propose a network architecture that strengthens information flow to improve the land use classification. The key is to create short paths from early layers to later layers, maximizing the data flow through the network. We adopt dense blocks as network component, motivated from the DenseNet architecture [10].
2. Utilize intermediate information from different stages of the network, to make the final 1-D vector representative of the input image and compensate the dataloss. We adopt global average pooling (GAP) [15] for this purpose. This results in many network variants, depending on the stage of the network from which intermediate information is utilized.
3. Conduct an extensive set of experiments to compare the proposed network with the baseline and also a comparison among the network variants. Highlight the benefits and drawbacks of the proposed methods.

Chapter 5

Methodology

This chapter presents a CNN for land use classification which is based on LuNet [19] (cf. Section 2.3.2). As mentioned earlier, the large variation of polygons in terms of geometrical extent is a challenge, because the CNN requires a fixed input size (256×256 pixels in case of network proposed in this chapter) while returning a land use label. In this work, the way in which the image patches are prepared follows the method of [19], which is introduced in section 5.1. The concept of dense connectivity which influences the "dense block" component of the proposed network is introduced in section 5.2. Section 5.3 outlines the "DenseLuNet" network architecture proposed in this thesis for land use classification. Section 5.4 describes the different network variants based on the network proposed in section 5.3 and section 5.5 describes the inference procedure.

5.1 Patch preparation

[19] propose two strategies for patch preparation: cropping and rescaling. The basic approach to prepare the input data is to extract a window of size equal to input size of CNN centred at the centre of gravity of the object from all data (RGB bands and binary object mask) and present it to the CNN. This is unproblematic if the polygon size corresponds well to the window size at the ground sampling distance (GSD) (small polygons); otherwise the window is either dominated by information outside the object or the object does not fit into the window. The method adopted to cope with the latter problem is cropping: the window enclosing the object is split into tiles (patches) (large polygons) of the desired size and all patches having a meaningful overlap with the object are classified independently. Finally, the results for the individual input patches are combined (cf. Section 5.5).

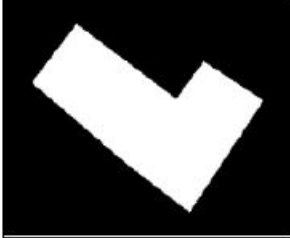

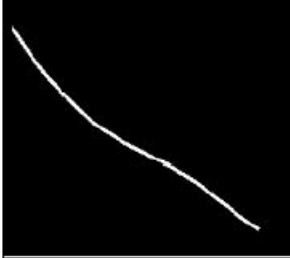

Object shape	RGB image	Size
		320 x 260 pixels
		3400 x 3100 pixels

Figure 5.1: The figure represents two database objects, the left image represents binary object mask and right image represents the corresponding RGB image patch from digital orthophoto

In this work, it is first checked if the polygon fits into a window of size 256×256 , in this case, the window is placed over polygon such that the centers of polygon and window centres coincide. For polygons that do not fit into a window of size 256×256 , the window enclosing the polygon is split into patches of a series of tiles of size 256×256 with an overlap of 50%. Each input patch has four bands: binary image mask (representing shape of the polygon) and RGB bands.

5.2 Dense connectivity

The dense block concept is adopted from [10] as network component for classification in the network proposed in this thesis and this section provides the motivation for adopting dense block. Inside a dense block, the key is to create short paths from early layers to later layers within a dense block, maximizing the data flow within the block. The spatial size of feature maps remains constant in a dense block (Figure 5.2), where each layer within the block obtains input (i.e. feature maps) from all the previous layers of the block. Suppose, each layer in a dense block produces k feature maps, then the l_{th} layer has $n + k(l - 1)$ input feature maps, where n is the number of input feature maps to the dense block. The feature maps from previous layers of the dense block are concatenated to build the feature maps of the l_{th} layer. The number of feature maps generated by each layer within a dense block, k , is called growth rate [10], which is very small ($k = 12$ in this thesis), thus adding only a small number of feature maps at every layer. Therefore, if

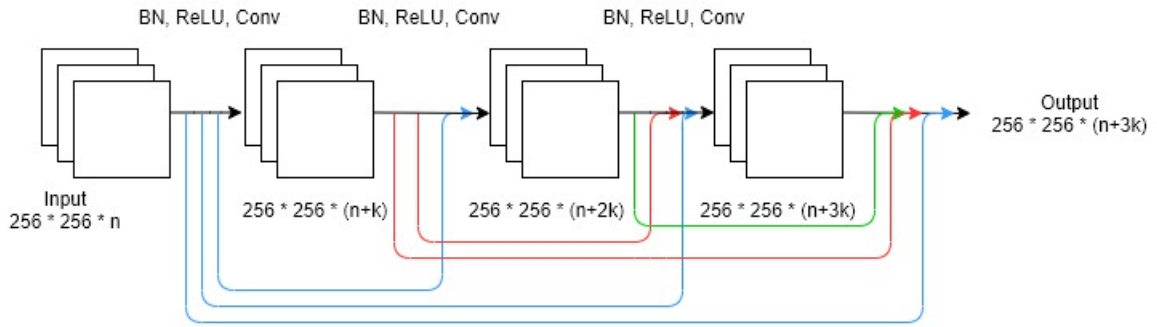


Figure 5.2: A 3-layer dense block with n input channels and k growth rate. Please refer to texts for the abbreviations.

there are L layers in a dense block, there are $(L \times (L + 1))/2$ connections, as opposed to just L connections in a traditional CNN architecture [13].

A dense block can consist of an arbitrary number of layers (4 layers per dense block in this thesis). Each layer in the dense block performs a composite function of three consecutive operations: batch normalization (BN), rectified linear unit (ReLU) processing and 3×3 convolution (Conv). According to [10], the dense connectivity strengthens feature propagation which is the key of its success in visual recognition.

5.3 DenseLuNet

This network is based on LuNet [19] architecture, which is the state-of-the-art in land use classification of geospatial objects (already introduced in section 2.3.2). The DenseLuNet consists of three dense blocks (cf. Section 5.2) with transition layers between them. A transition layer (TL) consist of BN, ReLU, 3×3 convolution and 2×2 max-pooling with stride 2 and the number of output channels is equal to the number of input channels. TL facilitates down-sampling in the network. Every dense block contains four layers, each layer generates 12 feature maps. After the last dense block, two-branch convolution (which is adopted from [19]) is applied for generating a 512 dimensional feature vector for classification. As shown in figure 5.3, the upper branch of the two-branch-convolution extracts global features that are representative of the complete image by performing max-pooling, followed by three convolution layers (3×3), BN and ReLU. The lower branch uses an ROI, to focus on the most relevant regions in the image. In this branch, these regions are focused by aligning a rectangular image grid enclosing the polygon. Due to varying size of ROI, it is rescaled to a size of 16×16 by performing bilinear interpolation. Bilinear interpolation is a resampling method that evaluates an unknown pixel value

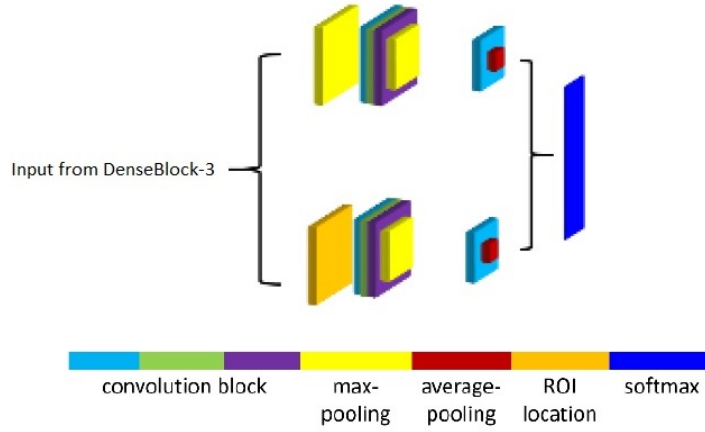


Figure 5.3: Structure of Two-Branch-Convolution

based on distance weighted average of four nearest pixel values. It is followed by three convolution layers (3×3), BN and ReLU. Both branches perform 3×3 convolution and 8×8 average pooling at the end. The output of the two branches are concatenated and given as input to the fully connected layer. The fully connected layer delivers a vector of class scores $(Z_{LU^1}, \dots, Z_{LU^M})^T$, where $C_{LU} = (C_{LU^1}, \dots, C_{LU^M})^T$ is a set of 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 is applied to the class scores:

$$P(C_{LU^c}|X) = \text{softmax}(Z_{LU}|C_{LU^c}) = \frac{\exp(Z_{LU})}{\sum_{i=1}^M \exp(Z_{LU^i})} \quad (5.1)$$

Training is based on mini-batch Stochastic Gradient Descent (SGD) and step learning policy. The function to be optimized is the cross-entropy loss:

$$L = -\frac{1}{N} \cdot \sum_{c,k} \left[y_{LU^c}^k \cdot \log \left(P(C_{LU^c}|X_k) \right) \right] \quad (5.2)$$

where X_k the k_{th} image in the mini-batch, N is the number of images in a mini-batch, $y_{LU^c}^k$ is 1 if the training label of X_k is C_{LU^c} and 0 otherwise.

5.4 Network Variants

The many stages of convolution and pooling operations can cause the final 1-D feature vector to capture no valid information of the input image. The intermediate information from different pooling stages could be helpful for classification. The intermediate information is introduced via GAP [15]. GAP, when applied on the output of a network layer, computes the average value of each feature map and results in a 1-D vector. GAP

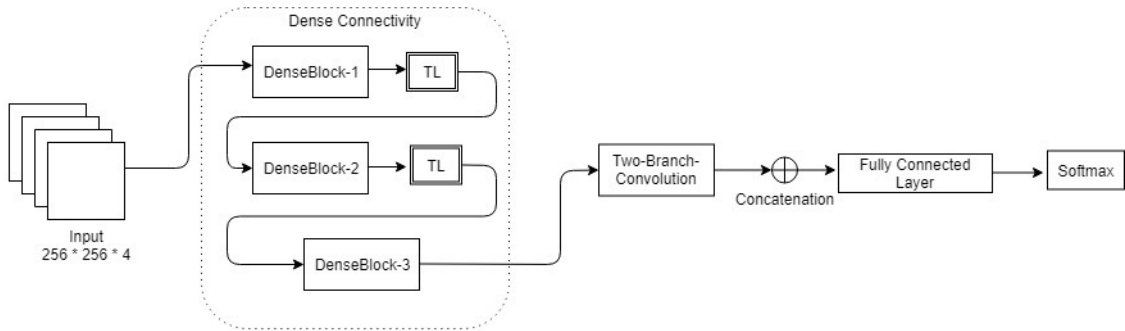


Figure 5.4: The architecture of DenseLuNet. TL: Transition layer, DenseBlock: cf.

Figure 5.2, Two-Branch-Convolution: cf. Figure 5.3

is performed on the output of dense block and is concatenated to the 1-D feature vector obtained from the two-branch convolution [19] (see figure 5.3), which serves as the final feature vector for classification.

In this thesis, an investigation of four networks differing by the stages at which the intermediate information using GAP is extracted on the DenseLuNet base architecture is presented:

1. DenseLuNet architecture as described in Section 5.3.
2. Applying the GAP at the output of the first dense block of DenseLuNet, referred to as DenseLuNet-1.
3. Applying the GAP at the output of the second dense block of DenseLuNet, referred to as DenseLuNet-2 (cf. Figure 5.5).
4. Applying the GAP at the output of the first and second dense block of DenseLuNet, referred to as DenseLuNet-12.

For training these variants, the mini-batch size is set to 10. All networks are trained for five epochs, using a base learning rate of 0.001 and reducing it to 0.0001 after two epochs. Chapter 6 presents a detailed description of the results of these networks applied on different datasets, and provides an analysis of contribution of intermediate information from different stages towards the classification process.

5.5 Inference of polygons

All the evaluations of network variants are performed for polygons. The networks described in sections 5.3 and 5.4 output a probabilistic score for each patch. Therefore, probabilistic class scores are to be derived for each polygon from their corresponding patches.

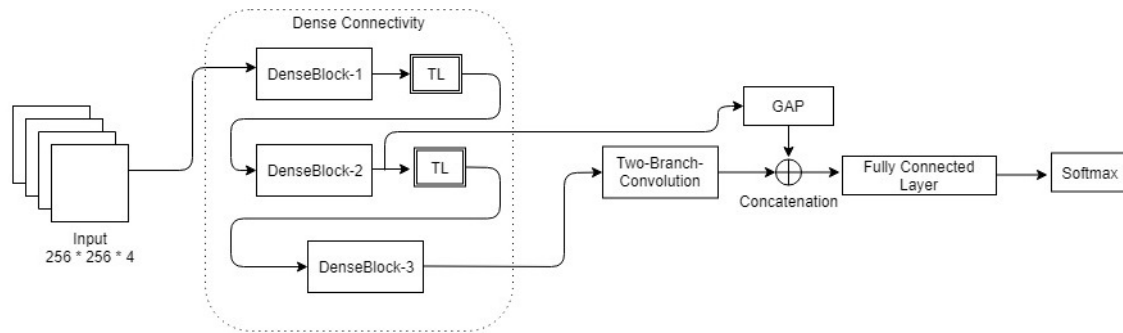


Figure 5.5: The architecture of DenseLuNet-2

If a polygon results in exactly one patch during cropping, its prediction is straightforward, the prediction score of the polygon is the same as the patch score; if a polygon is split into multiple patches, the product of the probabilistic patch scores is determined first, and then the prediction is made based on this product.

Chapter 6

Results and Discussion

In this chapter, the land use classification networks proposed in Chapter 5 are evaluated. First, a comparison of different network variants is presented, then effectiveness of using GAP is analysed. Also, influence of the polygon size on classification result is discussed.

6.1 Datasets and Test Setup

6.1.1 Datasets

The experiments for classification of land use are evaluated on two test sites, located in the cities of Hameln and Schleswig (Germany). Hameln covers an area of $2km \times 6km$. It contains densely built-up residential areas in the centre of the city as well as detached houses, rural areas, industrial areas and rivers. Schleswig covers an area of $6km \times 6km$, showing similar characteristics as Hameln. For both Hameln and Schleswig, digital orthophotos (DOP), and land use objects (corresponding to cadastral parcels) from the German Authoritative Real Estate Cadastre Information System (ALKIS) are available. The DOP are multispectral images (RGB + infrared / IR) with a ground sampling distance (GSD) of $20cm$. The reference for land use is derived from the German geospatial land use database. 10 land use classes are to be discerned for the Hameln and Schleswig test sites: residential (res.), non-residential (non-res.), urban green (green), traffic (traf.), square, cropland (cropl.), grassland (grassl.), forest, water body (water) and others. The class structure of land use is same as in [19].

6.1.2 Test setup

There are 2945 polygons in Hameln and 4345 polygons in Schleswig (cf. Table 6.1). Each test data set is split into two blocks for cross validation. The block size

Table 6.1: Class distribution of Hameln and Schleswig. The table lists number of small polygons, large polygons and total number of polygons.

Class number	Class name	Hameln			Schleswig		
		Small	Large	Total	Small	Large	Total
0	residential (res.)	94	434	528	238	743	981
1	non-residential (non-res.)	149	266	415	69	284	353
2	urban green (green)	179	181	360	186	304	490
3	traffic (traf.)	377	630	1007	211	808	1019
4	square	45	46	91	30	46	76
5	cropland (cropl.)	10	131	141	0	227	227
6	grassland (grassl.)	5	43	48	13	530	543
7	forest	12	87	99	31	345	376
8	water body (water)	9	50	59	56	87	143
9	others	110	87	197	76	61	137

is 10000×15000 pixels ($6km^2$) and 30000×15000 pixels ($18km^2$) for Hameln and Schleswig, respectively. In each test run, one block is used for training and the other one for testing. We evaluate land use classification based on the number of correctly classified polygon objects. We report overall accuracy (OA), i.e., the percentage of land use objects assigned the correct class label by the classification process, and F1 score, i.e., the harmonic mean of precision and recall. All the networks were implemented using tensorflow framework [1]. GPU with configuration (Nvidia GeForce GTX 1080 TI, 11GB) is used to accelerate training and inference.

Data augmentation is performed on the patches generated from cropping, the cropping procedure is explained in Section 5.1. Depending on the size of polygons, the polygons are differentiated into two sets: Large polygons, i.e. polygons that had to be split because they do not fit into the input window of the CNN, are augmented by horizontal and vertical flipping and by applying random rotations in intervals of 30° . In the other case, i.e. small polygons which fit the input size of the CNN, are augmented by horizontal and vertical flipping and by applying random rotations in intervals of 5° . In the end, there are 354178 and 479978 patches for Hameln and Schleswig, respectively.

Table 6.2: Results of land use classification. Network variants (cf. section 5.4). F1: F1 score, OA: Overall Accuracy, both evaluated on the basis of objects. Best scores are printed in bold.

Network Variant	F1 [%]										avg. F1 [%]	OA [%]
	res.	non-res.	green	traf.	square	cropl.	grassl.	forest	water	others		
Hameln												
LuNet	76.5	60.5	57.1	87.8	40.2	55.9	30.9	66.0	34.6	46.0	55.6	69.2
DenseLuNet	81.2	65.8	70.8	89.5	47.9	73.1	32.6	66.9	34.3	43.7	60.6	74.0
DenseLuNet-1	84.4	69.4	74.8	87.9	44.8	72.8	26.3	72.8	38.9	44.0	61.6	74.9
DenseLuNet-2	82.6	67.4	71.0	89.6	41.0	67.1	20.5	68.2	37.5	48.6	59.4	74.4
DenseLuNet-12	84.8	69.6	72.7	89.6	39.6	70.2	25.9	70.7	35.8	47.6	60.7	75.8
Schleswig												
LuNet	79.4	27.1	58.2	87.7	14.9	73.5	76.5	78.1	57.1	28.1	58.1	70.6
DenseLuNet	77.6	51.9	48.3	88.1	12.7	71.2	73.7	79.3	53.3	26.1	58.2	69.8
DenseLuNet-1	80.7	56.3	59.3	86.8	19.2	76.1	78.9	77.9	61.6	25.8	62.2	72.9
DenseLuNet-2	82.9	57.2	60.1	87.8	22.7	75.6	75.5	81.3	58.6	30.7	63.2	73.4
DenseLuNet-12	84.6	58.1	61.8	87.8	20.4	77.0	79.0	76.0	53.5	20.8	61.9	74.5

6.2 Evaluation of Land Use Classification

6.2.1 Evaluation and comparison of network variants

In this section, four variants of networks (cf. Section 5.4) are compared using two datasets Hameln, and Schleswig. The LuNet network [19] serves as a baseline for all other variants. The evaluation results for land use classification evaluated on land use objects are given in Table 6.2. The best values achieved for every accuracy measure on each dataset are printed in bold font. To summarize the performance of the models, the F1 scores with respect to each land use class along with average F1 scores and OA are provided. Analysing Table 6.2, it is evident that DenseLuNet and its variants perform better than LuNet in terms of either OA or average F1 score on both datasets. The best performing variant on Hameln is DenseLuNet-12 which shows an improvement of 6.6% and 5.1% in OA and F1 scores, respectively, in comparison with LuNet. For Schleswig, an improvement of 3.9% and 3.8% in terms of OA and F1 scores, respectively, was reached by DenseLuNet-12, which is the best performing model on this dataset, in comparison with LuNet. On the contrary, DenseLuNet shows about 1% decrease in OA on Schleswig dataset, whereas the F1 score remains the same. The reason for this is unclear and requires further investigation. Overall, it is easy to point out that incorporating dense connectivity leads to better classification results.

In general, all the network variants face difficulties in classifying objects belonging

to the classes *square*, *grassland* and *others* which can be attributed to the fact that only a very small amount of training data is available for these classes, also *others* is a class of heterogeneous appearance. DenseLuNet-1 shows highest improvement of the F1 score for the class *green* by a margin of 17.7% on Hameln. On Schleswig, DenseLuNet-12 shows the highest improvement by a margin of 31% on *non-residential*.

6.2.2 Effectiveness of using global average pooling

In the network variants DenseLuNet-1 and -2, GAP is applied at the output of the 1st and 2nd dense block, respectively, and concatenated to the 1-D feature vector obtained towards the end of the network. In the variant DenseLuNet-12, we apply GAP at the output of both 1st and 2nd dense block. According to the study in section 5.4, the intermediate information computed using GAP is helpful in the classification as it can compensate for information that was lost due to many pooling operations in the network. Analysing Table 6.2, it is easy to notice that the DenseLuNet variants with GAP perform better than DenseLuNet on both, Hameln and Schleswig in terms of either OA or average F1 score. However, when compared to the performance of DenseLuNet, for Hameln the difference is not so pronounced: DenseLuNet-2 shows a slight decrease (1.2%) in average F1 score and OA being almost identical when compared to DenseLuNet. However, on the Schleswig dataset, improvements are seen in both OA and average F1 score by all the three DenseLuNet variants incorporating GAP. Therefore, it is considered that GAP has a positive impact on the classification of land use polygons.

Among the three DenseLuNet variants incorporating GAP, DenseLuNet-12 is the best performing variant on both Hameln and Schleswig in terms of OA, although, the results pertaining to average F1 score do not show a particular trend. This is taken as an indication that the more intermediate information added to the classification process, the better are the classification results.

6.2.3 Influence of the object size

Table 6.3 shows the OA and average F1 scores of small and large polygons along with the combined results which are the same as the ones shown in Table 6.2. The results are given for all the network variants on Hameln and Schleswig. The small set consists of polygons that were represented as a single patch in the classification process. The large set consists of polygons that were split into patches during the input patch generation (cf. Section 5.1). In general, the large set accuracy is greater when compared to the small set because large numbers of patches belonging to the large set are available during classification.

Table 6.3: Results of land use classification represented separately for large, small and all polygons (cf. Table 6.2). The results are provided for all the network variants on Hameln and Schleswig dataset. The number of polygons in each set is given in parenthesis.

Network Variant	Hameln						Schleswig					
	OA[%]			avg. F1 [%]			OA[%]			avg. F1 [%]		
	Large (1955)	Small (990)	All (2945)	Large (1995)	Small (990)	All (2945)	Large (3435)	Small (910)	All (4345)	Large (3435)	Small (910)	All (4345)
LuNet	72.5	62.7	69.2	56.5	38.9	55.6	73.9	58.1	70.6	58.5	39.7	58.1
DenseLuNet	76.7	68.7	74.0	60.5	47.6	60.6	74.1	53.7	69.8	57.5	39.1	58.2
DenseLuNet-1	78.2	68.4	74.9	63.0	45.9	61.6	77.1	57.1	72.9	62.4	43.6	62.2
DenseLuNet-2	77.6	68.1	74.4	59.9	49.5	59.4	77.5	57.7	73.4	63.6	42.2	63.2
DenseLuNet-12	79.1	69.2	75.8	62.3	48.2	60.7	78.4	59.7	74.5	62.0	42.0	61.9

DenseLuNet-12 shows best performance on Hameln and Schleswig in terms of OA of small, large and all polygons, however, the average F1 scores do not show a particular trend. Coming to the classification of small set, DenseLuNet-12 shows 6.5% improvement in the OA in comparison to LuNet on Hameln. This can be attributed to a maximization of data flow due to dense connectivity and utilization of intermediate information from two stages of the network. However, in Schleswig, DenseLuNet-12 shows 1.6% improvement in the OA of small set in comparison to LuNet, while the other DenseLuNet variants show similar performance to that of LuNet in classification of small polygons.

6.2.4 Analysis of confusion matrix

In this section, an analysis of confusion matrices of the best performing network variant DenseLuNet-12 is provided on both datasets Hameln and Schleswig. Also, the confusion matrices of the remaining network variants can be found in Appendix A.2.

R/C	0	1	2	3	4	5	6	7	8	9	Comp.
0	15.69	1.46	0.38	0.03	0.07	---	---	0.03	0.07	0.20	87.50
1	2.86	9.31	0.20	0.58	0.44	0.03	0.03	---	0.07	0.54	66.19
2	0.30	0.47	8.28	0.24	0.37	0.71	0.44	0.65	0.55	0.20	67.75
3	0.14	0.24	0.14	30.77	0.71	0.10	0.20	0.10	0.41	1.38	89.99
4	---	0.47	0.14	0.75	1.22	0.07	0.03	---	0.07	0.34	39.58
5	---	0.03	0.41	---	0.07	3.56	0.38	0.07	0.24	0.03	74.31
6	---	---	0.14	0.24	---	0.54	0.44	0.10	0.13	0.03	26.99
7	---	---	0.31	0.10	---	0.07	---	2.55	0.34	---	75.75
8	---	---	0.34	0.34	---	0.17	---	0.07	0.95	0.14	47.48
9	0.07	0.68	0.24	1.43	0.20	0.10	0.24	0.27	0.48	2.99	44.59
Corr.	82.32	73.47	78.43	89.25	39.53	66.52	24.91	66.28	28.74	51.00	

Figure 6.1: Confusion Matrix of DenseLuNet-12 on Hameln dataset

R/C	0	1	2	3	4	5	6	7	8	9	Comp.
0	17.91	1.59	0.53	0.58	0.22	0.04	0.13	0.20	---	0.37	83.07
1	1.83	4.05	0.45	0.33	0.16	0.04	0.12	0.05	0.05	0.20	55.69
2	0.50	0.30	6.00	0.21	0.28	0.10	1.34	0.88	0.56	0.47	56.40
3	0.16	0.10	0.55	20.14	0.31	---	0.20	0.72	0.33	0.31	88.27
4	0.11	0.29	0.18	0.46	0.29	---	0.05	---	---	0.09	19.58
5	---	0.04	0.06	---	---	3.97	1.16	0.06	0.06	0.06	73.51
6	---	---	0.36	0.08	0.03	0.68	12.43	0.31	0.42	0.02	86.70
7	---	---	0.13	0.21	---	---	1.13	8.18	0.10	0.31	81.30
8	---	---	0.11	0.73	---	0.02	0.34	0.36	1.83	0.03	53.49
9	0.28	0.30	0.42	0.31	0.06	0.07	0.25	0.69	0.07	0.56	18.73
Corr.	86.19	60.66	68.27	87.40	21.36	80.75	72.49	71.40	53.42	23.46	

Figure 6.2: Confusion Matrix of DenseLuNet-12 on Schleswig dataset

A confusion matrix is used to measure the performance of a supervised machine learning algorithm. The confusion matrix shown in figure 6.1 is a 10×10 matrix for 10 class classification. The diagonal elements of this matrix show the percentage of total number of polygons assigned correctly to that class (percentage of true positives i.e., (true positives of class 0/total polygons)*100). Considering row 0, the non-diagonal elements represent the percentage of total number of polygons incorrectly assigned to different classes (percentage of false negatives). The last column and last row represent completeness and correctness respectively. Completeness (recall) is the percentage of the total number of items correctly assigned to a particular class compared to total number of items belonging to that class (ground truth). Correctness (precision) is the percentage of total number of items correctly assigned to a particular class compared to total number of items assigned to that class by the classification algorithm.

Analysing the confusion matrix of DenseLuNet-12 on Hameln and Schleswig in figure 6.1 and 6.2, one can observe that class 0 (*residential*) is mostly confused with class 1 (*non-residential*). It is because of similarity of the two land use classes because of built up areas. Class 2 (*urban green*) is equally confused with all the other classes because of the presence of trees in all the land use classes (residential areas may have trees along the roadside). Class 4 (*Square*) (also known as junction or crossroad) is confused with class 3 (*traffic*) which is well understood. There is a general confusion between class 2 (*urban green*), class 5 (*cropland*) and class 6 (*grassland*). Because of less available training samples belonging to class 6 (*grassland*), it is confused with class 5 (*cropland*), another reason is textural similarities of the two classes.

Chapter 7

Conclusion and Outlook

This thesis proposes a CNN architecture for classification of land use objects in a geospatial database incorporating dense connectivity; we call it DenseLuNet. Three variants of DenseLuNet are investigated differing by the stages at which the intermediate information is extracted using GAP on two test sites, Hameln and Schleswig. DenseLuNet and its variants perform better than LuNet [19] in terms of either overall accuracy or the average F1 score on both datasets. Also, it is observed that intermediate information obtained using GAP has a positive impact on the classification of land use polygons. Compared to LuNet, DenseLunet-12 shows an improvement of 6.6% and 5.1% in OA and F1 scores, respectively, for the Hameln dataset. DenseLuNet-12 shows best performance on Hameln and Schleswig in terms of OA of small, large and all polygons. We conclude that the more the intermediate information via GAP is utilized in the classification process, the better are the classification results.

Future research should focus on including more object knowledge, e.g., in terms of height information. We are also interested to incorporate a hierarchical and more detailed class structure into our approach and to investigate the influence of partly incorrect training data; the latter as a way to be able to use large parts of existing geospatial database content for training. Although some of that information will be outdated and thus wrong, the problem of needing vast amounts of training data could be alleviated in this way. Finally, dense connectivity requires significant amounts of GPU memory and we faced memory issues implementing the network with more than three dense blocks. To overcome these issues, network implementations using shared memories and gradient checkpointing [17] can be performed.

Appendix A

Appendix

The implementation details are discussed in the appendix in terms of softwares and packages used for performing the experiments in this thesis. Also, the confusion matrices are given as an addition to the discussion in Section 6.2.4.

A.1 Programming

The software has been written in Python-3 and run in Python 3.6.3. The following libraries have been used in the software development:

- Tensorflow 1.7.0
- Numpy
- cv2
- tiff file
- skimage
- PIL
- scipy

The network variants proposed in this thesis are written using Tensorflow 1.7.0 [1]. Tensorflow is an open-source software library developed by Google for performing manipulations on matrices called 'tensors'. It is mainly used for developing machine learning and deep learning based applications.

A.2 Confusion Matrices

R/C	0	1	2	3	4	5	6	7	8	9	Comp.
0	15.83	1.56	0.07	0.07	---	0.17	---	0.03	0.10	0.10	88.25
1	4.25	8.10	0.13	0.37	0.38	---	0.03	0.07	0.17	0.57	57.56
2	2.57	0.41	5.57	0.03	0.24	1.09	0.58	0.48	0.75	0.51	45.60
3	0.24	0.85	0.03	28.54	0.48	0.45	0.68	0.27	0.85	1.81	83.47
4	0.10	0.68	0.17	0.64	1.12	0.07	0.03	---	0.10	0.17	36.36
5	0.24	0.44	0.64	---	0.03	2.93	0.21	---	0.14	0.17	61.13
6	---	---	0.03	0.31	---	0.37	0.68	0.07	0.17	---	41.52
7	0.10	---	0.30	---	---	0.17	0.14	2.21	0.38	0.07	65.63
8	---	0.03	0.10	0.14	---	0.37	---	0.03	1.19	0.14	59.51
9	0.14	0.65	0.24	0.71	0.24	0.07	0.41	0.17	1.03	3.05	45.62
Corr.	67.49	63.69	76.36	92.64	45.07	51.46	24.59	66.45	24.41	46.40	

Figure A.1: Confusion Matrix of LuNet on Hameln dataset

R/C	0	1	2	3	4	5	6	7	8	9	Comp.
0	14.98	2.24	0.34	0.03	0.14	---	---	0.03	0.07	0.10	83.52
1	2.90	8.93	0.20	0.54	0.68	0.07	0.03	0.03	0.14	0.54	63.47
2	0.67	0.41	8.05	0.20	0.34	0.61	0.37	0.55	0.55	0.48	65.85
3	0.10	0.27	0.17	30.19	0.54	0.10	0.24	0.17	0.92	1.49	88.30
4	---	0.44	0.14	0.44	1.56	0.07	0.03	---	0.03	0.37	50.65
5	0.13	0.03	0.65	---	0.03	3.42	0.27	0.07	0.14	0.03	71.46
6	0.03	---	0.07	0.20	---	0.17	0.58	0.21	0.31	0.07	35.33
7	0.03	---	0.41	0.10	---	0.03	0.10	2.38	0.31	---	70.62
8	---	0.07	0.24	0.31	---	0.07	---	0.03	1.12	0.17	56.05
9	0.10	0.68	0.24	1.22	0.14	0.03	0.27	0.27	0.96	2.78	41.48
Corr.	79.01	68.34	76.63	90.80	45.50	74.90	30.25	63.55	24.72	46.10	

Figure A.2: Confusion Matrix of DenseLuNet on Hameln dataset

R/C	0	1	2	3	4	5	6	7	8	9	Comp.
0	14.74	2.24	0.34	0.07	0.14	0.03	0.03	---	---	0.34	82.18
1	1.89	9.85	0.37	0.57	0.64	0.03	0.07	---	---	0.64	70.03
2	0.14	0.58	8.83	0.10	0.17	0.98	0.41	0.48	0.14	0.41	72.23
3	0.10	0.38	0.10	29.03	0.58	0.27	1.02	0.24	0.47	2.00	84.90
4	0.03	0.44	0.03	0.54	1.39	0.07	0.07	---	---	0.51	45.14
5	0.03	0.10	0.44	---	---	3.87	0.21	0.03	0.07	0.03	80.75
6	---	---	0.03	0.24	---	0.34	0.64	0.17	0.10	0.10	39.50
7	---	---	0.34	0.03	---	0.03	0.20	2.65	0.07	0.03	78.74
8	---	---	0.37	0.31	---	0.17	0.14	0.07	0.78	0.17	38.91
9	0.07	0.72	0.51	0.95	0.21	0.03	0.48	0.27	0.38	3.08	46.01
Corr.	86.71	68.86	77.58	91.16	44.56	66.30	19.66	67.75	38.87	42.17	

Figure A.3: Confusion Matrix of DenseLuNet-1 on Hameln dataset

R/C	0	1	2	3	4	5	6	7	8	9	Comp.
0	15.22	1.93	0.27	0.10	0.17	0.03	---	0.03	0.03	0.14	84.87
1	2.63	9.14	0.17	0.61	0.85	0.10	0.07	0.03	---	0.47	64.96
2	0.58	0.47	7.92	0.07	0.37	1.05	0.24	0.61	0.24	0.68	64.76
3	0.20	0.27	0.10	30.12	0.78	0.10	0.31	0.10	0.61	1.59	88.08
4	0.03	0.44	0.10	0.57	1.46	0.03	---	---	0.03	0.40	47.34
5	0.03	0.14	0.65	---	0.03	3.56	0.17	---	0.17	0.03	74.32
6	---	---	0.07	0.20	---	0.64	0.31	0.10	0.24	0.07	18.78
7	0.03	---	0.41	0.10	---	0.07	0.03	2.45	0.24	0.03	72.69
8	0.03	---	0.24	0.21	0.03	0.14	0.03	0.17	0.92	0.24	45.80
9	0.14	0.65	0.13	1.09	0.34	0.10	0.21	0.31	0.41	3.33	49.67
Corr.	80.48	70.07	78.67	91.08	36.15	61.13	22.47	64.30	31.75	47.63	

Figure A.4: Confusion Matrix of DenseLuNet-2 on Hameln dataset

R/C	0	1	2	3	4	5	6	7	8	9	Comp.
0	17.90	1.39	0.61	0.71	0.56	---	0.02	0.03	0.04	0.30	83.02
1	3.97	1.45	1.10	0.30	0.22	---	---	---	0.02	0.21	19.99
2	0.74	0.30	6.14	0.58	0.48	0.10	0.71	0.51	0.38	0.72	57.66
3	0.31	0.03	0.50	20.40	0.49	---	---	0.33	0.07	0.69	89.40
4	0.44	0.03	0.31	0.26	0.29	0.03	0.02	---	---	0.09	19.99
5	---	---	0.12	---	---	4.43	0.76	0.04	0.05	---	82.06
6	0.02	0.06	0.92	0.05	0.10	2.07	10.07	0.57	0.27	0.21	70.22
7	---	---	0.26	0.24	0.03	---	0.20	7.95	0.02	1.37	78.97
8	---	0.03	0.12	0.81	0.05	---	0.14	0.36	1.70	0.22	49.73
9	0.15	0.14	0.37	0.37	0.25	0.03	0.07	0.52	---	1.11	36.89
Corr.	76.09	42.30	58.69	85.99	11.87	66.54	84.02	77.24	66.91	22.66	

Figure A.5: Confusion Matrix of LuNet on Schleswig dataset

R/C	0	1	2	3	4	5	6	7	8	9	Comp.
0	16.82	3.46	0.25	0.32	0.21	0.03	---	0.12	---	0.36	77.98
1	1.75	4.82	0.03	0.07	0.12	---	0.06	0.05	0.06	0.32	66.32
2	2.04	1.07	4.12	0.20	0.13	0.02	1.24	0.60	0.42	0.81	38.75
3	0.31	0.85	0.19	19.66	0.23	0.06	---	0.65	0.31	0.56	86.15
4	0.31	0.64	---	0.24	0.15	0.03	0.02	---	---	0.07	10.43
5	---	0.06	0.14	---	---	4.14	1.05	---	---	0.02	76.60
6	---	0.02	1.31	0.02	0.11	1.86	10.57	0.26	0.18	---	73.77
7	0.07	---	0.06	0.30	---	---	0.96	8.50	0.06	0.12	84.43
8	---	0.09	0.12	0.81	---	0.03	0.21	0.53	1.64	---	47.92
9	0.47	0.29	0.21	0.20	---	0.05	0.26	0.67	0.07	0.79	26.26
Corr.	77.20	42.65	64.19	90.14	16.11	66.53	73.61	74.69	60.06	26.03	

Figure A.6: Confusion Matrix of DenseLuNet on Schleswig dataset

R/C	0	1	2	3	4	5	6	7	8	9	Comp.
0	16.24	3.76	0.38	0.41	0.18	0.02	0.02	0.10	0.04	0.43	75.29
1	1.12	5.18	0.23	0.18	0.12	0.02	0.09	0.05	0.09	0.20	71.30
2	0.82	0.72	5.48	0.45	0.14	0.15	1.22	0.83	0.38	0.46	51.46
3	0.09	0.56	0.50	19.49	0.62	0.06	0.06	1.01	0.23	0.20	85.41
4	0.13	0.48	0.06	0.37	0.28	0.02	0.05	0.03	---	0.06	19.18
5	---	---	0.09	---	---	4.33	0.93	0.06	---	---	80.11
6	---	0.02	0.63	0.04	0.11	1.30	11.68	0.37	0.15	0.04	81.50
7	0.02	0.03	0.10	0.18	---	0.02	0.80	8.70	0.11	0.10	86.50
8	---	0.05	0.07	0.68	---	0.05	0.18	0.37	2.00	0.03	58.34
9	0.25	0.34	0.31	0.31	0.02	0.02	0.26	0.77	0.07	0.67	22.19
Corr.	86.99	46.51	69.95	88.15	19.22	72.41	76.42	70.86	65.17	30.72	

Figure A.7: Confusion Matrix of DenseLuNet-1 on Schleswig dataset

R/C	0	1	2	3	4	5	6	7	8	9	Comp.
0	16.88	2.53	0.42	0.78	0.40	0.02	0.02	0.09	0.02	0.40	78.29
1	1.37	4.50	0.38	0.37	0.20	0.04	0.04	0.03	---	0.34	61.85
2	0.55	0.44	6.08	0.37	0.12	0.13	1.24	0.53	0.36	0.82	57.12
3	0.06	0.40	0.39	20.23	0.40	---	0.05	0.29	0.50	0.50	88.65
4	0.18	0.24	0.18	0.40	0.36	---	---	---	---	0.11	24.17
5	---	---	0.15	---	---	4.27	0.82	0.07	0.08	---	79.10
6	---	---	1.39	0.04	0.11	1.29	10.78	0.28	0.31	0.14	75.19
7	---	---	0.10	0.31	---	0.06	0.84	8.35	0.14	0.27	83.03
8	---	0.04	0.09	0.58	---	0.03	0.23	0.30	2.04	0.12	59.61
9	0.11	0.29	0.41	0.18	0.07	0.07	0.20	0.54	0.09	1.04	34.42
Corr.	88.15	53.23	63.37	86.97	21.46	72.39	75.86	79.61	57.56	27.72	

Figure A.8: Confusion Matrix of DenseLuNet-2 on Schleswig dataset

References

- [1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2019). Tensorflow: Large-scale machine learning on heterogeneous systems. 2015. software available from tensorflow. org. URL <https://www.tensorflow.org>.
- [2] Albert, L., Rottensteiner, F., and 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.
- [3] Barnsley, M. (1997). A graph based structural pattern recognition system to infer urban land-use from fine spatial resolution land-cover data. *Computer, Environment and Urban Systems*, 21:209–225.
- [4] Barnsley, M. J. and Barr, S. L. (2000). Monitoring urban land use by earth observation. *Surveys in Geophysics*, 21(2-3):269–289.
- [5] Bauer, T. and Steinnocher, K. (2001). Per-parcel land use classification in urban areas applying a rule-based technique. *GeoBIT/GIS*, 6:24–27.
- [6] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.
- [7] Dixon, B. and Candade, N. (2008). Multispectral landuse classification using neural networks and support vector machines: one or the other, or both? *International Journal of Remote Sensing*, 29(4):1185–1206.
- [8] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- [9] Hermosilla, T., Ruiz, L., Recio, J., and Cambra-López, M. (2012). Assessing contextual descriptive features for plot-based classification of urban areas. *Landscape and Urban Planning*, 106(1):124–137.

- [10] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.
- [11] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.
- [12] Johnsson, K. (1994). Segment-based land-use classification from spot satellite data. *Photogrammetric Engineering and Remote Sensing*, 60(1):47–54.
- [13] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- [14] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- [15] Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.
- [16] Paola, J. D. and Schowengerdt, R. A. (1995). A detailed comparison of backpropagation neural network and maximum-likelihood classifiers for urban land use classification. *IEEE Transactions on Geoscience and remote sensing*, 33(4):981–996.
- [17] Pleiss, G., Chen, D., Huang, G., Li, T., van der Maaten, L., and Weinberger, K. Q. (2017). Memory-efficient implementation of densenets. *arXiv preprint arXiv:1707.06990*.
- [18] Yang, C., Rottensteiner, F., and Heipke, C. (2018). Classification of land cover and land use based on convolutional neural networks. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 4 (2018), Nr. 3*, 4(3):251–258.
- [19] Yang, C., Rottensteiner, F., and Heipke, C. (2019). Towards better classification of land cover and land use based on convolutional neural networks. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.
- [20] Zhang, C., Sargent, I., Pan, X., Li, H., Gardiner, A., Hare, J., and Atkinson, P. M. (2018). An object-based convolutional neural network (ocnn) for urban land use classification. *Remote sensing of environment*, 216:57–70.

-
- [21] Zhang, C., Sargent, I., Pan, X., Li, H., Gardiner, A., Hare, J., and Atkinson, P. M. (2019). Joint deep learning for land cover and land use classification. *Remote sensing of environment*, 221:173–187.
- [22] Zhu, X. X., Tuia, D., Mou, L., Xia, G.-S., Zhang, L., Xu, F., and 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.

IMPROVING THE CLASSIFICATION OF LAND USE OBJECTS USING DENSE CONNECTIVITY OF CONVOLUTIONAL NEURAL NETWORKS

A. Gujrathi^{1,2*}, C. Yang¹, F. Rottensteiner¹, KM. Buddhiraju², C. Heipke¹

¹ Institute of Photogrammetry and GeoInformation, Leibniz Universität Hannover, Germany - (gujrathi, yang, rottensteiner, heipke)@ipi.uni-hannover.de

² Centre of Studies in Resources Engineering, Indian Institute of Technology Bombay, India - bkmohan@csre.iitb.ac.in

Commission II, WG II/6

KEY WORDS: Land use classification, CNN, Geospatial land use database, DenseNet, global average pooling

ABSTRACT:

Land use is an important variable in remote sensing which describes the functions carried out on a piece of land in order to obtain benefits and is especially useful to the personnel working in the fields of urban management and planning. The land use information is maintained by national mapping agencies in geo-spatial databases. Commonly, land use data is stored in the form of polygon objects; the label of the object indicates land use. The main goal of classification of land use objects is to update an existing database in an automatic process. Recently, Convolutional Neural Networks (CNN) have been widely used to tackle this task utilizing high resolution aerial images (and derived data such as digital surface model). One big challenge classifying polygons is to deal with the large variation in their geometrical extent. For this challenge, we adopt the method of Yang et al. (2019) to decompose polygons into regular patches of fixed size. The decomposition leads to two sets of polygons: *small* and *large*, where the former suffers from a lower identification rate. In this paper, we propose CNN methods which incorporate *dense connectivity* and integrate it with intermediate information via *global average pooling* to improve land use classification, mainly focusing on *small* polygons. We present different network variants by incorporating intermediate information via *global average pooling* from different stages of the network. We test our methods on two sites; our experiments show that the dense connectivity and integration of intermediate information has a positive effect not only on the classification accuracy on the whole but also on the identification of small polygons.

1. INTRODUCTION

Land use is an important variable in remote sensing which describes the socio-economic function of a piece of land in order to obtain benefits (Barnsley & Barr 2000). In the region of central Europe, the government surveying authorities maintain geospatial database containing objects whose boundaries are related to property boundaries. The information of land use of property objects becomes outdated quickly as the property owners are not obliged to inform the government of changes in land use. Thus, a system is required to analyse the change in land use of the objects stored in the geospatial database. This can be done by extracting land use information from recently acquired aerial images. The extracted information is checked against the information stored in the database and thus a database update can be performed (Gerke & Heipke, 2008; Albert et al., 2017).

The land use information is maintained by national mapping agencies in geo-spatial databases in the form of polygon objects with class labels indicating the object's land use. This setting is adopted in this paper, where the primitive considered for land use classification is a polygon object of the geospatial database. The main goal of land use classification is to update an existing database in an automatic process. Traditional approaches for land use classification require hand-crafted features derived from image data, and then apply a supervised classifier such as Random Forests to deal with these features. Here, contextual models like Conditional Random Fields (CRF) have also been applied for classification purpose, e.g. (Albert, et al., 2017). However, these methods incorporating hand-crafted features are strenuous and time consuming. The rapid progress in remote sensing technology has resulted in a bulk of images of the earth

surface taken by satellites, airplanes or drones, with different imaging modalities. With the large availability of data, the focus shifts to the automatic extraction of valuable information. Approaches based on CNN are known to provide impressive results when large amount of training data is available; CNNs are currently being used in many remote sensing applications (Zhu et al., 2017).

For land use classification, a major challenge is the large variation of polygons in terms of their geometrical extent; for instance, *road* objects are thin and long, whereas *residential* objects cover both, very large and quite small areas. Recently, CNN-based method for land use classification proposed by Yang et al. (2019) solved the problem by decomposing large polygons into smaller patches of fixed size which suits the input of CNN. To represent a polygon, they use a combination of its shape in the form of a binary mask and the image data (e.g. RGB) and decompose it to form patches of fixed size. We adopt this methodology for the generation of input patches from polygons. During the decomposition, two types of polygons are differentiated: *large* polygons have multiple smaller patches whereas *small* polygons have exactly one patch. In the analysis of their classification results, the authors observed that the *small* polygons are hard to be classified correctly. Possible reasons for lower classification accuracy of *small* polygons are the following: (i) One problem of CNN is that as input passes through many layers of a neural network, the information can vanish by the time it reaches the end of the network. (ii) The final 1-D feature vector before classification may not capture valid information of the *small* polygons due to many pooling operations.

* Corresponding author.

In this paper, we build on the methods proposed in (Yang et al., 2019) with the aim of improving the classification of land use objects, mainly focussing on *small* polygons. In our work, we only use the binary mask and RGB data as input. The scientific contribution of this paper can be summarized as follows:

- We propose a network architecture incorporating dense connectivity (Huang et al., 2017) that strengthens information flow to improve the land use classification. The key is to create short paths from early layers to later layers, maximizing the data flow through the network.
- We apply global average pooling (GAP) (Lin et al., 2013) at different stages of the network, resulting in many network variants, and utilize it as intermediate information in the classification process, to compensate the data loss caused by the many pooling operations in the network.
- We conduct an extensive set of experiments to compare these network variants, and to highlight the benefits and drawbacks of the proposed methods.

In section 2, we give a review of related work. Our approaches for land use classification are presented in sections 3. Section 4 describes the experimental evaluation of our approach. Conclusions and an outlook are given in section 5.

2. RELATED WORK

We start with a brief introduction to land use classification. We then briefly discuss a history of deep learning (especially CNN) in land use classification. After that, we present the current state-of-the-art in land use classification based on polygon objects in geospatial databases.

Land cover is the physical material present on a piece of land (e.g., *water, grass, concrete* etc.). Land use corresponds to the socio-economic function of a piece of land (e.g., *residential, agricultural* etc.). Classification of land cover is simpler because there is a direct relationship between land cover and exitant spectral reflectance, but land use is an abstract concept. The technique suggested in Barnsley & Barr (2000) for land use classification is to divide the classification procedure into two stages: the first being semantic segmentation of the image for land cover classification; the second being land use classification based on the spatial pattern of land cover. The first stage can be performed by a number of techniques ranging from a standard maximum likelihood classifier to artificial neural networks. The disadvantage of such a two stage process is that the accuracy of the land use classification depends on the accuracy of land cover classification, i.e., an error in the first stage is propagated through the second stage. Johnsson (1994) and Bauer & Steinnocher (2001) investigated segment-based land use classification. Segments are obtained by spectral classification. Spatial information of segments such as size, neighbours etc., are used for rule-based classification of image segments into land use categories. An interesting work on land use object classification combining high spatial resolution imagery, LiDAR data and cadastral plots is given in Hermosilla et al. (2012). Land use objects are characterised by image based, geometric and contextual hand crafted features. With the emergence of classifiers that work on both spatial and spectral dimensions, e.g., neural network classifier, it is possible to perform land use classification in one step.

As computers became more powerful and processing speed increased, computationally intensive but flexible neural network based classification has become more attractive. The LeNet-5 architecture (LeCun et al., 1998) is one of the first successful

applications of CNN and is the origin of most of the recent architectures. The building blocks of LeNet-5 are convolution, pooling and non-linearity layers. Then, Alexnet (Krizhevsky et al., 2012), a deep neural network architecture provided a seismic shift in the field of image classification. Another variant of classifiers called Support Vector Machines (SVMs) are frequently used for solving image classification problems. SVMs are independent of the dimensionality of feature space, therefore provide better classification results with limited training samples. Neural networks and SVMs show comparable results for land use classification (Dixon et al., 2008). However, neural network based classification is more robust to training site heterogeneity; and such heterogeneity is common in remote sensing images (Paola & Schowengerdt 1995).

As mentioned in Section 1, the first challenge in the classification of land use polygons using CNN is the variation in geometric extent of polygons. To the best of our knowledge, LiteNet (Yang et al., 2018) is the first architecture to perform classification of land use polygons using CNN. The network was trained separately using RGB data and a label image encoding land cover. The input patches for CNN were generated by decomposing the polygons. In the input patch, the area inside the polygon is represented by RGB data or land cover encoding and the area outside the polygon is set to 0. However, this underutilization of data leads to a loss of context information. Yang et al. (2019) represent a polygon using a combination of its shape in the form of a binary mask and the image data (e.g. RGB), finally decomposing it to form patches of a fixed size. We adopt this methodology for patch generation from polygons. LuNet (Yang et al., 2019), which is based on LiteNet, consists of four convolutional blocks and two branches towards the end called two-branch-convolution. The upper branch of the two-branch-convolution extracts global features that are representative of the complete image. The lower branch uses a region of interest (ROI) to focus on the most relevant regions in the image, which helps in the classification of polygons. We also adopt this two-branch convolution in our architecture, as it was demonstrated to enhance the classification of land use polygons.

Another work on urban land use classification using object based CNN is presented in Zhang et al. (2018). The objects generated using mean shift clustering algorithm are classified into two types: linearly and non-linearly shaped objects. Two CNNs with different model structures and window sizes predict the labels for linearly and non-linearly shaped objects and a rule based decision fusion is performed to combine the results. However, such two-scale feature representation might be insufficient to characterize complex geometric polygons. A joint deep learning framework for land cover and land use classification that involves Multi Layer Perceptron (MLP) and CNN classification models was proposed in Zhang et al. (2019). The intrinsically hierarchical relationships between land cover and land use were modelled via an iterative Markov process. However, their method focuses solely on urban and suburban areas, leading to an insufficient model transferability.

Recent work by He et al. (2016) and Huang et al. (2017) has shown that shorter connections between layers close to input and those close to output in very deep CNNs leads to more accurate and efficient to train networks; ResNet (He et al., 2016) uses identity connections to bypass signal and summation operations when combining input and output layers. These networks are easier to optimize and gain accuracy from considerably increased depth. Many ResNet layers contribute very little and there is a large amount of redundancy in deep residual networks. Stochastic depth (Huang et al., 2016) randomly drops the layers

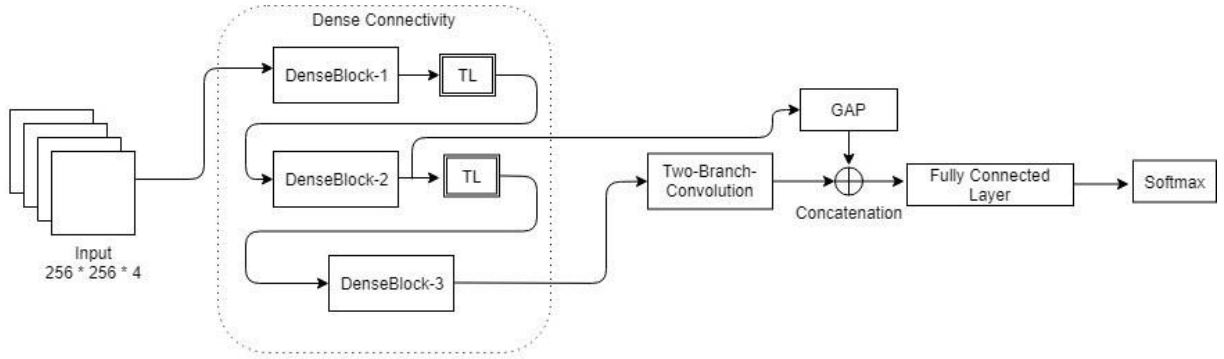


Figure 1. The architecture of DenseLuNet-2. TL: Transition layer, DenseBlock: cf. Fig. 2, Two-Branch-Convolution: cf. Yang et al. (2019)

during training to overcome this problem. Feed-forward neural network can be considered as an algorithm with a state variable, where the state is passed on from layer to layer. Every neural network layer reads the state from its previous layer and writes to the subsequent layer its own state in addition to the previous state. The network architectures that make the state preservation implicit are desirable to overcome redundancy in network layers.

The DenseNet architecture (Huang et al., 2017) differentiates between the information that is added to the network and information that is preserved. DenseNet allows maximum information flow within the network, by connecting all layers within a dense block. The DenseNet architecture encourages improved flow of information and gradients throughout the network, alleviates the vanishing-gradient problem, and helps in strengthening feature propagation. Also, this architecture significantly reduces the number of parameters to be learnt and encourages feature reuse. GAP (Lin et al., 2013) computes the average value of each feature map at a particular layer of the network. An advantage of GAP is that it sums up the spatial information which might be useful in classification of data. GAP also introduces global context (Yu et al., 2018) providing high level semantic information.

Our approach follows the concepts of Huang et al. (2017) and Lin et al. (2013). We use dense block as main classification unit and GAP to obtain intermediate information, which we believe helps in feature propagation and compensates the data loss in our CNN architecture.

3. LAND USE CLASSIFICATION USING CNN

In this section, we propose a CNN for land use classification which is based on LuNet (Yang et al., 2019). As mentioned earlier, the large variation of polygons in terms of geometrical extent is a challenge, because our CNN requires a fixed input size (256 x 256 pixels) while returning a land use label. In this work, the way in which the image patches are prepared follows the method of Yang et al., (2019), which is introduced in section 3.1. The concept of dense connectivity is introduced in section 3.2. Section 3.3 outlines the network architecture used for land use classification. Section 3.4 describes the network variants and section 3.5 describes the procedure.

3.1 Patch preparation

The basic approach to prepare the input data is to extract a window of 256 x 256 pixels centred at the centre of gravity of the object from all data (RGB bands and binary object mask) and present it to the CNN. This is unproblematic if the polygon size corresponds well to the window size at the ground sampling

distance (GSD); otherwise the window is either dominated by information outside the object (for very small objects) or the object does not fit into the window. The method we adopt to cope with the latter problem is *cropping*: we split the window enclosing the object into tiles (patches) of the desired size and classify all patches having a meaningful overlap with the object independently. Finally, the results for the individual input patches are combined (cf. section 3.5).

3.2 Dense connectivity

We adopt the *dense block* concept from Huang et al. (2017) as network component for classification. The key is to create short paths from early layers to later layers, maximizing the data flow through the network. The spatial size of feature maps remains constant in a dense block (Fig. 2), where each layer within the block obtains input (i.e. feature maps) from all the previous layers of the block. Suppose, each layer in a dense block produces k feature maps, then the l^{th} layer has $n + k \times (l - 1)$ input feature maps, where n is the number of input feature maps to the dense block. The feature maps from previous layers of the dense block are concatenated to build the feature maps of the l^{th} layer. The number of feature maps generated by each layer within a dense block, k , is called growth rate (Huang et al. 2017), which is very small ($k = 12$ in our paper), thus adding only a small number of feature maps at every layer. Therefore, if there are L layers in a dense block, there are $(L \times (L + 1)) \div 2$ connections, as opposed to just L connections in a traditional CNN architecture (Krizhevsky et al., 2012).

A dense block can consist of an arbitrary number of layers (we use 4 layers per dense block in our paper). Each layer in the dense block performs a composite function of three consecutive operations: batch normalization (BN), rectified linear unit (ReLU) processing and 3×3 convolution (Conv). According to Huang et al. (2017), the dense connectivity strengthens feature propagation which is the key of its success in visual recognition.

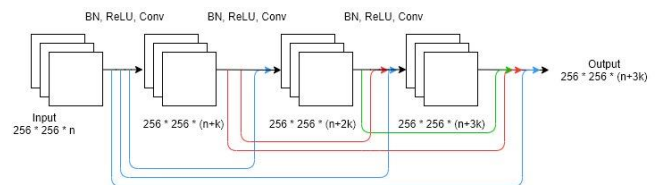


Figure 2. A 3-layer dense block with n input channels and k growth rate. Please refer to texts for the abbreviations.

Network Variant	F1 [%]										avg. F1 [%]	OA [%]
	<i>res.</i>	<i>non-res.</i>	<i>green</i>	<i>traf.</i>	<i>square</i>	<i>cropl.</i>	<i>grassl.</i>	<i>forest</i>	<i>water</i>	<i>others</i>		
Hameln												
LuNet	76.5	60.5	57.1	87.8	40.2	55.9	30.9	66.0	34.6	46.0	55.6	69.2
DenseLuNet	81.2	65.8	70.8	89.5	47.9	73.1	32.6	66.9	34.3	43.7	60.6	74.0
DenseLuNet-1	84.4	69.4	74.8	87.9	44.8	72.8	26.3	72.8	38.9	44.0	61.6	74.9
DenseLuNet-2	82.6	67.4	71.0	89.6	41.0	67.1	20.5	68.2	37.5	48.6	59.4	74.4
DenseLuNet-12	84.8	69.6	72.7	89.6	39.6	70.2	25.9	70.7	35.8	47.6	60.7	75.8
Schleswig												
LuNet	79.4	27.1	58.2	87.7	14.9	73.5	76.5	78.1	57.1	28.1	58.1	70.6
DenseLuNet	77.6	51.9	48.3	88.1	12.7	71.2	73.7	79.3	53.3	26.1	58.2	69.8
DenseLuNet-1	80.7	56.3	59.3	86.8	19.2	76.1	78.9	77.9	61.6	25.8	62.2	72.9
DenseLuNet-2	82.9	57.2	60.1	87.8	22.7	75.6	75.5	81.3	58.6	30.7	63.2	73.4
DenseLuNet-12	84.6	58.1	61.8	87.8	20.4	77.0	79.0	76.0	53.5	20.8	61.9	74.5

Table 1. Results of land use classification. Network variants cf. section (3.4). F1: F1 score, OA: Overall Accuracy, both evaluated on the basis of objects. Best scores are printed in bold.

3.3 DenseLuNet

This network is based on LuNet (Yang et al., 2019) and consists of three dense blocks (cf. Section 3.2) with transition layers between them. A *transition layer* (TL) consist of BN, ReLU, 3×3 convolution and 2×2 max-pooling with stride 2 and the number of output channels is equal to the number of input channels. TL facilitates down-sampling in our network. Every dense block contains four layers, each layer generates 12 feature maps. After the last dense block, two-branch convolution (Yang et al., 2019) is applied for generating a 512 dimensional feature vector for classification. The upper branch of the two-branch-convolution extracts global features that are representative of the complete image by performing max-pooling, followed by three convolution layers, BN and ReLU. The lower branch uses an ROI, to focus on the most relevant regions in the image. In this branch, we focus on these regions by aligning a rectangular image grid enclosing the polygon. The output of the two branches are concatenated and given as input to the fully connected layer. The fully connected layer delivers a vector of class scores $(Z_{LU^1}, \dots, Z_{LU^M})^T$, where $\mathbb{C}_{LU} = \{C_{LU^1}, \dots, C_{LU^M}\}$ is a set of 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 is applied to the class scores:

$$P(C_{LU^c}|X) = \text{softmax}(Z_{LU}, C_{LU^c}) = \frac{\exp(Z_{LU^c})}{\sum_{i=1}^M \exp(Z_{LU^i})}, \quad (1)$$

Training is based on mini-batch Stochastic Gradient Descent (SGD) and step learning policy. The function to be optimized is the cross-entropy loss:

$$L = -\frac{1}{N} \cdot \sum_{c,k} [y_{LU^c}^k \cdot \log(P(C_{LU^c}|X_k))], \quad (2)$$

where X_k is the k^{th} image in the mini-batch, N is the number of images in a mini-batch, $y_{LU^c}^k$ is 1 if the training label of X_k is C_{LU^c} and 0 otherwise.

3.4 Network variants

The many stages of convolution and pooling operations can cause the final 1-D feature vector to capture no valid information of the input image. The intermediate information from different pooling stages could be helpful for classification. We introduce the intermediate information via GAP (Lin et al., 2013). GAP, when

applied on the output of a network layer, computes the average value of each feature map and results in a 1-D vector. GAP is performed on the output of *dense block* and is concatenated to the 1-D feature vector obtained from the two-branch convolution (Yang et al., 2019), which serves as the final feature vector for classification.

In this paper, we investigate four network variants differing by the stages at which the intermediate information using GAP is extracted on the DenseLuNet base architecture: i). DenseLuNet architecture as described in Section 3.3. ii). Applying the GAP at the output of the first dense block of DenseLuNet, referred to as DenseLuNet-1. iii). Applying the GAP at the output of the second dense block of DenseLuNet, referred to as DenseLuNet-2 (cf. Fig. 1). iv). Applying the GAP at the output of the first and second dense block of DenseLuNet, referred to as DenseLuNet-12. For training these variants, the mini-batch size is set to 10. All networks are trained for five epochs, using a base learning rate of 0.001 and reducing it to 0.0001 after two epochs.

3.5 Inference of polygons

All network variants output a probabilistic score for each patch. If a polygon results in exactly one patch during cropping, its prediction is straightforward, the prediction score of the polygon is the same as the patch score; if a polygon is split into multiple patches, the product of the probabilistic patch scores is determined first, and then the prediction is made based on this product.

4. EXPERIMENTS

4.1 Datasets and test setup

4.1.1. Datasets: Our experiments for classification of land use are evaluated on two test sites, located in the cities of Hameln and Schleswig (Germany). Hameln covers an area of $2 \text{ km} \times 6 \text{ km}$. It contains densely built-up residential areas in the centre of the city as well as detached houses, rural areas, industrial areas and rivers. Schleswig covers an area of $6 \text{ km} \times 6 \text{ km}$, showing similar characteristics as Hameln. For both Hameln and Schleswig, digital orthophotos (DOP), and land use objects (corresponding to cadastral parcels) from the German Authoritative Real Estate Cadastre Information System (ALKIS) are available. The DOP are multispectral images (RGB + infrared / IR) with a ground sampling distance (GSD) of 20 cm . The reference for land use is derived from the German geospatial land use database.

We distinguish 10 land use classes for the Hameln and Schleswig test sites: *residential (res.)*, *non-residential (non-res.)*, *urban green (green)*, *traffic (traf.)*, *square*, *cropland (cropl.)*, *grassland (grassl.)*, *forest*, *water body (water)* and *others*. The class structure of land use is same as in (Yang et al., 2019).

4.1.2. Test setup: There are 2945 polygons in Hameln and 4345 polygons in Schleswig. Each test data set is split into two blocks for cross validation. The block size is 10000×15000 pixels (6 km^2) and 30000×15000 pixels (18 km^2) for Hameln and Schleswig, respectively. In each test run, one block is used for training and the other one for testing. We evaluate land use classification based on the number of correctly classified database objects. We report overall accuracy (OA), i.e., the percentage of land use objects assigned the correct class label by the classification process, and F1 score, i.e., the harmonic mean of precision and recall. All the networks were implemented using tensorflow framework (Abadi et al., 2015). We use a GPU (Nvidia GeForce GTX 1080 TI, 11GB) to accelerate training and inference.

We perform data augmentation on the patches generated from cropping. Here, we differentiate two scenarios: *Large polygons*, i.e. polygons that had to be split because they do not fit into the input window of the CNN, are augmented by horizontal and vertical flipping and by applying random rotations in intervals of 30° . In the other case, i.e. *small polygons* which fit the input size of the CNN, are augmented by horizontal and vertical flipping and by applying random rotations in intervals of 5° . In the end, there are 354178 and 479978 patches for Hameln and Schleswig, respectively.

4.2 Evaluation of land use classification

4.2.1. Evaluation and comparison of network variants: In this section, we compare four variants of networks (cf. Section 3.4) using two datasets Hameln and Schleswig. The LuNet network serves as a baseline for all other variants. The evaluation results for land use classification evaluated on land use objects are given in Table 1. The best values achieved for every accuracy measure on each dataset are printed in bold font. To summarize the performance of the models, the F1 scores with respect to each land use class along with average F1 scores and OA are provided. Analysing Table 1, it is evident that DenseLuNet and its variants perform better than LuNet in terms of either OA or average F1 score on both datasets. The best performing variant on Hameln is DenseLuNet-12 which shows an improvement of 6.6% and 5.1% in OA and F1 scores, respectively, in comparison with LuNet. For Schleswig, an improvement of 3.9% and 3.8% in terms of OA and F1 scores, respectively, was reached by DenseLuNet-12,

which is the best performing model on this dataset, in comparison with LuNet. On the contrary, DenseLuNet shows about 1% decrease in OA on Schleswig dataset, whereas the F1 score remains the same. The reason for this is unclear and requires further investigation. Overall, we point out that incorporating dense connectivity leads to better classification results.

In general, all the network variants face difficulties in classifying objects belonging to the classes *square*, *grassland* and *others* which can be attributed to the fact that only a very small amount of training data is available for these classes, also *others* is a class of heterogeneous appearance. DenseLuNet-1 shows highest improvement of the F1 score for the class *green* by a margin of 17.7% on Hameln. On Schleswig, DenseLuNet-12 shows the highest improvement by a margin of 31% on *non-residential*.

4.2.2. Effectiveness of using global average pooling: In our network variants DenseLuNet-1 and -2, we apply GAP at the output of the 1st and 2nd dense block, respectively, and concatenate it to the 1-D feature vector obtained towards the end of the network. In the variant DenseLuNet-12, we apply GAP at the output of both 1st and 2nd dense block. We believe that the intermediate information computed using GAP is helpful in the classification as it can compensate for information that was lost due to many pooling operations in the network. Analysing Table 1, it is easy to notice that the DenseLuNet variants with GAP perform better than DenseLuNet on both, Hameln and Schleswig in terms of either OA or average F1 score. However, when compared to the performance of DenseLuNet, for Hameln the difference is not so pronounced: DenseLuNet-2 shows a slight decrease (1.2%) in average F1 score and OA being almost identical when compared to DenseLuNet. However, on the Schleswig dataset, improvements are seen in both OA and average F1 score by all the three DenseLuNet variants incorporating GAP. Therefore, we consider that GAP has a positive impact on the classification of land use polygons.

Among the three DenseLuNet variants incorporating GAP, DenseLuNet-12 is the best performing variant on both Hameln and Schleswig in terms of OA, although, the results pertaining to average F1 score do not show a particular trend. We take this as an indication that the more intermediate information added to the classification process, the better are the classification results.

4.2.3. Influence of the object size: Table 2 shows the OA and average F1 scores of *small* and *large* polygons along with the combined results which are the same as the ones shown in Table 1. The results are given for all the network variants on Hameln and Schleswig. The *small* set consists of polygons that were represented as a single patch in the classification process. The

Network Variant	Hameln						Schleswig					
	OA[%]			avg. F1 [%]			OA[%]			avg. F1 [%]		
	<i>Large (1955)</i>	<i>Small (990)</i>	<i>All (2945)</i>	<i>Large (1955)</i>	<i>Small (990)</i>	<i>All (2945)</i>	<i>Large (3435)</i>	<i>Small (910)</i>	<i>All (4345)</i>	<i>Large (3435)</i>	<i>Small (910)</i>	<i>All (4345)</i>
LuNet	72.5	62.7	69.2	56.5	38.9	55.6	73.9	58.1	70.6	58.5	39.7	58.1
DenseLuNet	76.7	68.7	74.0	60.5	47.6	60.6	74.1	53.7	69.8	57.5	39.1	58.2
DenseLuNet-1	78.2	68.4	74.9	63.0	45.9	61.6	77.1	57.1	72.9	62.4	43.6	62.2
DenseLuNet-2	77.6	68.1	74.4	59.9	49.5	59.4	77.5	57.7	73.4	63.6	42.2	63.2
DenseLuNet-12	79.1	69.2	75.8	62.3	48.2	60.7	78.4	59.7	74.5	62.0	42.0	61.9

Table 2. Results of land use classification represented separately for large, small and all polygons (cf. Table 1). The results are provided for all the network variants on Hameln and Schleswig dataset. The number of polygons in each set is given in parenthesis.

large set consists of polygons that were split into patches during the input patch generation (cf. Section 4.1.2). In general, the large set accuracy is greater when compared to the small set because large numbers of patches belonging to the large set are available during classification. DenseLuNet-12 shows best performance on Hameln and Schleswig in terms of OA of *small*, *large* and *all* polygons, however, the average F1 scores do not show a particular trend. Coming to the classification of *small* set, DenseLuNet-12 shows 6.5% improvement in the OA in comparison to LuNet on Hameln. This can be attributed to a maximization of data flow due to dense connectivity and utilization of intermediate information from two stages of the network. However, in Schleswig, DenseLuNet-12 shows 1.6% improvement in the OA of *small* set in comparison to LuNet, while the other DenseLuNet variants show similar performance to that of LuNet in classification of *small* polygons.

5. CONCLUSION

In this paper, we proposed a CNN architecture for classification of land use objects in a geospatial database incorporating dense connectivity; we call it DenseLuNet. We investigate four variants of networks differing by the stages at which the intermediate information is extracted using GAP on two test sites. DenseLuNet and its variants perform better than LuNet (Yang et al., 2019) in terms of either overall accuracy or the average F1 score on both datasets. Also, we observe that intermediate information obtained using GAP has a positive impact on the classification of land use polygons. Compared to LuNet, DenseLuNet-12 shows an improvement of 6.6% and 5.1% in OA and F1 scores, respectively, for the Hameln dataset. DenseLuNet-12 shows best performance on Hameln and Schleswig in terms of OA of *small*, *large* and *all* polygons. We conclude that the more the intermediate information via GAP is utilized in the classification process, the better are the classification results.

Future research should focus on including more object knowledge, e.g., in terms of height information. We are also interested to incorporate a hierarchical and more detailed class structure (Yang et al., 2020) into our approach and to investigate the influence of partly incorrect training data; the latter as a way to be able to use large parts of existing geospatial database content for training. Although some of that information will be outdated and thus wrong, the problem of needing vast amounts of training data could be alleviated in this way. Finally, dense connectivity requires significant amounts of GPU memory and we faced memory issues implementing the network with more than three dense blocks. To overcome these issues, network implementations using shared memories and gradient checkpointing (Pleiss et al., 2017) can be performed.

ACKNOWLEDGEMENTS

We thank the Landesamt für Geoinformation und Landesvermessung Niedersachsen (LGLN), the Landesamt für Vermessung und Geoinformation Schleswig Holstein (LVerGeo) and the Landesamt für innere Verwaltung Mecklenburg-Vorpommern (LaiV-MV) for providing the test data and for their support of this project. The first author is a Master's student at Centre of Studies in Resources Engineering, Indian Institute of Technology Bombay and a Combined Study and Practice Stays for Engineers from Developing Countries (KOSPIE) Scholar, funded by Deutscher Akademischer Austauschdienst (DAAD), whose support is gratefully acknowledged.

REFERENCES

- Abadi, M. et al. (2015). Large-scale machine learning on heterogeneous systems. <https://www.tensorflow.org> (accessed 11/04/2020).
- Albert, L., Rottensteiner, F. & Heipke, C. (2017). A higher order conditional random field model for simultaneous classification of land cover and land use. *ISPRS JPhRS* 130: 63-80.
- Bauer, T., & Steinnocher, K. (2001). Per-parcel land use classification in urban areas applying a rule-based technique. *GeoBIT/GIS*, 6, 24-27.
- Barnsley, M. J. & Barr, S. L. (2000). Monitoring urban land use by earth observation. *Surveys in Geophysics* 21(2): 269-289.
- Dixon, B., & Candade, N. (2008). Multispectral landuse classification using neural networks and support vector machines: one or the other, or both? *Int. J. RS*, 29(4), 1185-1206.
- Gerke, M. & Heipke, C. (2008). Image based quality assessment of road databases. *Int. J. of Geoinf. Science*, 22 (8), 871-894.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *CVPR*, 770-778.
- 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.
- Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016). Deep networks with stochastic depth. *ECCV*, 646-661.
- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *CVPR*, 4700-4708.
- Ioffe, S. & Szegedy, C. (2015). Batch Normalization: accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, 448-456.
- Johnsson, K. (1994). Segment-based land-use classification from SPOT satellite data. *PE&RS*, 60(1), 47-54.
- Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *NIPS'12*, 25 Vol. 1, 1097-1105.
- LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278-2324.
- Lin, M., Chen, Q., & Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.
- Paola, J. D., & Schowengerdt, R. A. (1995). A detailed comparison of backpropagation neural network and maximum-likelihood classifiers for urban land use classification. *IEEE Transactions on Geoscience and remote sensing*, 33(4), 981-996.
- Pleiss, G., Chen, D., Huang, G., Li, T., van der Maaten, L., & Weinberger, K. Q. (2017). Memory-efficient implementation of densenets. *arXiv preprint arXiv:1707.06990*.

Yang, C., Rottensteiner, F., & Heipke, C. (2018). Classification of land cover and land use based on convolutional neural networks. *ISPRS Annals IV-3*, 251-258.

Yang, C., Rottensteiner, F., & Heipke, C. (2019). Towards better classification of land cover and land use based on convolutional neural networks. *International Archives XLII-2/W13*, 139-146.

Yang, C., Rottensteiner, F., & Heipke, C. (2020). Exploring semantic relationships for hierarchical land use classification based on convolutional neural networks. *ISPRS Annals*, V-B2.

Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., & Sang, N. (2018). Learning a discriminative feature network for semantic segmentation. *CVPR*, 1857-1866.

Zhang, C., Sargent, I., Pan, X., Li, H., Gardiner, A., Hare, J., & Atkinson, P. M. (2018). An object-based convolutional neural network (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.

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.

Acknowledgements

Foremost, I would like to express my sincere gratitude to Deutscher Akademischer Austauschdienst (DAAD) for granting Combined Study and Practice Stays for Engineers from Developing Countries with Indian IITs (KOSPIE) Scholarship to carry out this thesis at Leibniz Universität Hannover (LUH), Germany. I would like to sincerely thank my supervisor, Prof. Krishna Mohan Buddhiraju, for his constant support, motivation and patience throughout the course of the thesis.

I express my indebtedness to Prof. Christian Heipke (Institut für Photogrammetrie und GeoInformation (IPI), LUH) for offering me the opportunity and resources to carry out this thesis in his research group, resulting in a submission of paper in ISPRS Congress 2020. I sincerely acknowledge his guidance, support and understanding in all the matters during the course of my research stay. I extend my thanks to Prof. Franz Rottensteiner for his many comments and inputs. I am very grateful to M. Sc. Chun Yang for his time, patience, ideas, resources and helping me at all times during the research and writing of this thesis. I thank the members of IPI research group for both simulating discussions and many office parties.

I take this opportunity to thank Centre of Studies in Resources Engineering (CSRE) as a whole. In particular, I am grateful to Prof. Alok Porwal for motivating me to pursue a career in research and Prof. Biplab Banerjee for his patience in answering all my technical queries throughout my master's journey in IIT Bombay. I thank the administrative staff at CSRE for their support to make the research stay possible. I extend my thanks to all my peers at CSRE for giving me the memories of a lifetime. Finally, I thank my family for always being there.

Aishwarya Gujrathi

IIT Bombay

31 May 2020