GEOSPATIAL OBJECT RECOGNITION USING DEEP NETWORKS FOR SATELLITE IMAGES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ONUR BARUT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

MAY 2018

Approval of the thesis:

GEOSPATIAL OBJECT RECOGNITION USING DEEP NETWORKS FOR SATELLITE IMAGES

submitted by **ONUR BARUT** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Tolga Çiloğlu _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. A. Aydın Alatan _____
Supervisor, **Electrical and Electronics Eng. Dept., METU**

Examining Committee Members:

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Eng. Dept., METU _____

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Eng. Dept., METU _____

Assoc. Prof. Dr. Selim Aksoy
Computer Eng. Dept., Bilkent Univ. _____

Assoc. Prof. Dr. Fatih Kamışlı
Electrical and Electronics Eng. Dept., METU _____

Assist. Prof. Dr. Elif Vural
Electrical and Electronics Eng. Dept., METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ONUR BARUT

Signature :

ABSTRACT

GEOSPATIAL OBJECT RECOGNITION USING DEEP NETWORKS FOR SATELLITE IMAGES

Barut, Onur
M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. A. Aydın Alatan

May 2018, 102 pages

Deep learning paradigm has been drawing significant interest during the last decade due to the recent developments in machine learning algorithms and improvements in the computational hardware. Satellite image analysis is also an important scientific area with many objectives, such as disaster and crisis management, forest cover, road mapping, city planning, even military purposes. Spatial correlations of land cover or geospatial objects between different images can be exploited by utilization of convolutional neural networks (CNN) for classification, segmentation, and detection in remotely sensed images. Since the number of high resolution satellite images due to new satellites around the Earth is increased, the visual data for training of such networks is also more available compared to past. In this study, three main research directions for satellite image analysis is examined and tested through simulations. Land cover image scene classification of image patches is obtained by using conventional CNNs. Next, the segmentation of natural scenes into different land cover is obtained by deep networks that are capable of providing segment labels for every pixel on the image. Finally, detection of geospatial objects on satellite images is obtained by object detection techniques based on deep networks. For all these purposes, a multispectral satellite image dataset is manually labeled for several natural scene classes and human-made objects. Different architectures, training techniques and the training parameters are examined through simulations in different datasets.

Keywords: Satellite remote sensing, raster data, deep learning, machine learning, convolutional neural network, geospatial object classification

ÖZ

UYDU GÖRÜNTÜLERİ İÇİN DERİN AĞLAR KULLANILARAK COĞRAFİ NESNELERİN TANIMLANMASI

Barut, Onur
Yüksek Lisans, Elektrik ve Elektronik Mühendislii Bölümü
Tez Yöneticisi : Prof. Dr. A. Aydın Alatan

Mays 2018 , 102 sayfa

Derin öğrenme paradigması, son on yıl içinde yeni makine öğrenimi algoritmalarındaki ve hesaplama donanımındaki son gelişmeler nedeniyle önemli bir ilgi çekmektedir. Uydu görüntü analizi, felaket ve kriz yönetimi, orman örtüsü, yol haritası, şehir planlaması, hatta askeri amaçlar gibi birçok hedefi olan önemli bir bilimsel araştırma alanıdır. Arazi örtüsü ya da farklı görüntüler arasındaki jeo-uzamsal nesnelerin mekânsal korelasyonları, insan yapımı yapılar ile arazi örtüsünün ve jeo-uzamsal nesnelerin tespiti ve sınıflandırılması için literatürde konvolüsyonel sinir ağlarının (CNN) yaygın bir şekilde kullanılmasına olanak sağlamaktadır. Yörüngede birçok yeni uyduların varlığından dolayı yüksek çözünürlüklü uydu görüntülerinin kullanılabilirliğindeki artışa dikkat çekerek, bu tür ağların eğitiminde kullanılacak olan verilere geçmişe kıyasla daha kolay ulaşılabilir. Bu çalışmada uydu görüntü analizi için üç ana araştırma yönü incelenmiş ve simülasyonlarla test edilmiştir. Görüntü parçalarında arazi örtüsü sınıflandırması, ilk olarak geleneksel CNN'ler kullanılarak elde edilir. Daha sonra, doğal manzaraların farklı arazi örtüsüne ayrılması, görüntüdeki her piksel için segment etiketleri sağlayabilen derin ağlar tarafından elde edilir. Son olarak, uydu görüntüleri üzerindeki jeo uzamsal nesnelerin saptanması, derin ağlara dayanan geleneksel nesne tespit teknikleri ile elde edilir. Tüm bu amaçlar için, çok bantlı bir uydu görüntü veri kümesi, çeşitli coğrafi ve insan yapımı nesneler için elle etiketlenir. Farklı mimarilerin, eğitim tekniklerinin ve eğitim parametrelerinin performansa etkisi simü-

lasyonlar aracılığıyla incelenmiştir.

Anahtar Kelimeler: Uydu uzaktan algılama, raster verileri, derin öğrenme, makine öğrenme, evrişimli sinir ağları, coğrafi nesne sınıflandırması

To my Fiancé and my family...

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Prof. Dr. A. Aydın Alatan for his precious suggestion and insistence for this study. He guided me to the right direction during the study and relied on me as I would solve the difficulties. He also provided considerable amount of labeled satellite imagery data to be used in this thesis work as it would make this research more realistic and valuable.

I am also very much appreciated for the permission and support of ASELSAN Elektronik Sanayi ve Ticaret A.Ş. and my supervisors in HAEMM department during this study.

I would like to note here that I am grateful to Emre Can Kaya for his generous support and help with his experience and knowledge in deep learning.

Finally, I would like to thank my dearest Tuğba Ceren for her constant and endless support throughout the study and my family for their endeavour to raise me up this way.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
BoVW	Bag-of-Visual-Words
CNN	Convolutinoal Neural Network
CRF	Conditional Random Field
DL	Deep Learning
DBN	Deep Belief Network
DBM	Deep Boltzmann Machines
DNN	Deep Neural Network
FC	Fully Connected
FCN	Fully Convolutional Network
GEOBIA	Geographic Object Based Image Analysis
GPU	Graphical Processing Unit
HOG	Histogram of Oriented Gradient
ILSVRC	Imagenet Large Scale Visual Recognition Challenge
LR	Learning Rate
LULC	Land Use Land Cover
mAP	mean Average Precision
ML	Machine Learning
NIPS	Neural Information Processing Systems
NIR	Near Infra-Red
OBIA	Object Based Image Analysis
PCA	Principle Component Analysis
RGB	Red Green Blue
RS	Remote Sensing
SAE	Stacked Autoencoder
SIFT	Scale Invariant Feature Transform

SSD	Single Shot Detector
SURF	Speed Up Robust Features
SVM	Support Vector Machines
UAV	Unmanned Air Vehicle
USB-BBR	Unsupervised Score-Based Bouding Box Regression
VOC	Visual Object Challenge
YOLO	You Look Only Once

CHAPTER 1

INTRODUCTION

The rapid development in sensors, especially in camera and imaging technology results in huge amount of data produced every instant due to mobile devices, such as smart phones. Processing this amount of data is troublesome using manpower; however, nowadays, the performance increase in computational hardware, such as Graphical Processing Units (GPUs) assisted the vision researchers to understand, process and derive meaningful information from vast amount of data by deep neural networks. The breakthrough of deep neural networks on image classification problem became very popular with the study of Krizhevsky et al. [2] in Neural Information Processing Systems (NIPS) 2012 and success in International Large Scale Visual Recognition Challenge (ILSVRC) 2012 ImageNet [3] competition. This performance jump led to many other researchers to focus on deep neural networks in their own specific problems and deep learning is currently a quite hot topic in vision research; almost every day a new scientific paper is published to improve the solutions to vision problems by deep learning. On the other hand, the availability of the high quality sensors along with the advancing aerospace and satellite industry enables the researchers to obtain higher amount of remote sensing data with higher spectral and spatial resolution. Thus, increasing quality and the number of remote sensing images let researchers attack this problem and makes deep neural networks and deep learning possible for remote sensing.

One of the tasks that deep learning in remote sensing trying to achieve is to generate a fully automated system that can classify geospatial objects and land cover into separate classes such as airplane, barren land, building, cultivated field, forest, roadway,

runway, ship, storage tank, water etc. It is quite important to be able to classify land use in order to monitor the constant changes on the Earth and manage urban development. Utilization of machine learning techniques for this purpose is quite critical and challenging due to the small number of remote sensing imagery that is available with ground truth labels. Therefore, many computer vision scientists have proposed number of different algorithms to extract information from remote sensing images and significantly contributed to literature of the computer vision and remote sensing field.

The research areas for satellite image analysis can be mainly categorized under three major classes:

1. Scene classification of (rectangular-shaped) regions for land-use
2. Semantic segmentation of (arbitrarily-shaped) regions in terms of land-use
3. Detection of geospatial objects in satellite images (by bounding boxes)

Scene classification techniques simply takes image patches as input and try to classify the *whole* patch into one of the predetermined classes depending the a priori training data. If a rectangular input region contains more than one class of land use, the classifier is assumed to return the most dominant one. On the other hand, semantic segmentation algorithms convert an input (satellite) image patch into *arbitrarily shaped* regions into masks that denote land use labels for every pixel. In this scenario, this resulting output mask has disjoint labels that are based on initial a priori training data. Finally, detection of geospatial objects are achieved by estimating the coordinates of the bounding boxes around the objects by the help of a pretraining network.

In this thesis, the above three research classes are all examined by a satellite image data set with ground-truth information.

1.1 Motivation

Human has five different type of receptors on his sensory system to observe the changes in his environment. These receptors are his tongue to taste, his nose to smell,

his skin to feel, his ear to hear, and his eyes to see. Any type of recognition that is performed in the brain depends on the data collected by those sensory receptors. The decision maker, the brain of the human, collects the data, processes it with its neurons and makes decisions about the environment. Similarly, human-like robots which are called as androids possesses sensors that pretend as the human receptors. In order to illustrate this, a camera attached to a robot acts like the eye of the human. It takes the light from the outside and generate an image accordingly. This image is converted to a digital data and sent to the robots processor. This whole visual process, which is usually referred as computer vision or machine vision, is one of main problems of artificial intelligence.

Recent technological developments in imaging and satellite industry allowed to use very high quality cameras on the satellites to observe the Earth for reconnaissance purposes. This scientific field is denoted as remote sensing. Before application of deep neural networks to vision problems, image processing and vision experts applied several different techniques and algorithms towards Earth observation problems. Their earlier work for the classification of image scenes mainly based on handcrafted features, such as color histograms, scale invariant feature transform (SIFT), GIST, histogram of oriented gradients (HOG) and texture descriptors. The classification success is quite dependent on the selected handcrafted features, and therefore on the expert himself.

In the later studies, the researchers start using machine learning techniques to learn better representations from training data. *Unsupervised* feature learning from unlabeled data became more popular in order to replace the handcrafted features in remote sensing images. By the help of such *learned* representations, a significant progress is achieved in image analysis for remote sensing. Main methods used in this area includes principal component analysis (PCA), sparse coding and auto-encoders (AE).

Along with the advances in cameras to generate high resolution images, the processors and image processing algorithms also have improved. High computational speed of hardware and the increasing number of satellite images make path to use deep learning models in order to get better information from these high resolution images. Unlike the unsupervised learning methods, deep learning based methods are able to

extract more powerful feature representations of raw input data using multiple layers. Since the success of the classification is very much related to the separability of the extracted representations, these deeper neural networks allow researchers to surpass the previous results which were obtained either using handcrafted features or unsupervised learning features.

Deep learning is one of the subcategories of machine learning discipline. Artificial neural networks try to represent a mathematical modeling of human neural system. An artificial network mainly contains three layers: the first layer is called the input layer. Second layer is called hidden layer, the third and the last one is called the output layer. The number of hidden layers and the number of hidden neuron numbers are the design parameters. Deep networks are usually referred to the networks with very huge number of hidden layers. Deep Boltzmann machines (DBMs), deep belief networks (DBNs), stacked auto-encoders (SAEs) and convolutional neural networks (CNNs) are the most popular deep neural network architectures used in remote sensing applications.

Based on the relevant literature, convolutional neural networks (CNNs) are known to be the most powerful feature extractors for vision problems. Although the most popular networks are designed and trained to recognize daily internet images, such as ImageNet, Pascal VOC, MS COCO, it is shown through simulations that these networks are also capable of recognizing geospatial objects and land cover in the satellite images.

1.2 Scope and the organization of the thesis

The main goal of the thesis is to investigate the convolutional neural networks that are popular and widely used in vision and machine learning community and apply those networks to land use scene classification, semantic segmentation of land cover and geospatial object detection problems of remote sensing images.

The image data set for each type of problem that is used in this thesis is generated from a large scale satellite image. The images have a high spatial resolution with 0.46 meter per pixel and they contain 4 color bands, namely Red, Green, Blue and

Near-infrared (RGB-NIR). The ground truth information is provided by a area expert. Since the labeling is handcrafted, it might contain minor positional and registration errors and omission of some objects and regions.

The thesis is organized as follows: A detailed literature overview on land use land cover and geospatial object analysis with both conventional techniques and deep learning methods is presented in Chapter 2, whereas the theory of deep learning and popular deep architectures are provided in Chapter 3. The experiments conducted on scene classification and semantic segmentation of remote sensing images and their results are given in Chapter 4. Chapter 5 contains the results of the experiments on geospatial object detection in satellite images. In the last chapter, Chapter 6, the thesis is concluded based on the simulations results.

CHAPTER 2

LITERATURE OVERVIEW ON LAND USE AND GEOSPATIAL OBJECT ANALYSIS

Recently with the rapid advances in the deep learning and remote sensing data acquisition technologies, numerous studies have been conducted to automatically analyze the satellite imagery. This chapter aims to provide the related studies in the literature during the last two decades. The methods used in the relevant literature of remote sensing image analysis are presented in a chronological order and the evolution of the image analysis methods for the scene classification, semantic segmentation and object detection is reviewed through the following sections.

2.1 Conventional Techniques for Land Use and Geospatial Object Analysis

The necessity of the Earth observation and remote sensing imagery analysis for applications, such as Land Use Land Cover (LULC) classification [4, 5, 6, 7, 8, 9, 10, 11], vegetarian mapping [12, 13], geographic image retrieval [14, 15], environment monitoring, geospatial object detection [16, 17, 18], hazard detection [19, 20] and urban planning goes back to 1970s. Given the fact that the remote sensing images had a coarse spatial resolution in those years, the size of object of interest in such low resolution images were close to the pixel size; therefore most of the methods to analyze earlier time remote sensing imagery were based on per-pixel analysis, or even sub-pixel analysis [21, 22]. With the technological advances in the remote sensing area, the spatial and spectral resolution of these images got better, which in turn resulted in that the objects of interest started to be composed of multiple pixels. Thus, per-pixel

based image analysis methods failed to make a good classification. In this case, researchers focused on the analysis of spatial patterns constructed by each pixel of the object, instead of analyzing each pixel individually.

Based on the conclusion that the per-pixel image analysis generates unsatisfactory results in the finer remote sensing images, researchers came up with the idea called Object Based Image Analysis (OBIA) [22] or Geographic Object Based Image Analysis (GEOBIA) [23] to analyze higher spatial resolution images for the objects consisting of several pixels in a spatial pattern [21]. Since object-level approaches are more capable to extract better representation of the image when compared to per-pixel methods, GEOBIA have become the most popular remote sensing image analysis method for the next decade [21, 22, 23]. Even though these per-pixel and object level approaches showed significant success for some types of remote sensing tasks such as land use, those approaches were incapable of extracting a semantic meaning from the image. For example, even GEOBIA methods are unable to classify whether a patch of a remote sensing image scene contains a runway or a roadway. Similarly, they are not able to distinguish a scene if it is a dense residential or a rural residential. With the recent developments in the machine learning techniques, researchers are now able to make a semantic scene classification in remote sensing images, in other words, label a patch of a remote sensing image scene with a specific semantic class such as airplane, dense residential, rural residential, roadway, runway, etc.

Recently, numerous of studies have been conducted on remote sensing image analysis and remarkable contributions are achieved in the literature. A number of feature extraction methods have been developed and applied to the problems. Before the application of deep learning techniques, handcrafted feature based methods [6, 8, 11, 14, 24, 25, 26, 27, 28] and unsupervised feature learning methods were widely used.

2.1.1 Handcrafted Feature Extraction Methods

Earlier works on image scene classification in remote sensing images mainly depend on the handcrafted feature extraction [6, 8, 11, 14, 24, 25, 26, 27, 28]; hence the expertise of the image analyst is utilized in remote sensing. Widely used feature extraction methods in remote sensing image analysis are color histograms [25], scale

invariant feature transforms (SIFTs) [29], GISTs [30], histogram of oriented gradients (HOGs) [31] and texture descriptors [32, 33].

2.1.1.1 Color Histograms

Color histogram [25] is a simple feature extractor compared to other handcrafted feature extractors. In remote sensing image analysis, color histogram feature extraction method is widely used especially in image scene classification of land use [11, 14, 24, 25]. In [11], Yang et al. compared color histogram with other feature extracting methods for scene classification of 21 classes such as agricultural, airplane, building, freeway, river, runway etc., while Dos Santos et al. used color histogram for coffee crop and pasture parcel recognition in [14]. Since it is only related to the color and independent of the spatial distribution, color histogram features are robust to the orientation of the objects in the image. On the other hand, such a selection also brings a disadvantage of misclassification between the classes with similar color. Moreover, different illumination conditions and quantization errors occurring while converting the sensor analog data to digital format cause color histogram feature to become useless.

2.1.1.2 Scale Invariant Feature Transform (SIFT)

Scale Invariant Feature Transform (SIFT) feature is extracted by taking the intensity gradient around some identified keypoints in four steps, namely, 1) scale space extrema searching, 2) sub-pixel keypoint refining, 3) dominant orientation assignment, 4) feature description. Moreover, as it is mostly utilized at sparse keypoints, there is also a dense SIFT technique, which is computed in uniformly and densely sampled local regions, as well as some other extensions, such as PCA-SIFT [34] and speed-up robust features (SURFs) [35]. Similar to color histogram, SIFT features are invariant to rotation. Moreover, as its name suggests, these extracted features are also invariant to scale and illumination. Zhang et al. [36] applied DSIFT with SVM and CRF classifiers to map rural residential areas.

2.1.1.3 GISTs

Using GIST [30], one can extract global representations from the spatial structure of dominant scales and orientations of an image scene by calculating the statistics of the outputs of local feature detectors in spatially distributed subregions. In order to extract GIST representations, the images are first convoluted with a number of steerable pyramid filters, which is followed by dividing the image into 4x4 grids to calculate orientation histograms for each. Being similar with the SIFT feature extraction, it is very simple to apply GIST and the results are quite effective, leading GIST to be applied in several researches for image scene representation [37, 38]. Li et al. combined GIST features with saliency features in [37] and used SVM classifier to detect geospatial objects, such as 'airplanes', 'boats', and 'buildings' in satellite images, whereas Yin et al. [38] used GIST features to detect craters with random forest classifier on Mars Orbiter Camera (MOC) database.

2.1.1.4 Histogram of Oriented Gradients (HOGs)

HOG features are shown to be successful for representation of the edge or local shape information of the objects by computing the distribution of the gradient orientations and intensities in spatially distributed subregions [31]. For image scene classification and geospatial object detection tasks, it has been shown that HOG features are capable to distinguish classes in several studies of Cheng et al. [6, 8, 28]. Moreover, some modified versions for remote sensing problems are also exist as Zhang et al. [39] proposed rotation invariant HOG, namely RIHOG, based airplane detection model whereas Shi et al. [18] proposed Circle-Frequency combined HOG, denoted as CF-HOG feature, to detect ships in high resolution satellite images.

2.1.1.5 Texture Descriptors

Several texture features, mainly local binary patterns [40, 41], gray level co-occurrence matrix (GLCM) [32], and Gabor features [33] are widely used in the earlier remote sensing image analysis studies [14, 26, 27]. Yang et al. [26] compared Gabor texture features with SIFT descriptors and [27] Huang et al. proposed a gray-level co-

occurance matrix textures to discriminate three different mangrove species using multispectral satellite images. These feature are mainly obtained by placing primitives in local image subregions and analyzing the relative differences, which makes them useful enough for image scene classification problems.

2.1.2 Unsupervised Feature Learning Methods

Although handcrafted features performs well for some specific tasks, they are very hard to generalize and it is difficult to obtain a more global feature set to employ for other vision analysis tasks. Hence, later studies in remote sensing image analysis applied unsupervised feature learning method and significant progress has been achieved in image classification [7, 9, 10, 42, 43]. Unsupervised features are learned from either handcrafted features or directly from raw pixel values. Since these learned features are better discriminators, the performance of the classifier is improved. Some of the commonly used unsupervised learning techniques are principle component analysis (PCA) [44], K-means clustering, sparse coding [45] and autoencoder [46].

2.1.2.1 Principle Component Analysis (PCA) based Features

PCA is a linear transformation operation that could be assumed as the first unsupervised feature extraction method. As the name implies, the transformation matrix is formed by orthogonal basis vectors which helps obtain the principle components of the input data. The main goal is to obtain a new representation by preserving the structure of the data [44]. In order to extend this idea, sparse PCA and PCANet algorithms have also been developed. An informative feature selection algorithm based on Sparse PCA is proposed in [42] for high resolution remote sensing image scene classification. In order to obtain robust invariant features for image classification by using PCA, multistage filter banks called as PCANet is proposed in [47]. However, PCA based features are limited features, since the operation is a linear transformation. Hence, more powerful and nonlinear feature learning methods in order to obtain more abstract representations have been developed.

2.1.2.2 K-means Clustering and Bag-of-Visual-Words Representation

The main aim of K-means clustering is to divide a set of data into K clusters such that the within-cluster variance is minimum and between-cluster variance is maximum. First, the number of clusters, K, is randomly determined and the data set is divided into K clusters and the data is assigned to the cluster whose centroid is the closest. Then new centroid of each K cluster is computed and the data is assigned to the closest cluster. This method is iterative until no data changes the belonging cluster. K-means clustering is assumed to be an unsupervised learning method, because the data set do not contain any labeling information. Since this method is very easy to apply, it has been widely used in the image scene classification problems to learn strong unsupervised representations, especially with the Bag of Visual Words (BoVW) based methods to extract visual dictionaries [48, 49, 50]. In Bag-of-Visual-Words representation, a histogram of the indices of the cluster centers of any local representation of the image describes the scene.

2.1.2.3 Features based on Sparse Coding

Sparse coding is an improved version of K-means clustering method to learn higher level feature representations of the input data from unsupervised training samples. Similar to K-means clustering, the input data can be represented as a linear combination of learned bases [45]. However, the extracted features, which are the coefficients of the learned bases, are forced to be sparse; i.e., most of them are forced to be zero, while only a few are allowed to be nonzero. This procedure simply consists of two alternating steps: The first one is the determination of the bases, and the second one is the estimation of the coefficients based on these bases.

Many sparse coding based methods have been applied for the scene classification of remote sensing images [7, 9, 10, 51]. In [7], low-level features are represented as sparse coded features. Multiple feature combination is obtained in [10] for satellite image scene classification by using sparse coding method. In addition to those, the authors in [9] proposed a method using multi-feature joint sparse coding with spatial relation constraint for satellite images. However, this method is computationally quite

expensive, especially in large-scale images. Therefore, Wang et al. [52] proposed locality constrained linear coding (LLC) method by the observation of the fact that nonzero coefficients are usually happened to be close to each other such that the input data can be reconstructed by using K-nearest neighbors among the bases, which transforms the representation from sparsity to locality [24, 52]. Hence, the sparse coded representations, i.e. the coefficients can be computed by solving a least-squares problem.

2.1.2.4 Learned Unsupervised Features : Autoencoders

Autoencoder [46] is formed by a symmetrical neural network to learn an encoded, i.e. compressed, feature representation of the input data as an unsupervised feature learning method that is obtained by minimizing the reconstruction error of the encoded feature. The autoencoders have been successfully applied to land use scene classification [15, 53, 43].

2.1.2.5 Brief Discussion on Feature Extraction Methods

Although the unsupervised feature learning methods succeeded a better performance in land use classification tasks when compared to the handcrafted feature extraction methods, these features still have the lack of semantic information due to absence of labels in the data during the unsupervised learning. In order to achieve a better classification result, one requires more powerful and discriminative features that can be extracted by using supervised deep learning methods.Such features are *learned* to improve the classification performances, on contrary to all aforementioned techniques in this chapter which do not have such a capability.

2.2 Deep Networks on Land Use Analysis

In the recent literature, the most significant classification results are mostly achieved by the use of supervised deep learning methods. In 2006, deep learning took a significant step by the works of Hinton and Salakhutdinov in [46]. Since then, many

of the researchers applied deep feature extraction techniques instead of using handcrafted features in many applications including remote sensing image classification [5, 24, 54, 55, 56, 57, 58, 59, 60, 61].

Obtaining powerful representations of the data with the use of handcrafted feature extraction methods requires domain expertise and significant engineering effort. On the other hand, deep learning method can extract more discriminative and abstract features directly from raw input data via multi-layer architecture of the network. Therefore, deep learning methods are gaining more attention from the researches every other day.

A number of deep learning methods exists and are being developed for many applications including remote sensing image analysis. Some of the popular methods are deep belief networks (DBNs) [62], deep Boltzmann machines (DBMs) [63], stacked autoencoders (SAEs) [64] and convolutional neural networks (CNNs) [2, 65, 66, 67]. The most widely used networks in remote sensing image analysis are SAEs and CNNs.

2.2.1 Stacked autoencoder (SAE)

Stacked autoencoder (SAE) [64] is an unsupervised learning method that consists of multiple autoencoders as hidden layers that are trained mostly using greedy layer-wise training [68] and has been used in remote sensing scene classification and semantic image segmentation tasks successfully [4, 43]. To illustrate this, only the first layer is trained using input and its reconstruction at the first step. Then only the second layer is trained using the first layer as input and reconstruction of the first layer as output. This process is applied for all hidden layers. At the last step, a fine-tuning is performed by training the whole network, but with a smaller learning rate. Moreover, SAEs are proven to be capable of extracting more powerful representations in [4, 43] when compared to a single layer autoencoder. This result is possible, since each hidden layer uses its previous layer as input, which results in obtaining higher level of abstraction at the last layer.

2.2.2 Convolutional Neural Network (CNN)

CNN is a supervised deep learning method that is widely used for image classification and have proven to output remarkable results. AlexNet proposed by Krizhevsky et al. in [2] succeeded an impressive amount of an increase in the ImageNet classification task. Similar to AlexNet, many other convolutional neural networks are proposed including VGGNet [65], GoogLeNet [66], SPPNet [69] and ResNet [67]. A CNN is typically formed by a 3 dimensional input layer, convolutional layers, pooling layers, normalization layers, fully connected layers and an output layer. CNN is a powerful feature extractor, since it takes the advantage of the properties, such as local connections, shared weights, pooling and the use of many layers [70].

Following the success of AlexNet, learning representations using CNNs became very popular and many solutions are proposed using CNNs in different image analysis problems, including remote sensing applications, such as land use scene classification [24, 54, 55, 56, 57, 58, 59, 60, 61], land cover segmentation [4, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81]. As a prominent example, Castelluccio et al. [60] and Nogueira et al. [61] examined the use of CNN in remote sensing image scene classification by first training the network from scratch, then fine-tuning with the ImageNet trained weights. Moreover, the last layer of the CNN is also used as extracted feature vector for an image and these features are used in different classifiers, such as Support Vector Machines (SVMs). It is shown that using pretrained weights and fine-tuning with new data resulted in being the best classification method, especially when dealing with limited small-scale datasets. On the other hand, Penatti et al. [24] used UC Merced Land Use (UCML) dataset and Brazilian Coffee dataset to compare traditional methods with deep networks. Sevo and Avramovic [82] applied GoogLeNet on UCML classification dataset for land use scene classification. Basu et al. [83] proposed SAT-4 and SAT-6 datasets with 28x28x4 size and compared DBN, SAE, LeNet-5 and their proposed DeepSat framework for barren land, buildings, grassland, roads, trees, waterway, and other. Zhong et al. [84] also used SAT-4 and SAT-6 and proposed SatCNN framework as classification network using Agile CNN and achieved 99.65% for SAT-4 and 99.54% for SAT-6. On the other hand, Sherrah [73] applied Fully Convolutional Network (FCN) with and without downsampling during pooling on ISPRS

Vaihingen dataset which has 5 classes as impervious surface, building, low vegetation, tree, and car. Audebert et al. [75] applied SegNet with multi-kernel ensemble and composite data fusion methods on ISPRS Vaihingen segmentation dataset. Kaiser et al. [78] implemented a variant of FCN on several datasets. Maggiori et al. [77] used two-scale FCN with Pleiades dataset.

2.3 Related Works on Geospatial Object Analysis Using Deep Networks

Although object-based image analysis methods showed an important increase in satellite images, deep learning based methods have attracted many researchers to focus on remote sense problems after the success achieved by a convolutional network, namely AlexNet proposed by Krizhevsky et al. [2]. Initially, for object detection tasks, region-based convolutional neural networks are proposed [85]. In those networks, one network is trained to propose candidate regions, and another network is trained and used to detect targets in the proposed regions. One implementation of region-based method to detect aircraft is developed by Han et al. [86]. Igor et al. [82] implemented a GoogLeNet based two-stage network in UCML dataset using image scenes instead of bounding boxes. A similar work performed by Yamamato and Kazama [87] to detect image scenes containing ships in satellite images by VGG based convolutional neural networks. Long et al. [88] proposed a two-stage unsupervised score-based bounding box regression (USB-BBR) method for object detection, while Xu et al. [89] utilized deformable ConvNets based on Region-based Fully Convolutional Networks (R-FCN) in aerial image object detection.

On the other hand, end-to-end training networks are also developed to detect objects in a single network. Such networks are designed and utilized for non-aerial images. YOLO network [90] is a single stage convolutional neural network with several hidden layers. The network divides the input image into 7x7 cells, and make prediction for each cell. Radovic et al. [91] used YOLO network to detect aircraft in UAV images, whereas Carlet and Abayowa [92] compared the performances of both Faster R-CNN and YOLO on vehicle detection in aerial imagery. An improved version of YOLO, also called YOLOv2 or YOLO9000 [93], uses fully convolutional layers (instead of fully connected layers) before the output layer and applies 5 anchor boxes for

each cell to make a prediction for the bounding boxes. Another single shot method, namely Single Shot Detector [94] (SSD), similarly uses several grid cells with different sizes such as 4x4 or 8x8, where the larger boxes predict the images with greater size and narrower boxes predict the smaller size images. Chen et al. [95] applied a modified version of SSD network to detect aircrafts for satellite images. Tang et al. [96] modified SSD to obtain oriented bounding boxes around vehicles in aerial images.

Since the processing speed of end-to-end training methods are much faster than region-proposal methods, in our study we focused on modified versions of YOLOv2 network to detect geospatial objects from satellite images in a preceding chapter.

CHAPTER 3

FUNDAMENTALS OF DEEP NEURAL NETWORKS

Machine Learning, which is a branch of Artificial Intelligence, has been widely used in various computer vision problems to model complex tasks such as scene classification object recognition. In the past the problems were modeled by hand in the computers. However with machine learning, the solutions of many difficult problems can be improved by replacing traditional handcrafted algorithms with a learning algorithm. As the availability of the data increases each day with the help of advanced sensor technology, machine learning is getting more popular and practical for especially computer vision problems.

Artificial Neural Networks (ANNs), a popular type of machine learning, are inspired from the biological neurons to model input-output relationship. As in the biological neurons, an artificial neuron takes number of inputs, evaluate them as a matrix multiplication, and output a single value with an activation function.

$$y = f(\mathbf{w}^T\mathbf{x} + b) \tag{3.1}$$

where y is the output, $f(.)$ is the activation function, $\mathbf{w}$ is weight vector and $\mathbf{x}$ is the input vector. b term is the *bias* parameter which adds non-linearity to the model before the activation. Typical non-linear activation functions are given in Figure 3.1.

A neural network is formed by using multiple layers consisting multiple neurons between input and output layers. Layers between input and output layers are called hidden layers. In a feed-forward neural network, each neuron in hidden layer collects signals from the neurons in the previous layers and generates an output according to

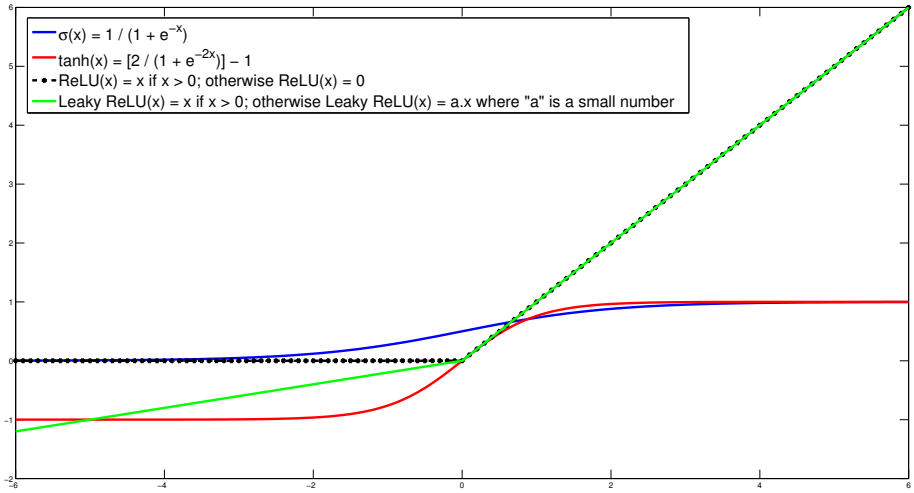


Figure 3.1: Widely used activation functions

the activation function. Fully connected neural network consists of the neurons in each layer connected both to all of the previous layer neurons and the next layer neurons. A simple feed-forward fully connected multi-layer neural network is illustrated in Figure 3.2.

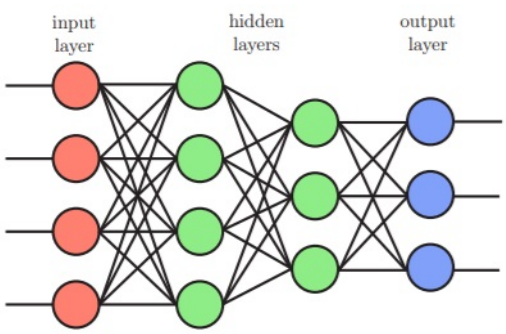


Figure 3.2: A simple feed-forward multi-layer (artificial) neural network (NN)

In a fully connected feed-forward neural network, the input layer takes the data as input and propagates to the next layer. The hidden layers are where the extensive computations take place in a neural network. The last layer, which is the output layer, extracts the high-level learned representations, or makes classification predictions, depending on the purpose. It is shown in [97] that a neural network with at least one hidden layer can be constructed such that it can compute almost any function.

Neural networks take the input, make some non-linear operations with the weights and output the values in the output layer. Since the operations are non-linear, it is not possible to obtain an analytic model for a network to find the optimum weights. Therefore, in order to train the network to perform a specific task, for example, classification, the weights, i.e. the connection weights of the neurons must be trained. This learning process can be achieved by defining a loss function of the output layer and minimize the loss function with respect to the connection weights. A typical loss function that is widely used in machine learning algorithms is mean squared loss function as shown in the equation 3.2.

$$loss = \sum_{i=1}^{N} (\widehat{y}_i - y_i)^2 \tag{3.2}$$

The *The back-propagation algorithm* [98, 99] is a very effective way of training the network weights in an iterative manner. Many optimization techniques such as Stochastic Gradient Descend [100], Momentum Optimizer [101], Adam Optimizer [102] exists in the literature. The basic idea is that the loss value calculated at the output layer is minimized with the backward propagation of the gradients towards the input layer. To explain in detail, the weights of the network is initialized randomly and an input is fed to the network. After the computations in the hidden layers, the output layer makes a prediction. This output is compared with the desired output in a loss function. The loss value is used to compute the gradient at the output with a chosen optimization method. At last, the gradient at the output is backward propagated through the network towards the input by using chain rule of derivatives and the connection weights are updated according to the learning rate. Learning rate is a small number that determines what portion of the gradient should be used for the weight update. A very small learning rate might cause the network train very slowly, even diminish the update, while relatively large learning rate would cause the network get stuck in the local minima, or even cause to diverge.

Learning process can be explained under three categories. The first one is the *online learning*. A single input data is fed to network and weight update is done after each input in online learning. Although this is a very fast and computationally low cost method, zig-zag movement of loss value can be observed due to the lack of predicting

the global gradient with a single input. The second one is *full batch learning*, where the gradients are computed and the network is updated after the whole dataset is fed to the network. Given the advantage of having a better prediction of global gradient with this method, the computation burden is very heavy, even impossible with large-scale input size and dataset. Hence, there is the third method which is called as *mini-batch learning*. In this method, after a small portion of the dataset is fed to the network, the gradients are computed and the network is updated.

Deep neural networks basically consists of many hidden layers between input and output layer. Training many layer network is called *deep learning*. As the depth of the network increases, higher-level representations can be extracted. However, the increasing depth may cause the gradients to vanish as they propagate towards the previous layers. Also, the number of samples in the training set should be significantly large to train very deep networks Moreover, the computational cost could be too high for very deep networks such that training the network would take several months, which makes very deep networks unpractical.

Recent advances in GPU technology and increasing number of dataset available makes deep learning practical and many researchers have been focusing to replace the traditional feature extraction methods using deep neural networks in several applications. In [2] it is shown that the first layers of deep networks extract feature that are computed by hand in earlier works to detect edges and blobs in an image. Hence, deep networks extract basic features in the earlier layers and more abstract and discriminative features can be learned in the deeper layers of the networks.

3.1 Fundamental Deep Architectures

Recent studies on audio recognition [103], natural language processing [104] and computer vision fields [2] outperformed the traditional methods with use of deep architectures. The idea behind is the imitation of deep mammal brain perception system such that the received input is hierarchically processed in the different parts of the system. For example in the visual system, multiple levels of abstractions are detected in [105].

So far, many researchers proposed number of deep architectures such as Deep Belief Networks (DBNs) [62], Stacked Autoencoders (SAEs) [43] and Convolutional Neural Networks (CNNs) [99]. DBNs are shown to achieved a great success when they are trained layer-wise using Restricted Boltzmann Machines (RBMs) [106]. Later, Sparse SAEs and denoising SAEs were developed to learn high level representations. In addition to those, the most popular architecture among the computer vision researchers that is still widely used to achieve incredible results is proposed by LeCun et al. in [99] and referred as Convolutional Neural Network (CNN). Nowadays CNN based methods outperform almost all of the traditional methods in several computer vision tasks such as image scene classification and object recognition since CNNs are very effective and powerful high level feature extractors when they are trained properly. More detailed information about these deep networks can be found in the following subsections.

3.1.1 Deep Belief Networks (DBNs)

Several stacked RBMs that are trained layer-wise are used while designing a deep belief network as can be seen in Figure 3.3. An RBM consists of two layers with visible layer and hidden layer. Input is fed in the visible layer and abstract features are learned in the hidden layer by minimizing the energy function given by

$$E(v, h : \theta) = -\sum_{i=1}^{N} b_i v_i - \sum_{j=1}^{M} a_j h_j - \sum_{i=1}^{N} \sum_{j=1}^{M} w_{i_j} v_i h_j \tag{3.3}$$

where $\theta = \{w_{i_j}, a_j, b_i\}$, w_{i_j} is the weight between i^{th} visible unit v_i and j^{th} hidden unit h_j, N is the number of visible units, M is the number of hidden units and a_j and b_i terms are the biases of hidden unit h_j and visible unit v_i, respectively.

RBM tries to reconstruct the input vector via hidden units and the weights by minimizing the energy function with the predicted probabilities of the training vector. The conditional distributions of input vector **v** and hidden units **h** are given by the logistic function

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

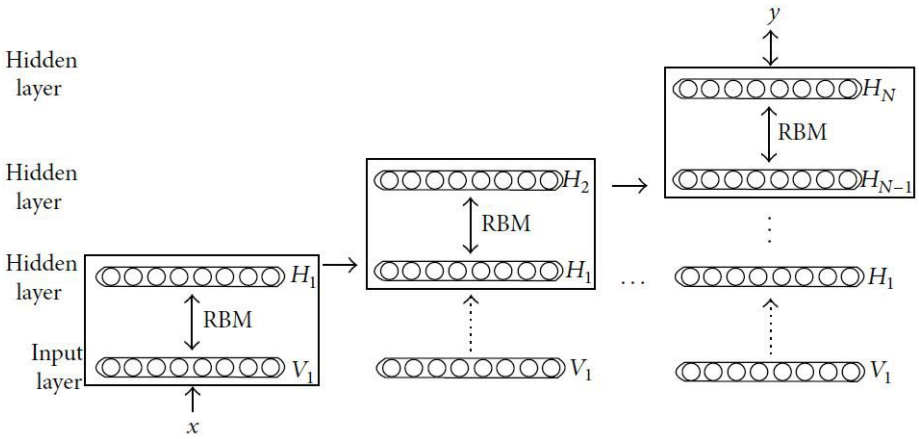


Figure 3.3: A simple Deep Belief Network (DBN) structure, adapted from [1]

$$p(v_i = 1|\mathbf{h}) = f(\sum_{j=1}^{M} w_{i_j} h_j + b_j) \tag{3.5}$$

$$p(h_j = 1|\mathbf{v}) = f(\sum_{i=1}^{N} w_{i_j} v_i + a_i) \tag{3.6}$$

The input vector is reconstructed after the hidden units are determined with the probability of each unit equals to 1 as in Equation 3.5. Then, the hidden units are updated to obtain a better reconstruction. The weights are learned by *contrastive divergence (CD)* method.

In DBNs, the first two layers are trained as an RBM. Then, the next RBM is trained using the hidden layer of the first layer as visible layer. This process, which is called layer-wise training, is followed until the last RBM is trained. At the last step, a fine-tuning supervised training is applied with limited number of labeled data. It is shown in [107] and [108] that DBNs used in remote sensing classification problems outperforms the classifiers such as support vector machines (SVMs) and conventional feature dimension reduction methods such as principle component analysis (PCA). Moreover, DBN-based methods have been developed successfully in [109] for object recognition and in [110] for scene classification in remote sensing images.

3.1.2 Stacked Autoencoders (SAEs)

Another unsupervised learning method in deep learning is to use autoencoders (AE) adjacent to each other in order to obtain a deep structure. An AE is a neural network that looks similar to Restricted Boltzmann Machines however works in a different way such that an AE tries to minimize the reconstruction error of the input in a deterministic way while an RBM structure predicts the joint distribution of visible and hidden layer. Stacked AEs consist of odd number of hidden layers such that the structure is symmetric, i.e. the number of encoding layers are equal to the number of decoding layers as shown in Figure 3.4.

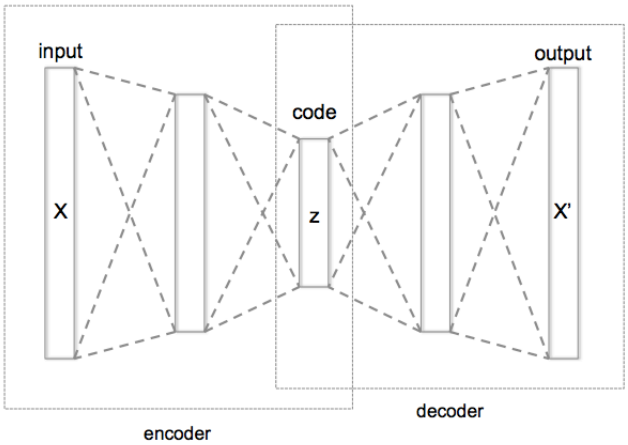


Figure 3.4: An example of Stacked Autoencoder

Like in the DBNs, the feature extracting weights of SAEs are learned using layer-wise training method in an unsupervised manner. The extracted features of the input is encoded in the "code" layer, which is in the center of the symmetry. In the encoding part, N-dimensional input vector is transformed to K-dimensional feature vector using a non-linear function as shown in Equation 3.7

$$z_i = f(W_1 x_i + b_1) \tag{3.7}$$

where z_i is the K-dimensional encoded feature of N-dimensional sample x_i in the training set, b_1 is the K-dimensional bias vector, W_1 is KxN encoder weight matrix,

and $f(.)$ is usually a sigmoid function as given in Equation 3.4.

In the decoding part, a similar procedure is followed to reconstruct the input with the Equation given in 3.8

$$x_i' = W_2^T z_i + b_2 \tag{3.8}$$

where x_i' is the N-dimensional reconstructed data, b_2 is the reconstruction bias vector, and W_2 is KxN decoder weight matrix.

Weight parameters of the AE is learned by minimizing the loss function given in Equation 3.9 below

$$J(X, X') = 0.5(\sum_{i=1}^{M} ||x_i' - x_i||^2 + \lambda ||W||^2) \tag{3.9}$$

where the first term is the reconstruction error, the latter term is called "regularization term", X is the input data, W is weight matrix, and X' is the reconstructed version of the input.

In order to have a sparse AE, let $\hat{\rho} = \frac{1}{M}\sum_{i=1}^{M}[z_i]$ be defined as the average activation of z averaged over the training set. Enforcing $\hat{\rho} = \rho$ where ρ is the sparsity parameter very close to zero, a sparsity constraint for each neuron $\hat{\rho}$ in the encoding layer can be satisfied. Hence, adding this sparsity term to the loss function gives

$$J(X, X') + \beta \sum_{j=1}^{K} KL(\rho||\hat{\rho}) \tag{3.10}$$

where β is the sparsity penalty parameter, K is the dimension of feature vector, and $KL(.)$ is the Kullback-Leibler divergence given by

$$KL(\rho||\hat{\rho}) = \rho \log \frac{\rho}{\hat{\rho}} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}} \tag{3.11}$$

In remote sensing image analysis, SAEs are mostly used [111, 112] as well as single AE as feature extractor [15].

3.1.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are first proposed in the late 1990s and the most popular of the time was applied to recognize handwritten digits in document by Y. LeCun et al [99]. The proposed network was called as LeNet-5 and given in Figure 3.5

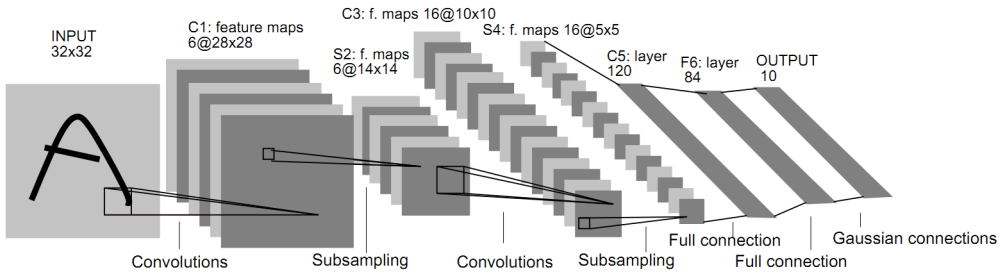


Figure 3.5: An example of Convolutional Neural Network (LeNet-5)

The main difference of a CNN when compared to fully connected neural network is that CNN architecture is designed to receive 2D or 3D volume of the image as input to the network, whereas in fully connected neural network receives the input as 1D feature vector. By keeping the shape of the input image, spatially-local correlated information of the input image can be exploit by CNN using local connectivity with weight sharing. This also reduces the number of weights and enables to train a CNN with less amount of training data that is required to train a fully connected multi-layer neural network. A typical CNN is composed of three different layer types: 1) convolutional layer, 2) pooling layer, 3) fully connected (dense connection) layer.

Convolutional layer

The term *convolutional* is inspired from the convolution operator in mathematics, which is a point-wise integration operation between two functions. A convolutional layer contains many feature maps which are the results of 2D convolution operations of a *convolution kernel*, or *filter*, or sometimes called *receptive field* with the input map. Each receptive field is convolved with the input map to produce a feature map. An activation function is followed by each convolution layer to add non-linearity to

the model. In CNNs the most common activation function that is successfully used in many computer vision tasks is a rectified linear unit (ReLU).

Let **W** be the receptive field of size c x c x k to be used in a convolutional layer where c is the receptive field size. Assuming that the **X** be the input map of size m x m x k where m is the number of rows and columns and k is the number of channels of the image, p is the number of padding, s is the number of stride **Y** is the output feature map of size $(\frac{m-c+2p}{s}+1)x(\frac{m-c+2p}{s}+1)$ can be obtained by using the 2D convolutional operation as given in Equation 3.12.

$$\mathbf{Y}^j = f(\mathbf{W}^j * \mathbf{X} + \mathbf{b}^j) \tag{3.12}$$

where j is the number of filters, i.e. the number of desired feature maps, $*$ operation is 2D convolution, $f(.)$ is the non-linear activation function, and $\mathbf{b}^j$ is the bias parameter. An illustration of 2D convolution is provided in Figure 3.6

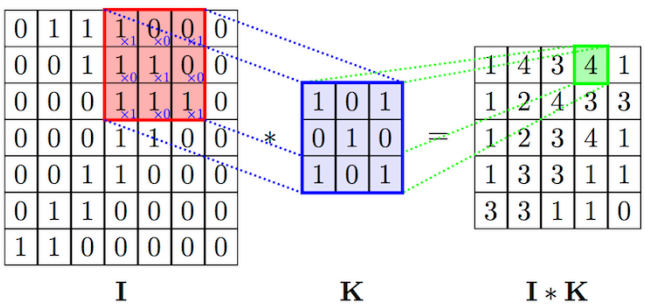


Figure 3.6: 2-dimensional convolution example with filter size 3x3 and stride 1, with zero padding

Hierarchical feature learning property of CNN is obtained by using multiple convolutional layers. The first layer filters learn to extract simple edges, whereas the second layer filters learn more complex features composed of basic shapes of edged. By going deeper in the network, more abstract and high-level features can be learned.

Pooling layer

This layer is another main difference of a CNN from other neural network architectures which basically applies a down-sampling operation over the feature map. By

choosing a window size nxn and stride number s, a spatial dimension reduction of the feature map can be applied by pooling operation by preserving the most of the spatial information. Moreover, pooling operation relieves the memory burden on the computation hardware, which allows to allocate that free memory to increase the depth of the network. Another advantage of pooling is the resultant feature map is more robust to transnational variance and noisy data.

There are two types of pooling operations which are widely used in deep networks. The first one is average pooling, which basically takes the average of the feature map underlying the mapping window. It should be noted that this operation may yield a suppressed feature map when used with tanh activation function as positive and negative features might cancel each other. The other widely used pooling type is max pooling, where the maximum of the feature map overlapped with pooling window is calculated. Max pooling is more likely to overfit the network, even though it does not have the problems of average pooling. However, this disadvantage of max pooling operation can be avoided when combined with other kind of regularization methods. Figure 3.7 shows a sample use of average pooling and max pooling layer.

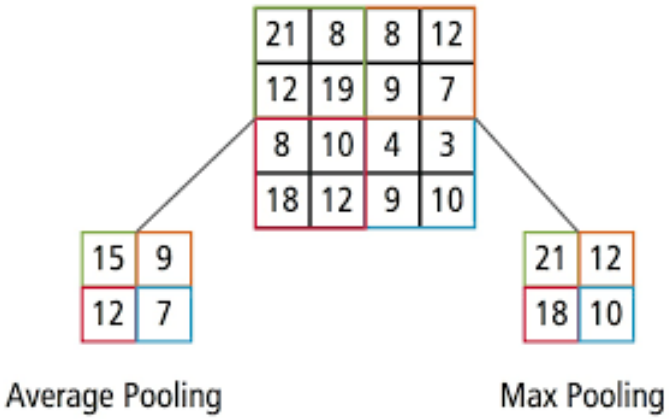


Figure 3.7: Average pooling and max pooling operations with 2x2 filter and stride 2

Fully connected layer

Fully connected (FC) layers are placed just before the output layer of CNN, where they act as regular neural network. As the name suggests, all of the neurons in FC layer is connected to all of the neurons in the previous layer. The last FC layer behaves

as the classifier, whereas the previous FC layers tries to extract some interesting relationships that convolution layer cannot capture due to local connectivity and weight sharing characteristics of receptive fields in convolution layers.

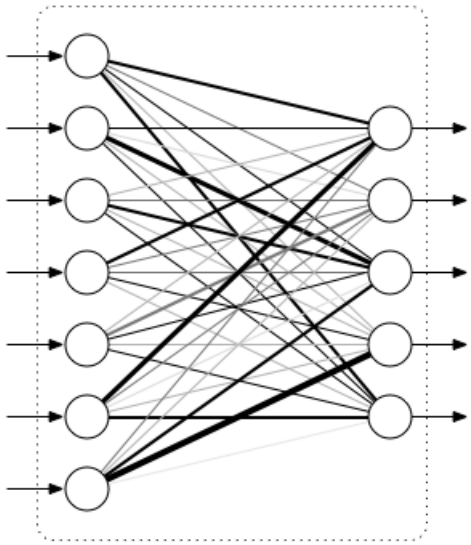


Figure 3.8: Fully connected layer

Since all neurons are connected to each other between layers in FC layer as in Figure 3.8, there exists so many number of trainable parameters. The feature map size of the last layer before FC layer should be down-sampled enough in order to have reasonable number of connections to train FC layers practically. This requirement also makes use of the pooling layers where the feature maps dimensions are reduced.

Fully convolutional layer

Fully connected layers have fixed number of weights that does not apply when the input size is changed. To be more precise, a fully connected layer has NxK number of weights where N is the input feature vector dimension and K is the output vector. However if input vector dimension is changed from N to M, the same FC layer is not applicable since NxK $\neq$ MxK. This disadvantage of FC layer can be overcome by using K number of 1x1xN convolutional filters with stride 1 because convolution layers can accept any size of input map. This approach is widely used in semantic image segmentation problems and called as *Fully Convolutional Network* since the network does not include any FC layer.

Deconvolutional layer

In segmentation networks, every single pixel needs to be classified. Since CNN downsamples the input due to the presence of pooling layers, a pixel-wise classification is not possible directly on the output of an ordinary CNN. Therefore, deconvolutional layer plays a crucial role in semantic image segmentation networks because it is used to upsample the downsampled output classification map of a CNN to have the output map size equal to input image size as shown in Figure 3.9.

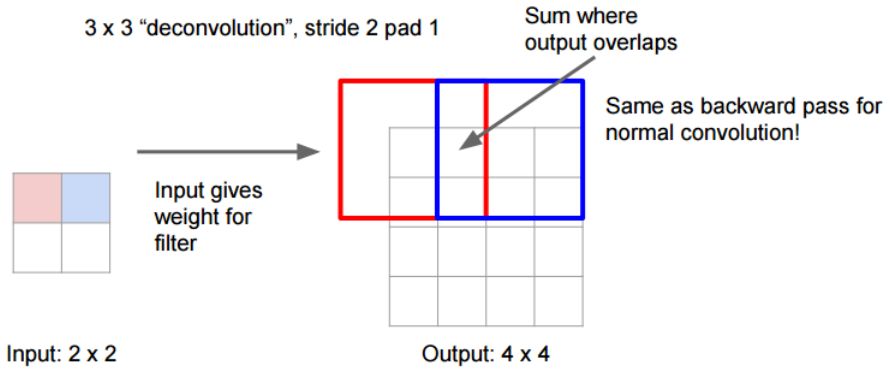


Figure 3.9: Deconvolution operation example

A simple resizing operation is an option to upscale the resolution of an output map. However, it is not as effective as a trainable upsampling convolutional layer. Deconvolution operation can also be considered as transpose of a convolution operation, i.e. the output of a convolution becomes the input of the deconvolution and the input of a convolution becomes output of the deconvolution.

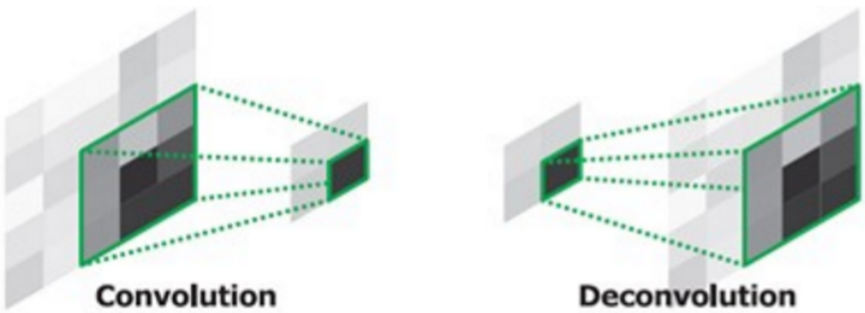


Figure 3.10: Deconvolution illustrated with convolution operation

3.1.4 What makes CNN most attractive?

The main motivation behind the existence of CNNs in deep learning applications is to point out number of limitations that traditional neural networks are exposed in applications such as image classification. Old-fashioned fully-connected neural networks simply cannot scale well because of their huge number of connections. CNNs contribute with some brilliant ideas to improve the efficiency of deep networks. The two important fundamental principles leveraged by CNNs are 1) Sparse Representation and 2) Parameter sharing.

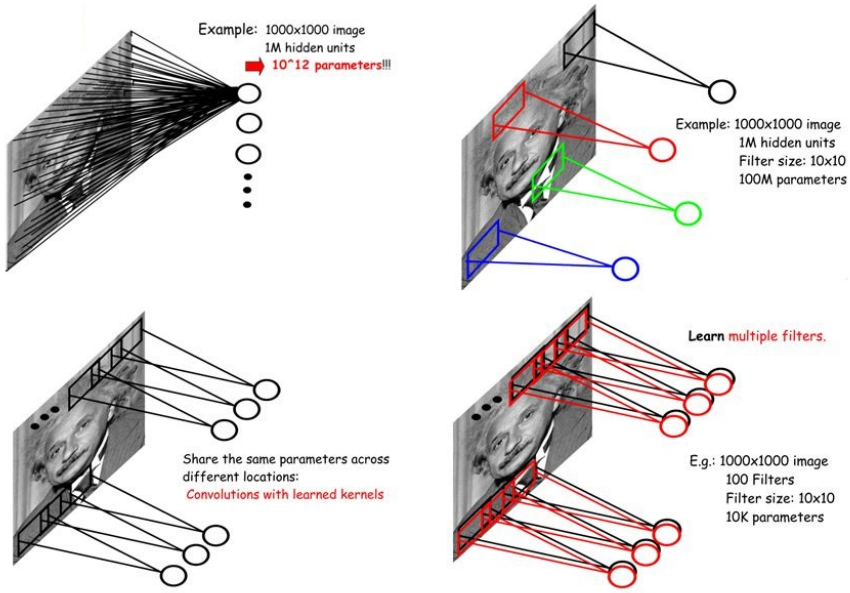


Figure 3.11: CNN exploits local correlation by using sparse representation and parameter sharing principles

Sparse Representation

Assuming an image classification problem that involves the analysis of large pictures of millions of pixels, traditional neural network models the information using matrix multiplication operations of every single input pixel and every single parameter, resulting in tens of billions of computations where CNNs are based on convolution operations between input data and a convolutional filter. By doing so, it turns out that the convolutional filter parameters are significantly less than the input data. This

operation of CNN simplifies the number of computations required to train the model or to make predictions.

Parameter Sharing

Another important property of CNNs is the parameter sharing. Conceptually, parameter sharing simply means that CNNs use the same filter across every position of the input data which will allow the model to learn a single set of weights once, resulting in less memory use compared to traditional models.

3.2 Training Convolutional Neural Networks (CNN)

Deep Neural Networks, especially Convolutional Neural Networks (CNN) are highly non-linear complex mathematical models that can extract high level representations of an input image data. CNNs proved incredible state-of-the-art results on many computer vision tasks such as image scene classification, semantic segmentation and object detection in many domains including remote sensing image analysis. In this section, training a CNN is described to obtain accurate results by getting most of the CNN.

3.2.1 Data augmentation

Since deep networks are consisted of huge number of weights to map input data to output, again a huge number of input data is required for the task. As can be seen in Figure 3.12, bigger amount of data increases the performance of deep network, whereas the traditional algorithms saturate with the performance for a bigger amount of data.

Finding labeled remote sensing dataset is not very easy due to the economical costs and commercial restrictions. Therefore, the number of data in the training set must be increased in order to train a CNN. Increasing the number of data in training dataset is called as *Data Augmentation*. Data augmentation is basically a process of increasing the number of training dataset with some operations such as rotating the input image,

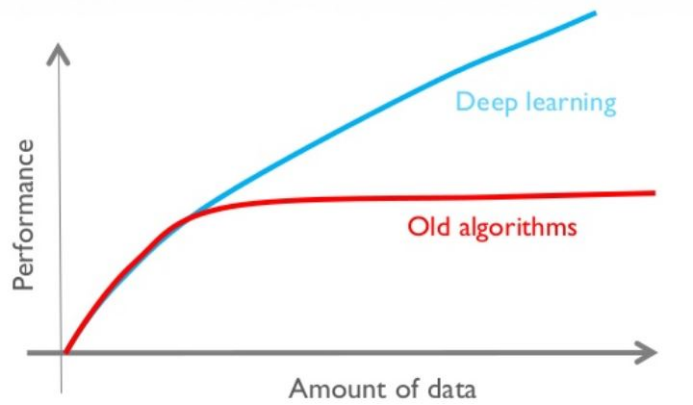


Figure 3.12: Performance of DNN increases as the amount of training data increases

flipping horizontally and vertically, jittering the colors of the image, randomly cropping the image, random zooming and re-sizing etc. To illustrate the process, assume the training image set of a remote sensing scene classification problem has 200 images for 2 different classes, say 100 images belonging to class 1, *sea*, and other 100 images belonging to class 2, which is *tree*. Simply by just having 4 times 90 degree rotations 4 x $200 = 800$ images can be obtained. If horizontal and vertical flipping applied to the new dataset, 4 x $800 = 3200$ image can be obtained in the training set. Starting from 200 data, it is certain that better results can be obtained by combining original 200 data with generated 3000 data using data augmentation technique.

3.2.2 Preprocessing on images

The optimized Deep Neural Networks for a given tasks usually have weights of very small numbers. The input images of a CNN usually has pixel values in an interval of $[0, 255]$, which are relatively large numbers. In order to boost the performance of the network, *normalization* and *zero-centering* of the training dataset can be applied. Zero-centering can be done by extracting the mean value of the training set from each images, and normalization can be done by divining each input data to the standard deviation of the training set so that the pixel values of training set is mapped between $[-1, 1]$.

3.2.3 Initialization of network weights

Before training a CNN, some values should be assign to the weights of the network to make loss calculation at the output layer.

By intuition, it can be considered that all the weights are initialized with *zero*. However, if each neuron outputs the same value, then the same gradients will flow backward through all the neurons during backpropagation, which will cause a same amount of update for each of the neurons. This initialization will not break the zero-symmetry of the network, therefore will fail.

One might think of initializing the network weights randomly, which would be a good idea. The better one would be that the network weights should be initialized with samples drawn from zero-mean unity variance Gaussian distribution, scaled with a small value, say 0.001. Since there will be no symmetry as in all-zero initialization, each neuron will get a different amount of update during the gradient flow and the network will eventually be trained to reach the optimum value.

When going deeper in detail with the above initialization, it can be realized as a problem that the variance of the output distribution of a randomly initialized neuron is getting higher with the increasing number of inputs to that neuron. In order to solve this problem, the variance of each neuron's output can be normalized by calibration the variances with $1/\sqrt{n}$, where n is the number of inputs so that the network when initialized will have approximately the same output distribution, which will improve the the convergence time. Taking another step with the initialization of the weights, He et al. proposed in [113] that the calibration of the variances of the neurons should be scaled with $\sqrt{2/n}$ to achieve a better initialization, especially when using ReLUs as the activation function.

3.2.4 Hyperparameter selection

In machine learning, there are several parameters to be selected before starting the training process. If it is a deep neural network, the number of parameters is a long list, so called hyperparameter optimization. There are three main methods to tune

hyperparameters. The first one is using *grid search*, where all the hyperparameters are listed with prior intervals and all the possibilities are selected and the network is trained for a short time period. Then the combination of which gave the better result is selected as optimal hyperparameter values and the network is trained using them. However, this procedure is a very exhausting and time-consuming, therefore not much preferred. The second one is *random search*, where a set of hyperparameter values are chosen and the network is trained for a small number of iterations. Then another set of hyperparameters are randomly selected on the hyper-sphere of the previous set with smaller hyper-radius and the network is trained for a short time. If the result is better, another set is randomly selected on hyper-sphere of newer set with even smaller hyper-radius. If not, another random point is selected using the previous set. This procedure is applied for a certain of iteration, then the best set is chosen. The third method is *Bayesian hyperparameter optimization*, where a prior over hyperparameter distribution is selected and sequentially updated with observation of coming experiments. Some important parameters are listed and discussed below.

Filter and pooling size

Although it seems as if the size of convolutional filters and pooling windows should depend on the feature map of interest and ,indirectly, the size of the input image, in [65] it is shown that instead of using 5x5, 7x7 or larger receptive field size, 3x3 size convolutional filters are able to capture the adequate information from the feature maps. Using smaller size filters has another advantage of having smaller number of weights in the network to be trained while increasing the performance of the network. In most cases, preserving the feature map size in convolutional layers is important because the pooling layers divide the size of the maps by 2 or 3 or another choice, where the size of the map should still be available to be divided into an integer value in the pooling layer. Therefore, it is very common that in the convolutional layer the feature map is padded accordingly to keep the size of the feature map the same. As an example, by choosing input size multiple of 2^n, where n is the number of pooling layers, convolution kernel size 3x3 with stride 1 and zero-padding and the pooling window size 2x2 with stride 2, one can construct a proper CNN with unlimited convolutional layers and at most n pooling layers deep.

Mini batch size

Deep networks are composed of huge number of weights. Hence a huge number of data, which could be considered as *Big Data*, is required to train the network. However providing such big data to the network at once to calculate the global gradient of the dataset is impossible due significant amount of memory requirement and hardware considerations. One possible solution is mini batch training, where only a small part of the training data is provided to the network in order to compute the local gradient. Using small number of batch size causes a high variance of gradient estimation between every batch. In that case, the plot of computed loss value looks very noisy as in Figure 3.13. It is usually recommended that the mini batch size should be selected as high as the hardware can support.

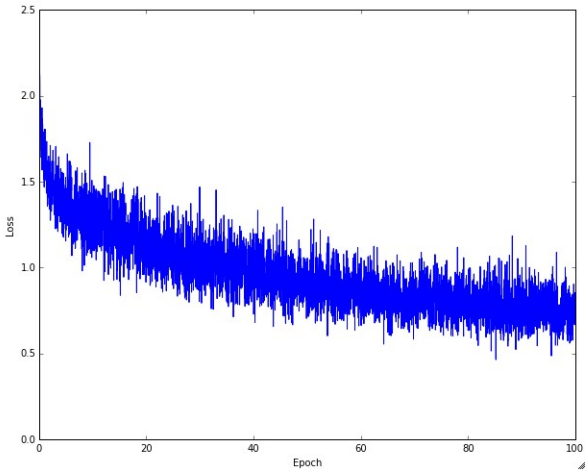


Figure 3.13: Loss vs Iteration plot looks very noisy for small number of batch size

Adjusting learning rate (LR)

Learning rate is another important parameter to be tuned for the best performance. This parameter is the multiplication coefficient of the computed gradient, in other words, step size of the weight updates, which plays a crucial role in training a deep network. Choosing a smaller learning rate causes the training to take very long time. On the other hand, a larger learning rate may cause the network to be stuck in a local minimum even though the loss decreases faster. A much higher learning rate might even cause the loss to diverge. Therefore, optimal selection of learning rate effects the

performance of the network. Typical learning rate values vary between 0.1 to 1e-5. Figure 3.14 shows how a good training loss should decrease.

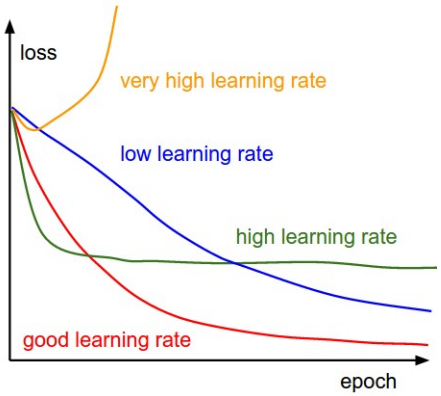


Figure 3.14: The effect of learning rate on training loss of a deep network

Selection of activation function

A neural network without non-linear activation function is just a simple cascaded matrix multiplication and addition operation, which means the network with many number of layers can be represented with a single layer neural network. For this issue, a non-linear activation function is called within each neuron. This phenomenon is inspired from the biological neuron activity.

Several important activation functions are mentioned in the beginning of this chapter (check Figure 3.1). In this section, those activation function as discussed in more detail.

Sigmoid function is widely used in neural networks for decades since it has a simple implementation and nice representation of neuron firing, i.e. if the input to the sigmoid function is relatively large number with negative sign, the neuron output is zero (not fired), and if the input is relatively large number with positive sign the output is one (fired). The input values around zero point outputs a relatively linear output. However, sigmoid activation function is rarely used nowadays because of two main drawbacks:

1. It is very important for a neural network while training that the gradients from the output layer should flow through the input layer without being killed. Since

sigmoid function saturates for very large number of negative and positive values, the local gradient of that neuron becomes too small such that the flowing gradient from upper layer cannot backpropagate through the lower layers. This undesirable event is called as *vanishing gradient problem* and it is a very serious one that comes out when using sigmoid activation function.

2. The non-zero output of the sigmoid function introduces a zig-zagging motion of gradient update. Since the output of the function is always greater than zero, i.e. positive, the gradient of the weights for the neuron in descendant layer is either all negative or all positive, which causes the zig-zagging motion. However, due to the mini-batch optimization, this effect is reduced with the sum of the gradients in a mini-batch.

Hyperbolic tangent function, tanh(x), which is another activation function used in neural networks shows similar characteristics with sigmoid function such as vanishing gradient problem due to activation saturation. However, unlike sigmoid, tanh concentrates its input data between [-1,1] output value. Since this function gives a zero-centered output, tanh is preferred to the sigmoid.

Rectifier Linear Unit, also known as ReLU, has been the most popular activation function preference for the last few years. It has a very simple mathematical formula as $f(x) = max(0, x)$, which is also considered as *thresholding at zero*. When compared to the previous activation functions, ReLU is inexpensive to operate and does not suffer from saturating. However, neurons with ReLU activation might *die* forever because ReLU performs as a transition gate to the gradients such that if the input to the neuron is positive, the gradient is flowed to the input neurons. On the other hand, the gradient flow is stopped if the input to the ReLU activated neuron is negative. This issue is mostly encountered with an improper, highly selected learning rate. Being careful with the learning rate may mitigate the possibility of the neurons to die.

Some attempts such as Leaky ReLU, Parametric ReLU, and Randomized ReLU have been made to deal with the *dying neuron* problem in ReLU activated neurons. In ordinary ReLU, $f(x) = x$ if $x > 0$ and $f(x) = 0$ if $x \leq 0$, where for other members of ReLU family $f(x) = x$ if $x > 0$ and $f(x) = \alpha x$ if $x \leq 0$ where α is set for a small

value for Leaky ReLU, learned by the network in Parametric ReLU, or sampled from a random distribution for Randomized ReLU. In any case, modified ReLU functions reduce the dying neuron number and improve the overall performance.

3.2.5 Regularization Methods

Overfitting is the phenomenon of modeling the training set too well such that the model cannot make accurate predictions for the unseen test data, i.e. it cannot generalize. There are several ways of avoiding overfitting the neural network such as L1 regularization, L2 regularization, Max norm constraints, Batch normalization and Dropout.

L1 regularization

A term $\lambda|w|$ is added to the loss function, in which the weights are forced to become sparse during optimization. That is to say, L1 regularization ends up having noise invariant network by using sparse subset of the most important inputs.

L2 regularization

The most commonly used regularization methods could be L2 regularization, where $\frac{1}{2}\lambda w^2$ term is added to the loss function. The gradient of this term is λw, which implies that higher values of the weights are penalized more than smaller ones, resulting in a diffused, non-peaky weight vectors.

Max norm constraints

This type of regularization is barely used, but slight improvements are reported on several applications. The idea here is to bound the magnitude of the weight vector after weights are updated such that $||\mathbf{w}_2|| < c$ where c is the constraint value, typically on orders of 3. The beauty of this regularization is that the weights cannot explode even if the learning rate is set too high since the updated weights are always bounded.

Batch normalization

Batch normalization is not an obvious regularization method, but it can be mentioned under this section since it is another way of improving the network performance similar to other regularization methods. The idea comes from the normalization of input parameters, where a neuron of input layer takes values between 1 and 100, and other neuron could take values between 0.01 to 1, for instance. This normalization in the input values increase the network training speed. The similar approach can be applied for each layer of the network instead of only input layer. Batch normalization is applied by subtracting the batch mean from the output of previous layer and dividing the resultant by batch standard deviation. In practice, this results in allowing higher learning rate to speed up the training since there is no activation value of a neuron that can go very high or very low due to large learning rate. In addition, it reduces the overfitting, which can be a reason of considering batch normalization as a regularization method, by introducing some noise to the activation value of each neuron in hidden layers.

Dropout

Dropout is a very simple and effective regularization method and considered as a type of layer. This can simply be interpreted as randomly *killing* each neuron with probability p in dropout layers during training. In other words, a randomly sampled sub-network is held responsible for the objective of the whole network and only their connections are updated during backpropagation. An illustration of using dropout can be found in Figure 3.15

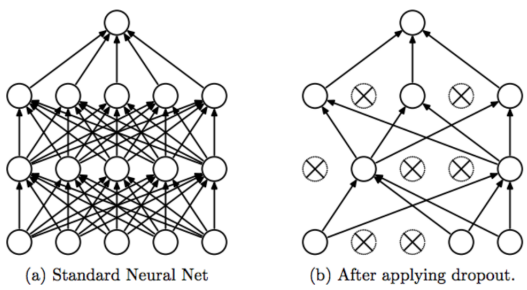


Figure 3.15: An illustration of dropout regularization

During testing, no dropout is used by setting dropout ratio $p = 0$. This means that during testing, all of the subnetworks are averaged together to achieve the final objective of the network. That is why dropout gives improved performance. In most of the cases, $p = 0.5$ is found to be the most effective dropout ratio.

3.2.6 Transfer learning

The state-of-the-art deep networks trained by famous research groups are trained using extensive amount of data, such as ImageNet [3]. As already shown in Figure 3.12, the performance of the deep networks increase with the amount of training data. Thanks to the generalization property of these networks, using pretrained networks as baseline allows the weights to reach the optimum value faster. This procedure is usually called as *fine-tuning the network*.

For some domains such as remote sensing applications, obtaining a huge training data as ImageNet data set is not very practical. In such cases, using pretrained networks, even trained in another domain, could save a lot of time with better performance because the training a deep network from scratch with a small data set results in a poor performance. It is a common sense that if one has a small data set on a similar domain of a pretrained network, only fine-tuning the top classifier layer could be sufficient. As the number of training data increases, it is always better to fine-tune more layers starting from the top classifier layer because deep networks extract global information in the first layers and more detailed and domain-specific information in the deeper layers. However, if one needs to train a deep network for a very different data set, whole parameters of each layer in a pretrained network should be fine-tuned, which requires a lot of training data on the new domain. In case of having a small data set for a different domain, using pretrained networks might not be a good idea.

3.2.7 Ensemble multiple networks

Ensemble methods in machine learning is an approach to combine several different trained models to achieve state-of-the-art accuracy. As already known in the literature, ensemble methods gives more accurate results when compared to single model.

For example, one way of ensemble method is to train the network with different random initialization. Another way would be choosing top several models of cross-validation while determining the best hyperparameters. In addition, using different checkpoints of a single model training and combining them to obtain final classifier would be a good idea and very inexpensive to achieve.

3.3 Popular CNN Architectures

ImageNet is a very large visual database created for visual object recognition. ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual contest where algorithms compete to achieve the most accurate scene classification task. Most of the state-of-the-art deep networks for semantic image segmentation and object detection problems use the networks trained with ImageNet data set as baseline since ImageNet database is huge enough to learn global filters hierarchically. Similarly, PASCAL Visual Object Classes (VOC) data set is a data set used for object detection applications and annual contest is held to detect objects in an image scene. In this section, several important deep networks used in image scene classification, semantic image segmentation and object detection tasks are introduced with their novel approaches.

3.3.1 LeNet-5

LeNet-5 is introduced by LeCun et al. [99] in 1998 to detect digits in bank checks. The network takes 32x32 size gray-scale input and consecutively convolves with 5x5 filters followed by 2x2 pooling windows with stride 2, and 2 fully connected layer until the output. The architecture can be summarized as [6c-2s(a)-16c-2s(a)-120c-fc84-out10 (5x5)] where c represents convolution operation and the number before it is the number of activation maps, s stands for the subsampling (pooling) operation with window size and stride 2, (a) indicates the average pooling, fc is the fully connected layer and the number after it is the number of activation neurons, and out10 is the output layer with 10 neurons. Visualization of the network is given in Figure 3.16.

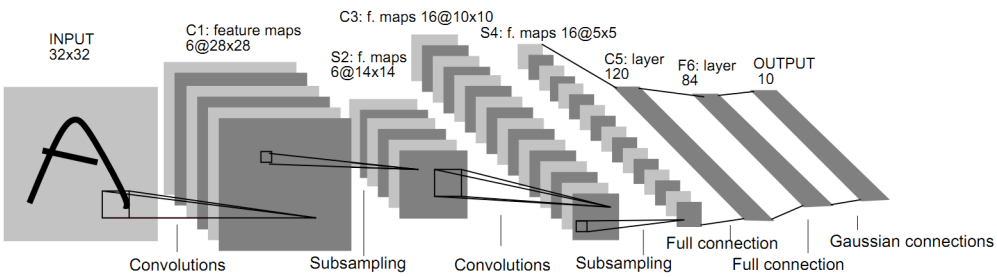


Figure 3.16: LeNet-5 Network introduced by LeCun et al. in 1998 to recognize digits in bank checks

This network can be considered as a relatively shallow convolutional network when compared to other state-of-the-art deep networks, however still performs adequate to recognize hand-written digits.

LeNet-5 is the pioneer of convolutional neural networks. However, deeper neural networks were not practical due to the reasons such as hardware restrictions and lack of huge amount of training data in those years.

3.3.2 AlexNet

AlexNet is an 8 layer convolutional neural network introduced by Alex Krizhevsky et al. [2] in ILSVRC'12 and achieved a breakthrough in the image scene classification challenge. Having a similar structure as LeNet, AlexNet is deeper with more filters per layer. This network is trained by ImageNet data set, therefore designed to receive 224x224x3 input data. The architecture can be summarized as [96c(11x11)-3s(m)-lrn-256c-3s(m)-lrn-384c-384c-256c-3s(m) (3x3)-fc4096-d-fc4096-d-out1000] where *bn* and *d* stand for local response normalization (which is not common anymore) and dropout layers, respectively. The network was trained on two different GPUs, that is why the structure in Figure 3.17 is split into 2 parts.

Another novelty of this network is that it was the first time of ReLU activation function. In order to improve the accuracy, heavy data augmentation, dropout, L2 regularization and ensemble methods were applied.

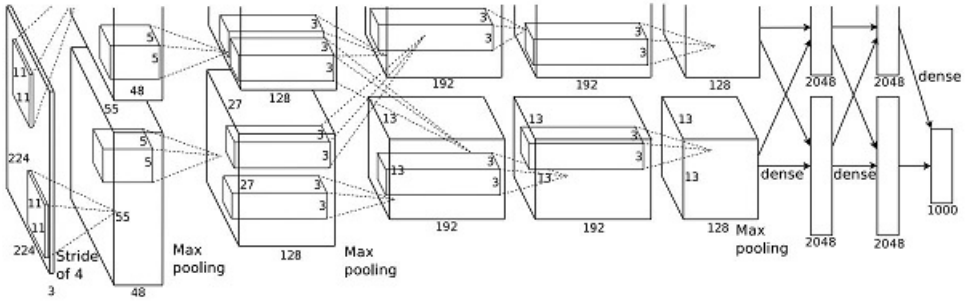


Figure 3.17: AlexNet architecture made breakthrough in ILSVRC competition using deep convolutional network first time in 2012

3.3.3 VGGNet

VGGNet proposed by Zisserman et al. [65] in ILSVRC'14 achieved a more accurate performance, second in classification and first in localization, by using smaller filters with deeper architecture when compered to AlexNet. VGGNet has 2 different versions such as VGG16 and VGG19, where VGG16 has 12 convolutional layers and 3 fully connected layers and VGG19 has 3 more convolutional layers. VGG16 architecture is given in Figure 3.18.

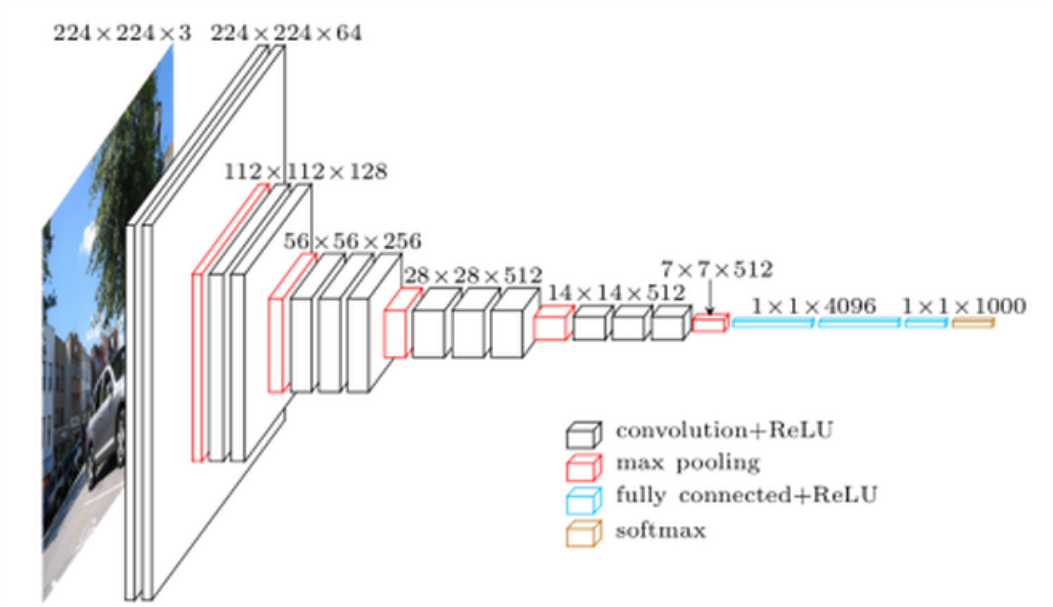


Figure 3.18: VGG16 architecture achieved better performance in ILSVRC'14 with constant 3x3 filters and deeper structure

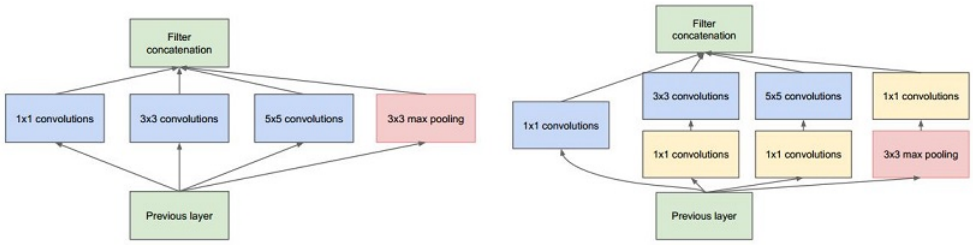
VGGNet uses 3x3 filters for all convolutional layers with zero-padding to preserve the size of the activation maps during convolution. Moreover, VGGNet uses three adjacent 3x3 filters which results in having a 7x7 effective receptive field. Having a deeper structure allows the network to learn more non-linearity; therefore, the last fully connected layer with 4096 neurons generalize well for other tasks. The architecture could be summarized as [64c-64c-2s(m)-128c-128c-2s(m)-256c-256c-256c-2s(m)-512c-512c-512c-2s(m)512c-512c-512c-2s(m) (3x3)-fc4096-d-fc4096-d-out1000].

3.3.4 GoogLeNet

This architecture is proposed by Szegedy et al. [66] in 2014 and outperformed all other competitors in ILSVRC'14 classification category with a performance very close to human level. By using efficient "Inception" module, GoogLeNet achieved the best of its time in image scene classification by only 5 million weights, which is 28x smaller than VGGNet and 12x smaller than AlexNet.

Inception module

The novelty introduced with GoogLeNet is the "Inception module", which can be considered as network inside a network. The module architecture is given in Figure 3.19.



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 3.19: Inception module led GoogLeNet to become the most accurate in ILSVRC'14 classification task

Naive Inception module has parallel 3 different convolutional operations with filter size 1x1, 3x3, 5x5 and 1 pooling operation with window size 3x3 that are followed

by a depth-wise filter concatenation. However this structure is very expensive in terms of computational complexity. Hence, a solution is proposed using 1x1 "bottleneck" convolutional layers by reducing the activation volume of the previous layer. This approach reduced the complexity of the inception module more than a half, which made stacking inception modules more practical.

GoogLeNet architecture is given in Figure 3.20. The structure begins with usual convolution, pooling, 2 consecutive convolutions, pooling, and continues with stacked inception modules with dimension reduction. The output layer at the deepest gives the classification result. Other two output layers inside the network are auxiliary classification outputs to inject additional gradients to the lower layers of the structure.

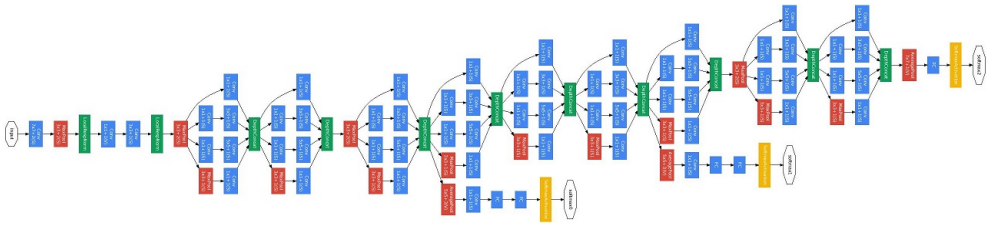


Figure 3.20: GoogLeNet - the classification winner architecture in ILSVRC'14

3.3.5 ResNet

In 2015, He et al. [67] introduced Residual block as in Figure 3.21 which helped train ultra deep network with 152 layers effectively and placed the first rank for all classification, detection, and localization competitions in ILSVRC'15 and COCO'15 beating the human level performance.

The idea comes from the hypothesis that deeper networks should perform better than shallower ones. However, stacking convolutional layers on top of each other gives worse performance because the optimization becomes more difficult as the network gets deeper. So, residual blocks solves this problem by opening a "highway" for the gradients coming from the top layer to reach the first layers. Having similar approach with VGGNet such as using always 3x3 convolutional filters except the first 7x7 convolutional layer followed by a pooling layer and doubling the number of filters as reducing the size of the activation maps with not pooling but having the convolution layer with stride 2, ResNet is much deeper than VGGNet while having

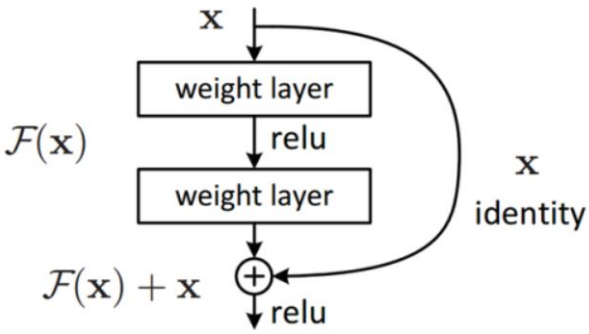


Figure 3.21: Residual block of ResNet152 that placed first rank in all competition categories beating human performance in ILSVRC'15 and COCO'15

less computational complexity and requiring less amount of memory. A 50 layer version of ResNet is shown in Figure 3.22.

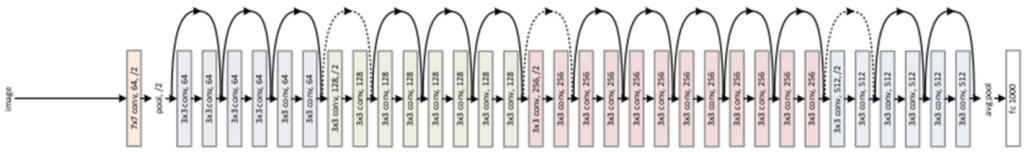


Figure 3.22: ResNet50 - 50 layer CNN introduced by He et al. in 2015

ResNet does not contain any fully connected layer except the output layer and implements heavy batch normalization after every convolutional layer, Xavier initialization, L2 regularization and no dropout.

3.3.6 FCN8s

Fully connected layers are defined with a constant input size. However, semantic image segmentation requires pixel-level classification; hence, the network should be capable of handling varying input size. In 2014, Long et al. [114] replaced fully connected layers with 1x1 convolutional layers, allowing the network to receive different input sizes and called the network Fully Convolutional Network as given in Figure 3.23.

The architecture uses VGGNet as baseline. Since VGGNet has 5 pooling layers, it

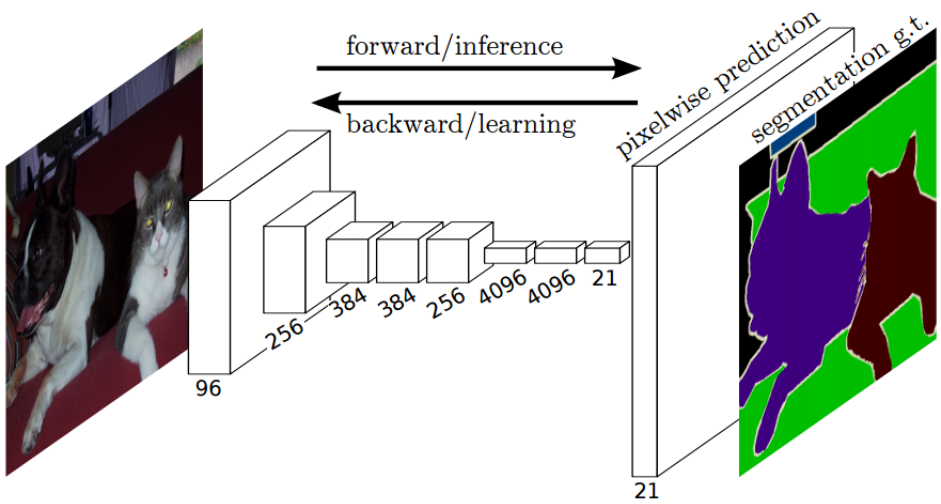


Figure 3.23: Fully Convolutional Network used for semantic image segmentation (FCN32s

divides the input size by $2^5 = 32$. This output map of VGGNet is deconvolved, i.e., upsampled by 32 to obtain the original input size. This network is called as FCN32s. However, the result was very coarse due to information loss caused by pooling layers. The solution was found out to be making an upsampling by 2 and adding the information from the previous pooling activations to this upsampled map. To obtain the original input size, this activation maps should be upsampled by 16, which led the name of the network FCN16s. Obviously, the results of FCN16s is finer than FCN32 by 3% in mAP. Taking one step more, VGGNet activation map is upsampled by 2 two consecutive times by each being added with 4th and 3rd pooling activations, and at the and upsampled by 8 to obtain the original size as in Figure 3.24, leading FCN8s network the finest of all by 0.5% more in mAP.

Since utilizing the higher resolution feature maps of previous pool2 and pool1 layers did not yield an important improvement in the performance, the authors only used pool3 and pool4 feature maps to increase the accuracy

3.3.7 YOLO

You Look Only Once (YOLO) network is proposed by Redmon et al. [90] in 2015 to detect PASCAL VOC objects in images and one of the fastest networks for object

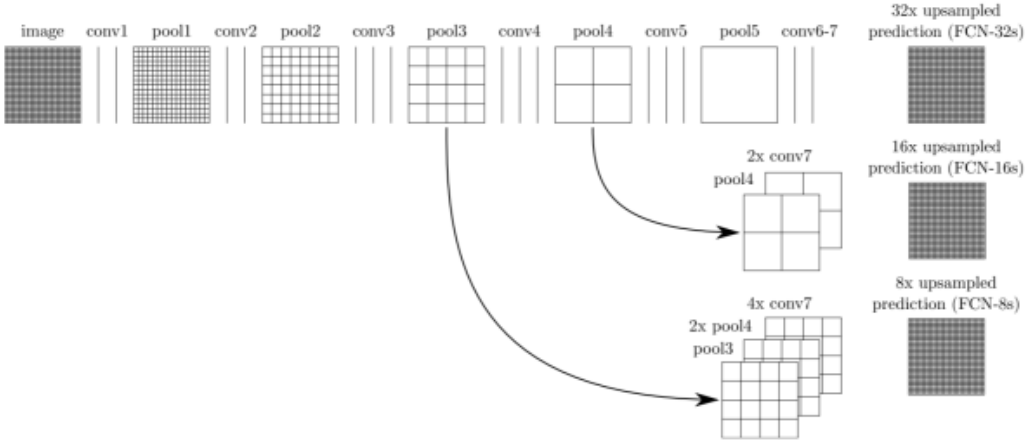


Figure 3.24: Detailed look at FCN8s with deconvolution layers

detection task. The architecture is inspired from GoogLeNet and given in Figure 3.25.

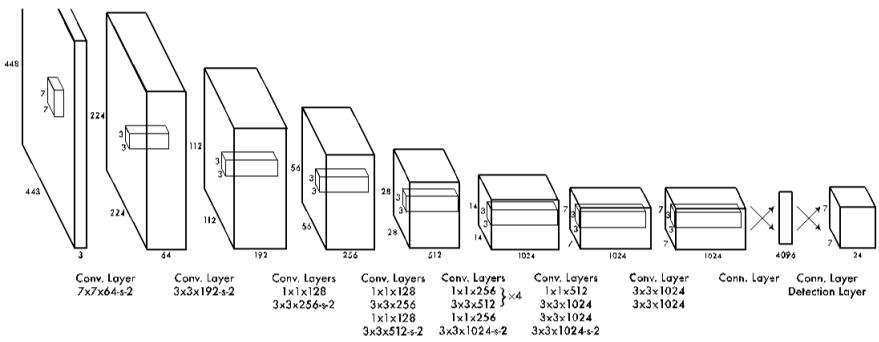


Figure 3.25: YOLO architecture proposed to detect PASCAL VOC objects

YOLO is a assumed to be a regressor rather than a classifier since it predicts the bounding box parameters with regression. The output volume is SxSx$(B$x$2 + C)$ where $S = \frac{inputsize}{2^6}$, B is the bounding boxes and C is the number of classes. The network divides the input image into SxS grids and predicts center point x and y, height h, and width w of each bounding box along with class probabilities. For example, YOLO network for 20 class COCO data set requires an output filter SxSx$(5$x$2 + 20)$. Here the network is designed to predict 2 bounding boxes for each cell. One of the bounding box predictor giving higher IoU is held 'responsible' for prediction, i.e. the loss is calculated using the responsible predictor. Sum-squared error is used to optimize the weights and the loss function is defined as the weighted sum of bounding box parameters and class predictions error. The final loss function of YOLO is presented in Equation 3.13

$$
\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \big((x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2\big) \\
&+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \big((\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2\big) \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
&+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
&+ \sum_{i=0}^{S^2} 1_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
\tag{3.13}
$$

where 1_i^{obj} denotes if object appears in cell i of output map and 1_{ij}^{obj} denotes that jth bounding box is responsible for that object prediction. x_i and w_i parameters in the loss function are normalized with respect to width of the input size and y_i and h_i parameters are normalized with respect to the height of the input size to make the terms in the loss function consistent since both confidence and class probabilities are in the interval of $[0, 1]$.

There might be two or more cells making prediction about the same object in the output map. In this case, a non-max suppression algorithm is used to get rid of all predictions except the one having the higher $P(obj)$.

The drawbacks of YOLO are that each cell can predict only 2 bounding boxes for only one class, small objects near each other cannot be detected, the predictions are computed on a relatively coarse features and the network cannot generalize due to learning the bounding boxes directly from the training set. Therefore, several improvements on these issues are discussed in [93] and named the improved network as YOLO9000 since it can detect more than 9000 classes.

YOLO9000

YOLOv2, sometimes called as YOLO9000, is proposed by Redmon and Farhadi [93] to improve the aforementioned limitations of original YOLO. In this network the

regularization is applied using batch normalization instead of dropout and 2 percent improvement is achieved. For YOLOv2, the network is pretrained for classification task using ImageNet with the same resolution of detection task, where in the previous version the classification network was pretrained using the half resolution of detection network. Full size classification training increased the mean average precision (mAP) by 4%. In addition, newer version replaced fully connected layer with convolutional layer, enabling to use anchor boxes. Unlike the previous version where only 2 bounding boxes are directly predicted, using anchor boxes in YOLOv2 allows the network to make thousands of predictions by output layer size SxSxBx$(5+C)$ where B is the number of anchor boxes, 5 stands for offset coordinates with respect to the grid center and the confidence, and C is the number of classes.

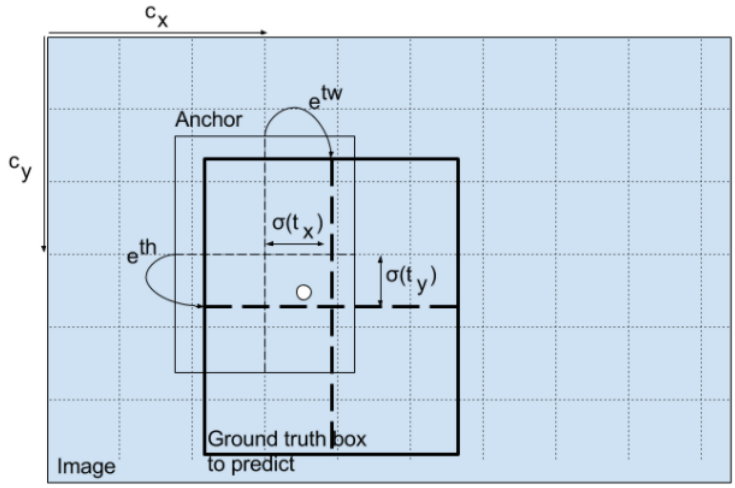


Figure 3.26: Anchor Boxes allow multiple detection in a single cell by predicting offset parameters from the center of 'responsible' cell

This approach made a slight decrease in localization; however, significantly improved the recall rate. Moreover, YOLOv2 makes use of fine-grained features with 13x13 feature map by a pass-through layer. Finer resolution features of 26x26x512 are converted to 13x13x2048 feature volume, which can be concatenated with features coming after the pooling. The previous version did not utilize the finer feature map fusion and had output resolution 7x7. Hence, fine-grained features with better resolution help the network localize smaller object better.

CHAPTER 4

LAND USE ANALYSIS USING CNN IN REMOTE SENSING

Accessing the remote sensing images have become widely available by the help of commercial satellite systems, such as WorldView, GeoEye, IKONOS etc., exposing a need for deep analysis and interpretation for a range of applications, such as mapping, urban planning, resource management, climate change observations and land use monitoring. These satellite images can be acquired using panchromatic or multispectral sensors on the satellite. Panchromatic images contain only a single band; however, their resolution can be impressive. On the other hand, multispectral images may contain various bands in electromagnetic spectrum such as Red, Green, Blue and Near-Infrared.

Land use analysis using satellite images is a challenging but necessary problem. Natural areas such as bare land, cultivated field, forest, lake, rocks, sea, shorelines etc. needs to be analyzed in remote sensing images. Moreover, dealing with the increasing number of remote sensing images requires to automatize land use analysis. For this purpose, a series of CNN-based deep learning experiments for image scene classification and semantic image segmentation tasks are discussed in this chapter.

4.1 Image Scene Classification for Land Use Analysis

In this section, a CNN-based land use classifier is proposed and training of this CNN-based image scene classification network are examined in detail.

4.1.1 Proposed Classification Network

Deep neural networks, especially convolutional neural networks, are proven to perform more accurate than traditional methods in image scene classification of remote sensing applications. Therefore, extensive number of training CNN versions, such as LeNet-5, AlexNet, and VGGNet are conducted using various set of hyperparameters on different datasets that are shared by other researchers, such as SAT-4, SAT-6 [83] and finally, self-generated 'Ekilialan' dataset for cultivated land classification. Basic tested architectures of the proposed classification networks are presented in Table 4.1.

Table 4.1: Tested architectures for the image scene classification network. The characters 'c', 's', 'fc', and 'out' stand for convolution, subsampling (pooling), fully connected and output layer, respectively. The last 'out' layer is also a fully connected layer, but with softmax activation to compute class probabilities for each. Input patch has the size 28x28 with 4 bands, whereas the output labels are equal to 4, 6 and 2 for SAT-4, SAT-6, and 'EkiliAlan' datasets, respectively

Input	**Proposed CNN structures**	**Output (# of labels)**
Image Patch (28x28x4)	c1-s1-c2-s2-fc3-out4	Class Label (4,6,2)
Image Patch (28x28x4)	c1-s1-c2-s2-fc3-fc4-out5	Class Label (4,6,2)
Image Patch (28x28x4)	LeNet-5	Class Label (4,6,2)
Image Patch (28x28x4)	c1-s1-c2-s2-c3-s3-fc4-out5	Class Label (4,6,2)
Image Patch (28x28x4)	AlexNet	Class Label (4,6,2)
Image Patch (28x28x4)	c1-c2-s1-c3-c4-s2-c5-c6-c7-out8	Class Label (4,6,2)
Image Patch (28x28x4)	VGG16	Class Label (4,6,2)

4.1.2 Experimental Results

4.1.2.1 Datasets for Classification Network Training

Three different datasets are used during simulations. SAT-4 and SAT-6 datasets are extracted from National Agriculture Imagery Program (NAIP) dataset in [83] and available to public as *.mat* file. *'Ekilialan'* dataset is generated by the author by using WorldView-2 and GeoEye-1 satellite raw raster image data.

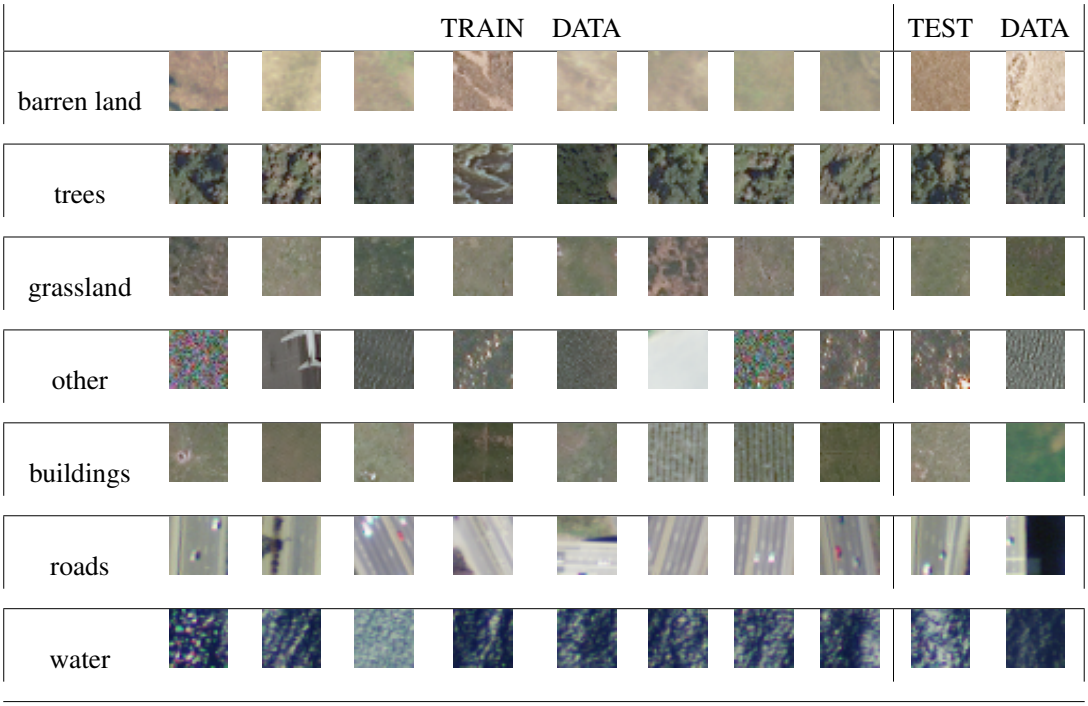


Figure 4.1: SAT-4 and SAT-6 datasets sample patches of size 28x28x4 for each class - *barren land, trees, grassland, other* for SAT-4, *barren land, trees, grassland, buildings, roads, water* for SAT-6

SAT-4

This dataset contains 400,000 training and 100,000 testing image patches of size 28x28 with RGB-NIR bands. Four classes of the dataset are 'barren land', 'trees', 'grassland' and 'other'. Ground-truth information is also provided as one-hot vector (i.e. the correct label is equal to 1, whereas the others are 0 for a 4-D vector) for each image patch.

SAT-6

Another dataset introduced along with SAT-4 is SAT-6 dataset of classes 'barren-land', 'trees', 'grassland', 'roads', 'buildings' and 'water'. SAT-6 consists of 324,000 image patches for training and 81,000 for testing purpose. As SAT-4, the image patches of this dataset are 28x28x4 and ground-truth information is provided as one-hot vector for each image patch.

Ekilialan

This dataset is generated using very large 4-band raster images of WorldView-2 and GeoEye-1 satellites. The images of *.img* format also contains hand-crafted ground-truth shape files as *.shp* format. In general, each image has around 20,000x20,000x4 pixels of 11-bit value and 0.46 meter pixel resolution in each channel. First, the pixel values are normalized to 8-bit to imitate SAT-4 and SAT-6 datasets. Then, 28x28x4 size image patches covering at least 70 percent of the specified class ground-truth are extracted along with one-hot label vector for each patch. Ekilialan dataset is a binary dataset with two classes, 'cultivated field' and 'other'. The class 'other' contains image patches of several classes such as sea, forest, building, road, aircraft etc. This dataset contains 628,039 training and 60,105 testing image patches of size 28x28x4. Half of the image patches belong to 'cultivated field' label whereas the remaining half contains 'other' class.

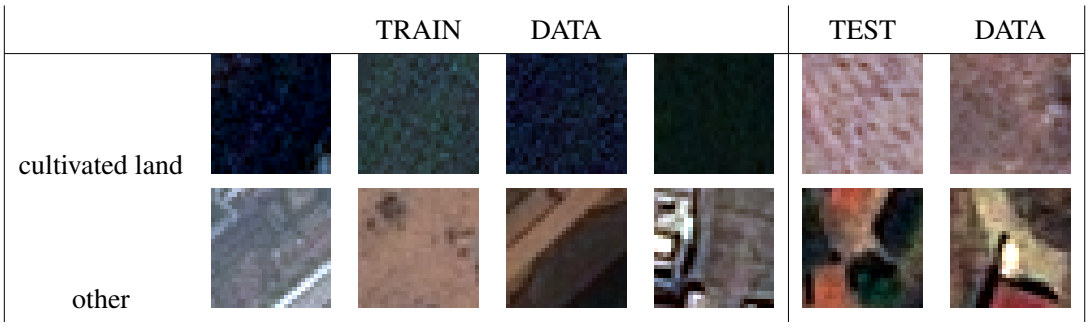


Figure 4.2: Generated Ekilialan dataset sample patches of size 28x28x4 for each class - *cultivated land, other*

4.1.2.2 Experimental Setup

Proposed CNNs were trained with various hyperparameter sets to reach the optimum results in remote sensing image classification application. The experiments were all conducted on a 6 GB GPU of nVidia GTX 1060 using tensorflow [115] deep learning framework. The effects of batch size, learning rate, dropout, number of fully connected (fc) layer, convolutional filter size, depth of network, pooling type, and CNN architecture are examined on the accuracy. In all experiments, the loss function is selected as cross-entropy with 0.9 momentum optimizer ratio. All the experiments are

tested in a controlled manner, giving a unique index to every particular experiment, denoted by $N\#$ where $\#$ denote the index of the architecture in that particular experiment for all the tables below. Only some of the experiments with prominent results are presented in this thesis.

Experiments on Batch size and learning rate

N35 and N37 networks that are described in Table 4.2 were trained to determine the effect of selecting batch size and learning rate.

Table 4.2: Proposed network parameters for batch size and learning rate

Exp#	Proposed CNN structure	Batch size	Learning rate
N35-	c6(5x5)-a(2,2)-c12(5x5)-m(2,2)-fc96-out4	5000	exp(.0002, .0001)
N37-	c6(5x5)-a(2,2)-c12(5x5)-m(2,2)-fc96-out4	2000	exp(.0005, .0001)

Both networks are trained using SAT-4 dataset, ReLU as activation function, no padding, and dropout ratio as 0.5 during training and 1.0 during testing. Initial value of exponentially decreasing learning rate is set to be equal to one over batch size.

Experiments on Dropout

N36, N40, N41 and N42 networks that are described in Table 4.3 were trained to determine the effect of dropout.

Table 4.3: Proposed network parameters for different dropout rates

Exp#	Proposed CNN structure	Dropout
N36-	c6(5x5)-m(2,2)-c12(5x5)-m(2,2)-fc128-out4	0.00
N40-	c6(5x5)-m(2,2)-c12(5x5)-m(2,2)-fc128-out4	0.25
N41-	c6(5x5)-m(2,2)-c12(5x5)-m(2,2)-fc128-out4	0.50
N42-	c6(5x5)-m(2,2)-c12(5x5)-m(2,2)-fc128-out4	0.75

All networks are trained using SAT-4 dataset, ReLU as activation function, no padding, batch size 2000 and learning rate is $lr = 0.0005e^{-0.0001i}$.

Experiments on Number of fc layer

Networks N68 and N69 in Table 4.4 were trained to determine the effect of using multiple fc layers before output layer.

Table 4.4: Proposed network parameters for additional fully connected layer effect

Exp#	Proposed CNN structure
N68-	c6(5x5)-a(2,2)-c12(5x5)-m(2,2)-fc48-out4
N69-	c6(5x5)-a(2,2)-c12(5x5)-m(2,2)-fc48-fc12-out4

Both of the proposed networks are trained using Ekilialan dataset with mean extracted, ReLU as activation function, no padding, 0.5 as dropout ratio, batch size 1000 and learning rate is $lr = 0.001e^{-0.0001i}$.

Experiments on Extracting Mean of dataset

Preprocessing is sometimes useful for deep neural networks to be trained to optimum. Extracting mean of the dataset is one of several preprocessing steps applied in training of deep networks because preprocessing helps network become more robust to bias between the mean values of the images and zero-mean randomly initialized weights. Hence, networks given in Table 4.5 are tested with and without mean-extracted (M/E) preprocessing.

Table 4.5: Proposed network for mean-extract dataset experiment

Exp#	Proposed CNN structure
N74-	c6(5x5)-a(2,2)-c12(5x5)-m(2,2)-fc48-out2
N75-	c6(5x5)-a(2,2)-c12(5x5)-m(2,2)-fc48-out2

Experiments on Convolutional filter size and stride

N78, N79, and N80 networks given in Table 4.6 were trained to determine the effect of convolutional filter stride and size.

All of the proposed networks are trained by Ekilialan dataset which is mean extracted, ReLU as activation function, no padding, 0.5 as dropout ratio, batch size 2000 and learning rate is $lr = 0.0005e^{-0.0001i}$.

Table 4.6: Proposed network parameters for different filter size and stride

Exp#	Proposed CNN structure	First layer conv. stride
N78-	c6(5x5)-a(2,2)-c16(5x5)-m(2,2)-fc64-out2	(1,1)
N79-	c6(5x5)-a(2,2)-c16(5x5)-m(2,2)-fc16-out2	(2,2)
N80-	c6(3x3)-a(2,2)-c16(4x4)-m(2,2)-fc100-out2	(1,1)

Experiments on Depth of network

N89, N90, N91 and N111 networks of Table 4.7 were trained to determine the effect of adding more layers to the network.

Table 4.7: Proposed network parameters for different depth of networks

Exp#	Proposed CNN structure
N89-	c8(3x3)-a(2,2)-c12(4x4)-m(2,2)-fc100-out2
N91-	c8(3x3)-a(2,2)-c12(4x4)-m(2,2)-c24(4x4)-fc100-out2
N90-	c8(3x3)-a(2,2)-c12(4x4)-m(2,2)-c24(4x4)-m(2,2)-fc100-out2
N92-	VGGNet16 (fully convolutional fc1000-fc1000)-out2

All of the proposed networks except VGGNet are trained using Ekilialan dataset with mean extracted, ReLU as activation function, no padding, 0.5 as dropout ratio, batch size 2000 and learning rate is $lr = 0.0005e^{-0.0001i}$. VGGNet is also trained using the same hyperparameters only with 256 batch size due to memory concern.

Experiments on Pooling

N96, and N97 networks of Table 4.8 were trained to determine the effect of pooling type.

Table 4.8: Proposed network parameters for different pooling types

Exp#	Proposed CNN structure
N96-	c8-c16 (5x5)-a(2,2)-c64-c256 (3x3)-a(2,2)-c1024(3x3)-fc100-out2
N97-	c8-c16 (5x5)-m(2,2)-c64-c256 (3x3)-m(2,2)-c1024(3x3)-fc100-out2

Both of the proposed networks are trained using Ekilialan dataset with mean extracted, ReLU as activation function, no padding, 0.5 as dropout ratio, batch size 500 and learning rate is $lr = 0.0005e^{-0.0001i}$.

Experiments on CNN architecture

N107, and N108 networks of Table 4.9 were trained to determine the effect of using convolutional layers instead of fully connected layers.

Table 4.9: Proposed network parameters on fully convolutional layers instead of fully connected layers

Exp#	Proposed CNN structure
N107-	c16-c64(3x3)-m(2,2)-c128-c128(3x3)-m(2,2)-c256(3x3)-fc512-fc512-out2
N108-	c16-c64(3x3)-m(2,2)-c128-c128(3x3)-m(2,2)-c256(3x3)-c512(2x2)-c512(1x1)-out2

Both of the proposed networks are trained using Ekilialan dataset, ReLU as activation function, no padding, 0.5 as dropout ratio, batch size 256, and learning rate is $lr = 0.004e^{-0.0001i}$.

4.1.2.3 Experimental Results

Training loss and validation loss along with test accuracy are the parameters to examine while determining the best network. In Table 4.10, the results of the proposed networks given in the previous section are compared. Moreover, number of total weights for convolutional layers and fully connected layers for each network is provided in the same table.

Selecting batch size as high as possible as long as hardware memory allows is the best suggestion. However, low batch size is sometimes is necessary because of insufficient memory on the hardware. For this case, setting the initial value of learning rate equal to one over batch size for classification networks is an empirical but a meaningful decision. When comparing N35 and N37 in terms of batch size and learning rate, the results are quite close to each other. The reason why N35 has slightly worse performance is the result of having a learning rate lower than the optimum value.

Selecting the optimum dropout value is turned out to be crucial. N36, N40, N41, and N42 results show that the worst accuracy is observed without using a dropout layer. The most accurate result is achieved with 0.25 dropout rate, which is followed by the network with dropout ratio 0.5 with a slight difference. The conclusion that can be

Table 4.10: The results of proposed classification networks - M/E: mean extracted

Exp#	**Dataset**	W_{conv}	W_{FC}	**Tr. loss**	**Val. loss**	**Acc.**
N35-	SAT-4	450	18,816	0.1621	0.1689	0.944
N37-	SAT-4	450	18,816	0.1308	0.1335	0.952
N36-	SAT-4	450	18,816	0.3250	0.3283	0.880
N40-	SAT-4	450	18,816	0.1210	0.1258	0.963
N41-	SAT-4	450	18,816	0.1574	0.1575	0.954
N42-	SAT-4	450	18,816	0.2114	0.2120	0.933
N68-	SAT-4	450	9,408	0.1186	0.4754	0.835
N69-	SAT-4	450	9,840	0.2241	0.5586	0.777
N74-	Ekilialan	450	9,312	0.0922	0.4579	0.848
N75-	Ekilialan M/E	450	9,312	0.1155	0.5466	0.831
N78-	Ekilialan M/E	550	16,512	0.0869	0.6176	0.832
N79-	Ekilialan M/E	550	288	0.2131	0.7037	0.715
N80-	Ekilialan M/E	310	40,200	0.1615	0.3024	0.881
N89-	Ekilialan M/E	264	30,200	0.1045	0.5483	0.794
N91-	Ekilialan M/E	648	9,800	0.1114	0.5336	0.777
N90-	Ekilialan M/E	648	2,600	0.1402	0.4762	0.821
N92-	Ekilialan M/E	40,066	-	0.0801	0.4132	0.892
N96-	Ekilialan M/E	12,696	102,600	0.0942	0.6651	0.771
N97-	Ekilialan M/E	12,696	102,600	0.0758	0.3643	0.882
N107-	Ekilialan	5,328	787,454	0.0246	0.2695	0.918
N108-	Ekilialan	7,888	1,024	0.0219	0.4819	0.858

deduced from these result is that the network could perform better with other values for dropout ratio instead of setting dropout rate equal to 0.5 as default.

Network N68 has one fully connected layer before output layer, whereas N69 has one more fully connected layer, making an addition about 5% in the number of weights. Since adding one more fully connected layer before the output layer increases the number of parameters of the network to be trained as much as whole number of parameters in the first 2 convolutional layers, N69 performed worse when trained with the same amount of training data. Therefore, adding more layers as fully connected do not always increase the performance unless the amount of training data is also increased.

Networks N74 and N75 are the same networks, except a mean-extract preprocessing. N75 is trained with mean-extracted Ekilialan dataset and has lower accuracy when

compared to N74 where the network is trained without mean-extract preprocessing. The results show that mean-extraction by itself cannot help network weights to converge the optimum values better.

Keeping the network structure similar, changing the first convolution layer stride from 1 in N78 to 2 in N79 reduces the feature map size, leading less number of features to reach the fully connected layers. Therefore, number of neurons in the fully connected layers were arranged according to the number of neurons after second pooling layer. This change also decreases the number of trainable parameters of the whole network significantly. Hence, small number of weights with coarse features resulted in a poor performance in N79, dropping accuracy from 83.2% to 71.5%. On the other hand, decreasing the first and second convolutional filter size to 3x3 and 4x4, respectively, boosted the performance of the network N80 to 88.1%. It gives a strong indication that smaller-sized convolutional filters typically learn better even though those networks have quite number of parameters as fully connected.

When comparing the results of N89 and N90, increasing depth of the network with convolutional and pooling layers boosted the accuracy by 3% since pooling layer helped extract more semantic information. On the other hand, result of N91 shows that increasing the depth with convolutional layer but not adding a pooling layer resulted worse. Yet, VGGNet in N92 increased the accuracy by 10% from baseline LeNet-5. These results indicate that number of trainable parameters regardless of network architecture is not an indicator to foresee whether a specific amount of weight is optimum, less or too extensive.

N96 network was constructed with average pooling where N97 with max pooling. After training both networks, N97 with max pooling performed nearly 11% better than N96 with average pooling. Therefore, max pooling should be preferred as priority option.

Two networks inspired by AlexNet with consecutive convolutional layers were trained. In N108, fully connected layers of N107 are replaced with convolutional layers. Although the training loss value of N108 is better than N107 because it has significantly less number of trainable parameters in the network, it has poorer generalization ability, i.e. it was less accurate for the unobserved test data.

The best performance is usually obtained by using some hyperparameter searching methods, such as grid search, random search, or Bayesian search. In this study, manual search similar to grid search was applied to gain intuition about neural network training.

The most accurate CNN architectures of best hyperparameter set and the accuracy values are given in Table 4.11 where all of the networks were trained using ReLU as activation function, no padding, 0.5 as dropout ratio, batch size 256, and learning rate is $lr = 0.004e^{-0.0001i}$. The results show that SAT-4 and SAT-6 datasets were almost perfectly classified with 16-layer VGGNet, initialized from ImageNet pretrained weights, while for ekilialan dataset the best result was obtained with a shallower network similar to AlexNet.

Table 4.11: Most accurate proposed classification network architectures and results

Exp#	Dataset	Proposed CNN structure	W_{conv}	W_{FC}	Tr. Loss	Val. Loss	Acc.
N107-	Ekilialan	c16-c64(3x3)-m(2,2)-c128-c128(3x3)-m(2,2)-c256(3x3)-fc512-fc512-out2	5,328	787,454	0.0246	0.2695	0.918
N108-	SAT-4	VGG16 with ImageNet pretrained weights	40,068	-	0.0008	0.0046	0.999
N109-	SAT-6	VGG16 with ImageNet pretrained weights	40,070	-	0.0003	0.0012	0.999

It is worth to note that for Ekilialan dataset, the most accurate network was obtained with shallower network than VGG16, while VGG16 has performed extremely accurate in SAT-4 and SAT-6 datasets. Therefore, in order to improve Ekilialan dataset classification accuracy, pretrained SAT-6 weights are fine-tuned by Ekilialan dataset. Nevertheless, the outcome of the fine-tuned networks is not as accurate as the N107 in the test data. The results of Ekilialan proposed networks are given in Table 4.12.

Table 4.12: Ekilialan proposed networks and results

Exp#	Proposed CNN structure	W_{conv}	W_{FC}	Tr. Loss	Val. Loss	Acc.
N107-	c16-c64(3x3)-m(2,2)-c128-c128(3x3)-m(2,2)-c256(3x3)-fc512-fc512-out2	5,328	787,454	0.0246	0.2695	0.918
N110-	VGG16 with ImageNet pretrained weights	40,066	-	0.0134	0.6654	0.887
N111-	VGG16 with SAT-6 pretrained weights	40,066	-	0.0098	0.4985	0.901

In Table 4.12, it can be observed that even though deeper networks such as VGG16 could learn to classify training set with lower loss, those networks failed to generalize to classify unseen test data.

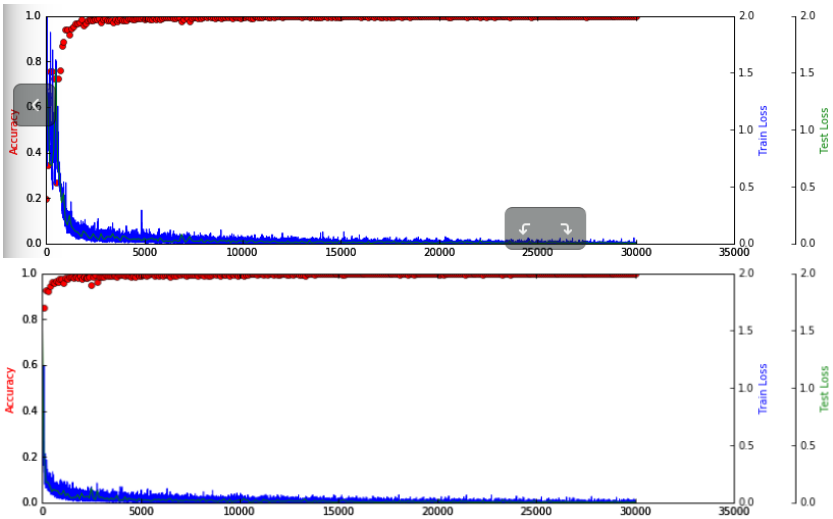


Figure 4.3: Network N110 for SAT-4 dataset on the top and N111 for SAT-6 dataset on the bottom with training loss, test loss and accuracy

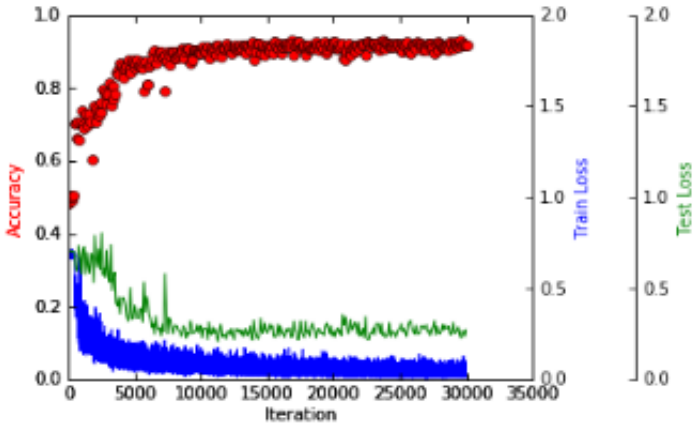


Figure 4.4: Network N107 for Ekilialan dataset - training loss, test loss and accuracy

4.2 Semantic Image Segmentation for Land Use

In this section, training a CNN-based semantic image segmentation method for land use analysis is examined in detail. On contrary to previous classification results, where the image patches (of size 28x28x4) are labelled as a whole into a number of classes, the results consist of output masks where every pixel has a segmentation label.

Table 4.13: Tested architectures for semantic image segmentation network

Input	Proposed CNN structures	Output (# of labels)
512x512x4	Reduced FCN8s (VGG10+upsampling)	10-class masks
512x512x4	Reduced FCN8s (VGG10 (5x5)+upsampling)	10-class masks
512x512x4	Modified FCN8s (VGG13 (5x5)+upsampling)	10-class masks
512x512x4	FCN8s (no skip connection)	10-class masks
512x512x4	FCN8s (VGG16+upsampling)	10-class masks

4.2.1 Proposed Segmentation Network

Following the advances in deep learning, CNNs are also utilized for semantic image segmentation. Hence, based on the idea of Long et al. [114], VGGNet as classification and deconvolutional layers to upsample the image into original size, several experiments were conducted for the natural scene segmentation in satellite images. The tested architectures for segmentation experiments are given in Table 4.13.

VGGNet is a classification network which contains fully connected layers. Since semantic segmentation task requires to perform a pixel-wise classification, the output of the segmentation should be a 2D classification map, unlike in original VGGNet, where the output is 1D class prediction vector. Therefore, VGGNet is modified to have a pixel-wise classification at the output by replacing fully connected layers with 1x1 size convolutional layers. Unlike fully connected layers, convolutional layers with 1x1 filters do not care about the input feature map, which leads to a pixel-wise classification. The benefit of using 1x1 convolutional layers instead of fully connected layers is that the input size of the network could be arbitrary, resulting in an output map size proportional to pooling layer and convolutional layer strides. Therefore, all the VGGNet versions trained in this study has 1x1 convolutional layers instead of fully connected layers.

Firstly, a shallower version of VGGNet is trained (referred as Reduced FCN8s VGG10 + upsampling). The reason of this selection is that remote sensing images with natural areas mostly contain textural information rather than edges or blob type forms. Since VGG16 is a network designed to classify 1,000 classes of ImageNet, 10 class pixel-wise classification network could perform accurate enough with less parameters.

Secondly, receptive field size is increased using 5x5 convolutional filters instead of default 3x3 size. The idea behind this preference is that larger receptive field could learn detailed texture information rather than edges or other shapes, since natural area segmentation images often contain three to five large portions of natural areas.

Thirdly, classification network is increased from 10 to 13 layer VGGNet with 5x5 filter size to check whether extending the network further boost the performance.

Later, VGG16 classification network is designed and ImageNet *pretrained weights* are imported to the network except the first convolutional layer filters due to dimension mismatch since pretrained weights are obtained with 3-band images while in this study 4-band images are used to train. First layer convolutional filters are initialized randomly with values from zero-mean normal distribution. Since RS images in this study contain 4 color bands (RGB-NIR), first convolutional layer weights are initialized randomly due to the dimension mismatch. Moreover, 3 consecutive deconvolution layers are added without the additional features coming from pool4 and pool3 feature maps. The pooling feature fusion is removed in this experiment to have less parameters to train, since the training set do not contain huge amount of data. Finally, full ImageNet pretrained FCN8s network in [114] is fine-tuned with seg2248 dataset.

4.2.2 Experimental Results

4.2.2.1 Dataset Generation for Segmentation Network

VGG16 network of FCN8s downscales the input size by 32 using 5 pooling layers, indicating the input size of the dataset multiple of 32. Therefore, the size of the images to generate the segmentation dataset is determined to be 512x512x4. Using the ground-truth information, 400 images of size 512x512x4 are extracted for 10 classes - 'sea', 'cultivated field', 'lake', 'rock', 'river', 'forest', 'shoreline', 'waterway', 'barren land', 'other' from the main satellite image data set. Each image is selected to include at least 3 classes. Data augmentation is a quite useful tool for such small size datasets to avoid overfitting while training a deep neural network. Hence, the a larger dataset is derived by using data augmentation techniques, such as rotating and flip-

ping the original images. Hence, *seg2248* dataset is obtained as 2,000 training images and 248 test images after data augmentation. Sample data from generated segmentation dataset is presented in Figure 4.5. It should be noted that dataset also contains images belonging to only a single class.

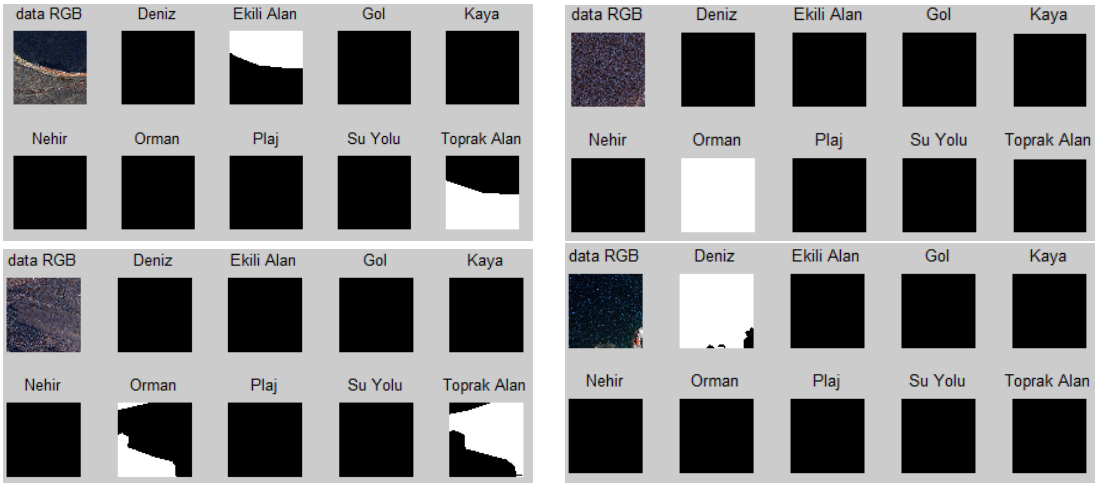


Figure 4.5: Sample data from generated segmentation dataset

4.2.2.2 Experimental Setup

FCN8s and its variants were trained for semantic image segmentation in satellite images using *Tensorflow* framework on a single GPU, nVidia GTX1060 (6GB). For full FCN8s implementations, batch size was selected as 2, where for the reduced and modified FCN8s versions the batch size was kept at 4. In addition, cross entropy loss function is used with stochastic gradient descend (SGD) optimization along with dropout ratio at 0.5 and learning rate 1E-6, constant throughout the training for all experiments.

The proposed networks given in Table 4.13 were trained with the given order for natural area segmentation in satellite images.

4.2.2.3 Results

Reduced FCN8s (VGG10+upsampling) with 3x3 filter size was trained from scratch using seg2248 dataset for 24 hours on 6 GB GTX 1060 GPU. Table 4.14 presents the

precision and recall rates for each class. Precision and recall indicators are calculated as in Equation 4.1.

$$Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN} \tag{4.1}$$

where TP, TN, and FN stand for True positive, True Negative, and False Negative, respectively.

Table 4.14: Precision and Recall values of seg2248 dataset trained on reduced FCN8s (VGG10) network from scratch

Class	**Sea**	**Cultivated Land**	**Lake**	**Rock**	**River**	**Forest**	**Shoreline**	**Waterway**	**Barren Land**	**Other**	**Average**
Recall	0.858	0.233	0.803	0.701	0.726	0.721	0.779	0.431	0.581	0.815	**0.665**
Precision	0.821	0.583	0.813	0.716	0.902	0.775	0.771	0.967	0.703	0.681	**0.773**

The results show that cultivated land recall and precision rates are the worst among the other classes, even though very similar classes such as 'sea' and 'lake' have much higher accuracy.

Another FCN8s network with VGG10 is trained using larger convolutional filters (5x5) instead of 3x3 filters to have larger receptive field because the images in the natural area segmentation problem usually contains smooth shapes with about three or four large portions for each class in a scene. The results given in Table 4.15 shows that larger receptive field size helped the network segment 'cultivated land' better and have higher recall for other classes as well, except 'lake', 'rock' and 'other'. However, while the average recall rate is increased by 10%, classes 'waterway' and 'barren land' had lower precision, resulting in average precision is reduced by 3%.

Table 4.15: Precision and Recall values of seg2248 dataset trained on modified FCN8s (VGG10 with 5x5 convolutional filters) network from scratch

Class	**Sea**	**Cultivated Land**	**Lake**	**Rock**	**River**	**Forest**	**Shoreline**	**Waterway**	**Barren Land**	**Other**	**Average**
Recall	0.852	0.416	0.771	0.675	0.731	0.767	0.765	0.474	0.713	0.786	**0.695**
Precision	0.882	0.592	0.849	0.768	0.951	0.754	0.774	0.874	0.674	0.718	**0.784**

Next, 3 more layers added and VGG13 with convolutional kernel size 5x5 was trained in a reduced FCN8s from scratch. After training the network, the results in Table 4.16 show that further increasing network depth with 5x5 filters did not have any significant contribution. In fact, average recall is worse than even the first proposed

network with 3x3 filters. Hence, this result indicates that the network has larger number of weights than the training set could train.

In order to benefit from the pretrained VGG16, another network is designed by using VGG16 with ImageNet trained weights and 3 consecutive deconvolutional layers, without any connection from pool4 and pool3 features. Not connecting pooling features during upsampling the pixel-wise classification has less parameters to be trained. Since seg2248 dataset consists of multispectral images with 4 channels, directly importing the ImageNet pretrained weights to the network is not possible, since ImageNet data contains 3 channel images. The problem occurs with the first layer convolutional filter due to the input channel differences. Therefore, pretrained weights were imported to the network except the first layer convolutional filters, which are randomly initialized. The recall and precision metrics are presented in Table 4.17.

The average recall and precision show that fine-tuning the pretrained weights with bilinear initialized deconvolutional layers gave the most accurate performance when compared to previously proposed approaches.

Finally, the original FCN8s network along with pool4 and pool3 feature skip connections was trained both from scratch using seg2248 dataset and over ImageNet pretrained weights. FCN8s network fine-tuned over ImageNet pretrained weights with seg2248 dataset results are presented in Table 4.18.

As in the previous networks, 'cultivated land' class has the poorest recall rate and second poorest precision rate after 'waterway' class in the fine-tuned network. In addition, fine-tuned FCN8s has the highest recall rate, but has precision 2% less than FCN8s without pooling connections. However, the effect of pooling feature connection can be observed clearly in Figure 4.9 that the network with higher resolution pooling feature connections can learn the boundary shape better.

In order to compare the effect of using pretrained network on remote sensing se-

Table 4.16: Precision and Recall values of seg2248 dataset trained on modified FCN8s (VGG13 with 5x5 convolutional filters) network from scratch

Class	**Sea**	**Cultivated Land**	**Lake**	**Rock**	**River**	**Forest**	**Shoreline**	**Waterway**	**Barren Land**	**Other**	**Average**
Recall	0.817	0.272	0.908	0.523	0.737	0.329	0.843	0.459	0.694	0.797	**0.638**
Precision	0.857	0.573	0.727	0.796	0.957	0.872	0.664	0.795	0.658	0.689	**0.759**

Table 4.17: Precision and Recall values of seg2248 dataset fine-tuned ImageNet pre-trained FCN8s (no skip connection) network

Class	Sea	Cultivated Land	Lake	Rock	River	Forest	Shoreline	Waterway	Barren Land	Other	Average
Recall	0.955	0.753	0.893	0.848	0.884	0.847	0.813	0.734	0.796	0.779	**0.830**
Precision	0.876	0.718	0.918	0.787	0.905	0.834	0.862	0.833	0.776	0.881	**0.839**

mantic segmentation performance, FCN8s was trained also from randomly initialized weights. Training such a large network from scratch took 48 hours on 6 GB GTX 1060 GPU. The confusion matrix for randomly initialized network is given in Table 4.19. Average recall of this network is calculated as **0.699** with average precision **0.767**. It can be concluded that using pretrained weights significantly improves the accuracy, especially training with a small dataset, such as satellite images even without using feature skip connections.

From confusion matrix given in Table 4.19, it can be observed that between similar classes such as 'sea', 'lake', 'river', and 'waterway' the confusion rate is quite high, as expected. Due to the fact that labeled pixel number is smaller than unlabeled pixel number in most of the images in the dataset, 'other' class dominates the network decision mechanism.

Small sized natural areas, such as 'river', and 'waterway', are the most difficult classes to be segmented by FCN8s model because the spatial information might got lost by the network during pooling operations. However, results of the conducted experiments indicate that using pretrained weights is more beneficial than training shallow networks from scratch. Fine-tuned FCN8s is obviously more accurate than the network trained from scratch due to the small amount of the data in the training set. Hence, ImageNet pretrained weights boost FCN8s network to segment natural areas in satellite images more accurately. The results of the experiments are combined in Table 4.20.

Table 4.18: Precision and Recall values of seg2248 fine-tuned ImageNet pretrained FCN8s network

Class	Sea	Cultivated Land	Lake	Rock	River	Forest	Shoreline	Waterway	Barren Land	Other	Average
Recall	0.969	0.602	0.871	0.874	0.936	0.891	0.892	0.829	0.842	0.744	**0.845**
Precision	0.911	0.705	0.929	0.687	0.926	0.807	0.764	0.921	0.687	0.852	**0.818**

Table 4.19: Confusion matrix for FCN8s image segmentation network trained with seg2248 dataset from scratch

Class	**Sea**	**Cultivated Land**	**Lake**	**Rock**	**River**	**Forest**	**Shoreline**	**Waterway**	**Barren Land**	**Other**	**Recall**
Sea	4658806	31378	301329	4525	591	6767	26932	4052	8714	382173	0.859
Cultv. L.	135351	2577323	197145	6951	6237	113089	8839	1895	387375	1172539	0.559
Lake	293869	160877	3391115	1657	6337	11583	615	3994	11795	290218	0.813
Rock	1686	6236	269	2108922	186	34246	8798	1886	56658	761826	0.708
River	6543	14638	47842	434	429478	1961	82	2516	15648	126006	0.666
Forest	7915	55053	4880	31265	336	4796853	139	1772	55830	1212715	0.778
Shoreline	44028	5475	1381	7867	118	1798	1154696	659	130493	341431	0.684
Waterway	622	3112	653	234	3651	1824	467	245796	28757	257957	0.453
Barren L.	8435	214467	5764	17163	2139	79119	76718	8522	5100268	1684826	0.709
Other	75276	116408	52071	62824	198	40895	9202	6431	416430	5527794	0.761
Precision	0.821	0.705	0.795	0.752	0.869	0.797	0.771	0.754	0.703	0.710	

Table 4.20: Precision and Recall values of all proposed networks

Reduced FCN8s (VGG10+upsampling)											
Class	**Sea**	**Cultivated Land**	**Lake**	**Rock**	**River**	**Forest**	**Shoreline**	**Waterway**	**Barren Land**	**Other**	**Average**
Recall	0.858	0.233	0.803	0.701	0.726	0.721	0.779	0.431	0.581	0.815	**0.665**
Precision	0.821	0.583	0.813	0.716	0.902	0.775	0.771	0.967	0.703	0.681	**0.773**
Reduced FCN8s (VGG10 (5x5)+upsampling)											
Recall	0.852	0.416	0.771	0.675	0.731	0.767	0.765	0.474	0.713	0.786	**0.695**
Precision	0.882	0.592	0.849	0.768	0.951	0.754	0.774	0.874	0.674	0.718	**0.784**
Modified FCN8s (VGG13 (5x5)+upsampling)											
Recall	0.817	0.272	0.908	0.523	0.737	0.329	0.843	0.459	0.694	0.797	**0.638**
Precision	0.857	0.573	0.727	0.796	0.957	0.872	0.664	0.795	0.658	0.689	**0.759**
FCN8s - fine-tuned over ImageNet weights (no skip connection)											
Recall	0.955	0.753	0.893	0.848	0.884	0.847	0.813	0.734	0.796	0.779	**0.830**
Precision	0.876	0.718	0.918	0.787	0.905	0.834	0.862	0.833	0.776	0.881	**0.839**
FCN8s - fine-tuned over ImageNet weights											
Recall	0.969	0.602	0.871	0.874	0.936	0.891	0.892	0.829	0.842	0.744	**0.845**
Precision	0.911	0.705	0.929	0.687	0.926	0.807	0.764	0.921	0.687	0.852	**0.818**
FCN8s - trained from scratch											
Recall	0.859	0.559	0.813	0.708	0.666	0.778	0.684	0.453	0.709	0.661	**0.699**
Precision	0.821	0.705	0.795	0.752	0.869	0.797	0.771	0.754	0.703	0.710	**0.767**

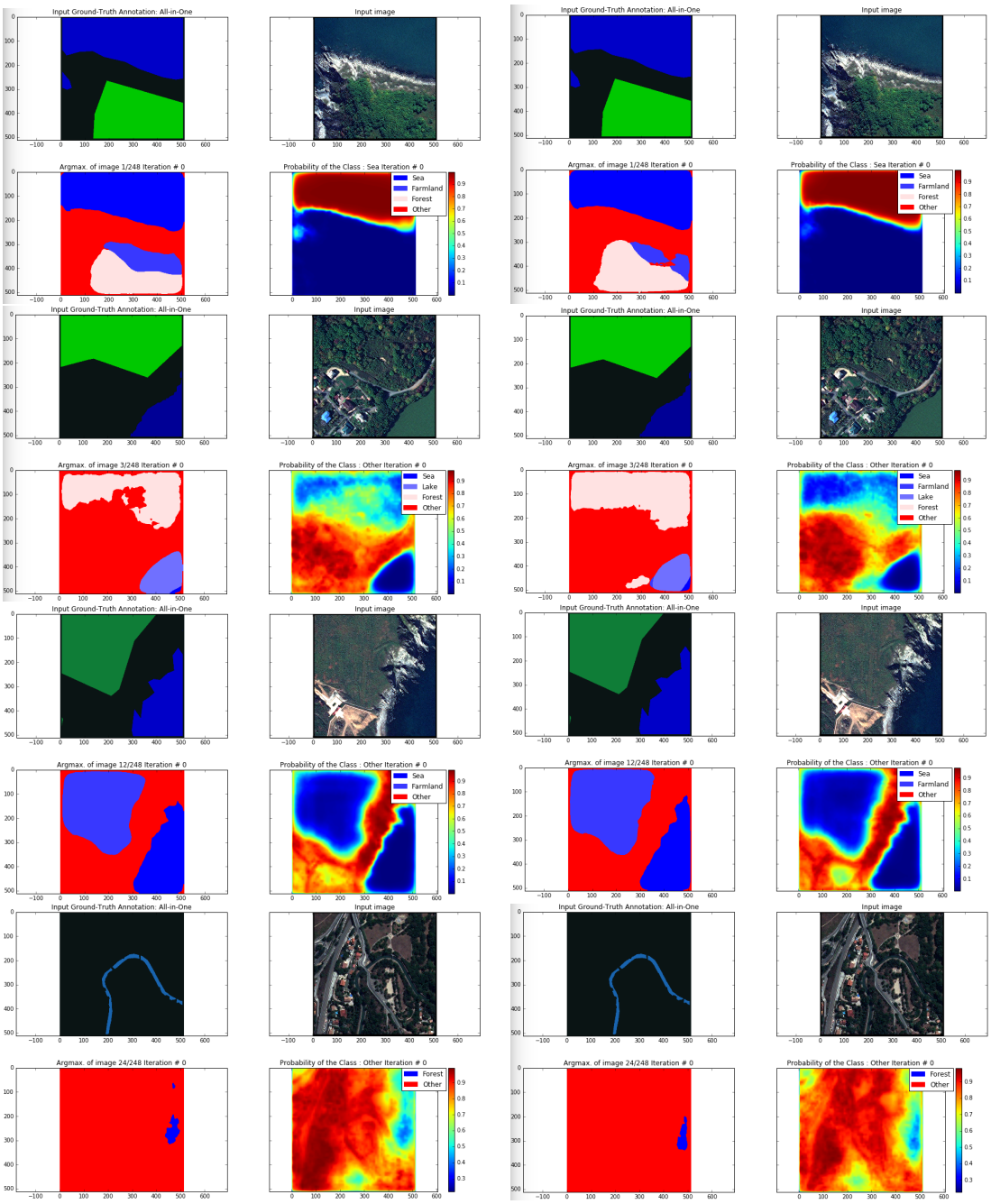


Figure 4.6: Sample outputs. Left - VGG10 (3x3), Right - VGG10 (5x5)

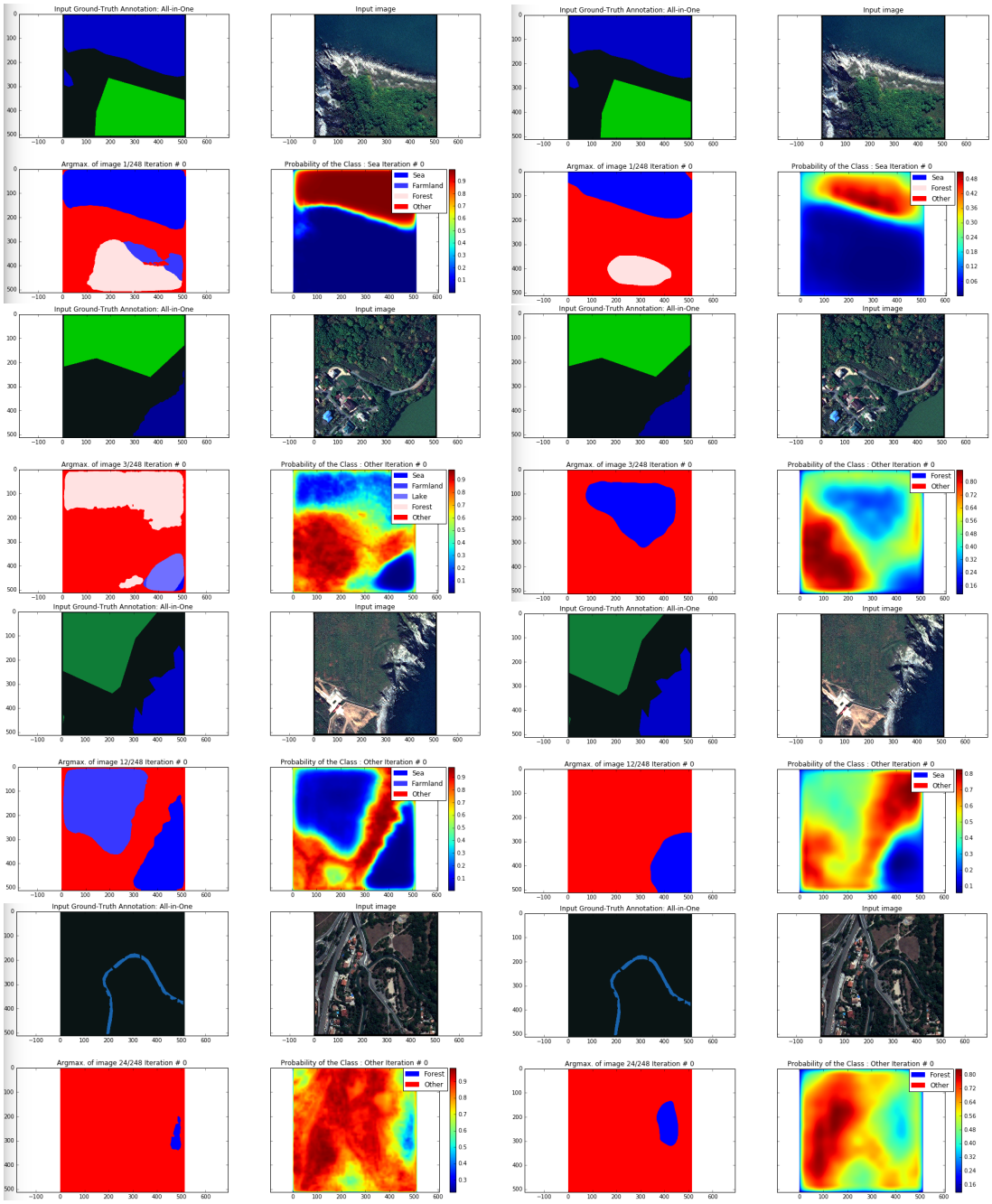


Figure 4.7: Sample outputs. Left - VGG10 (5x5), Right - VGG13 (5x5)

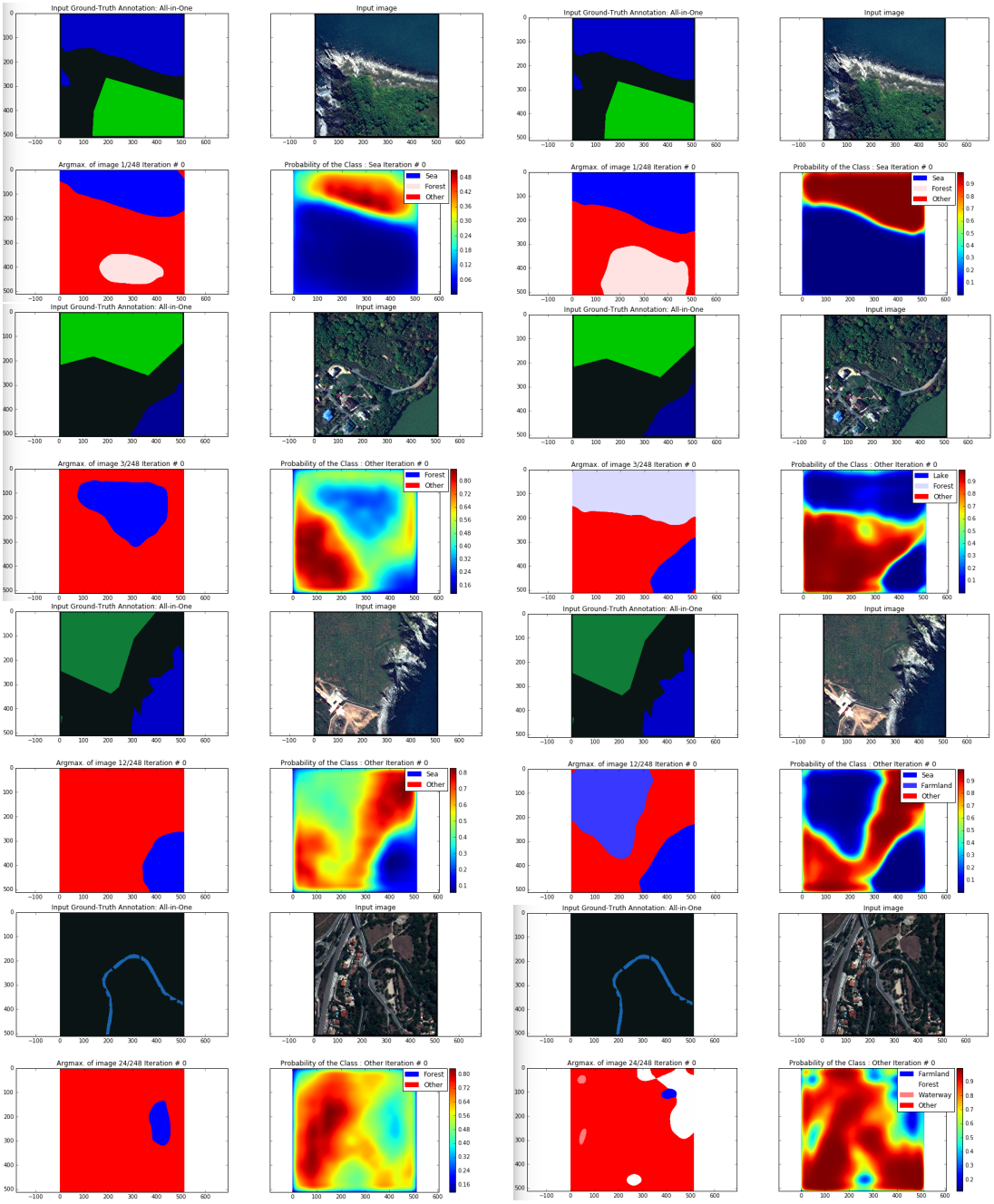


Figure 4.8: Sample outputs. Left - VGG13 (5x5), Right - FCN8s without pooling connections

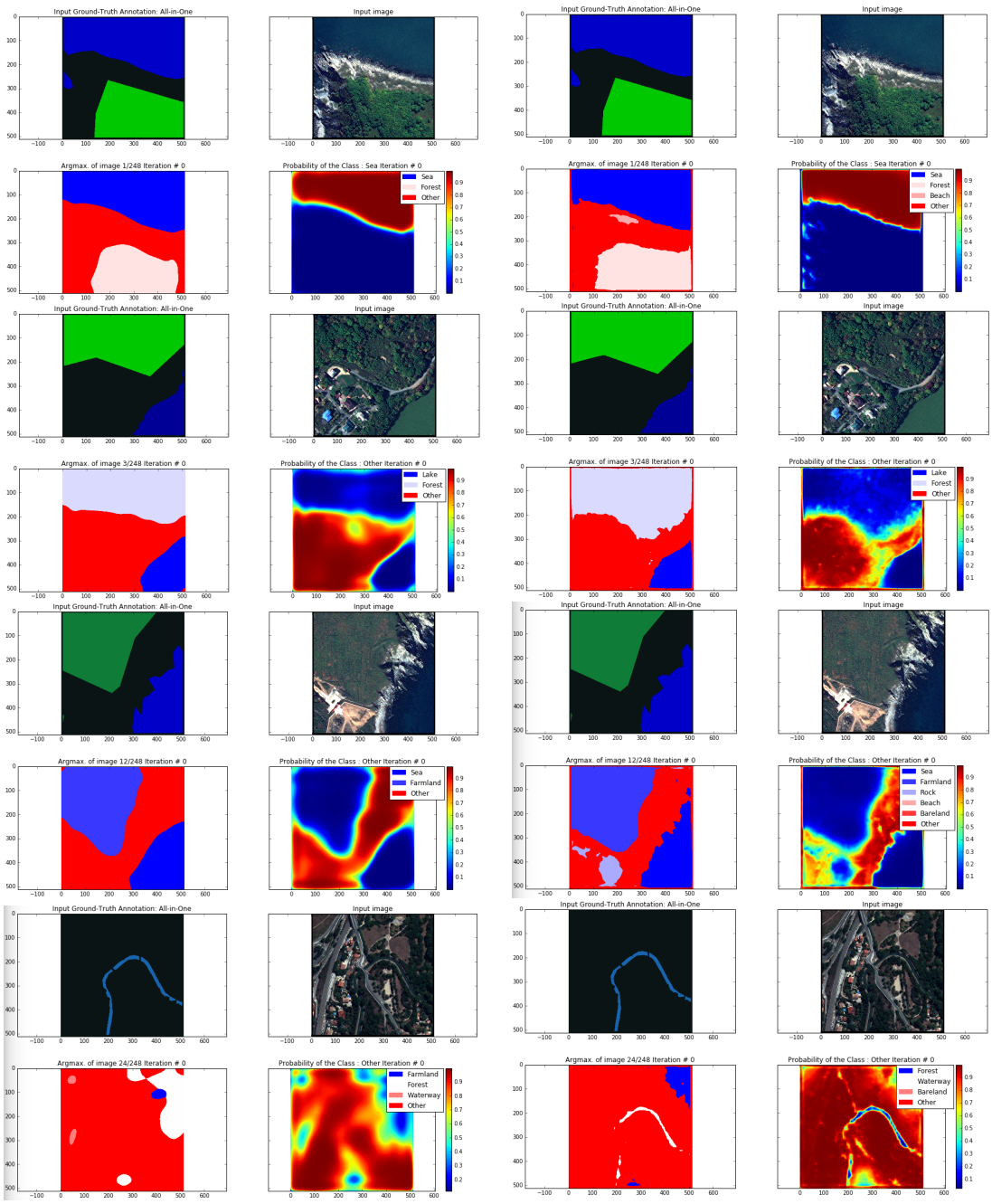


Figure 4.9: Sample outputs. Left - FCN8s without pooling connections, Right - original FCN8s. Both networks are ImageNet pretrained.

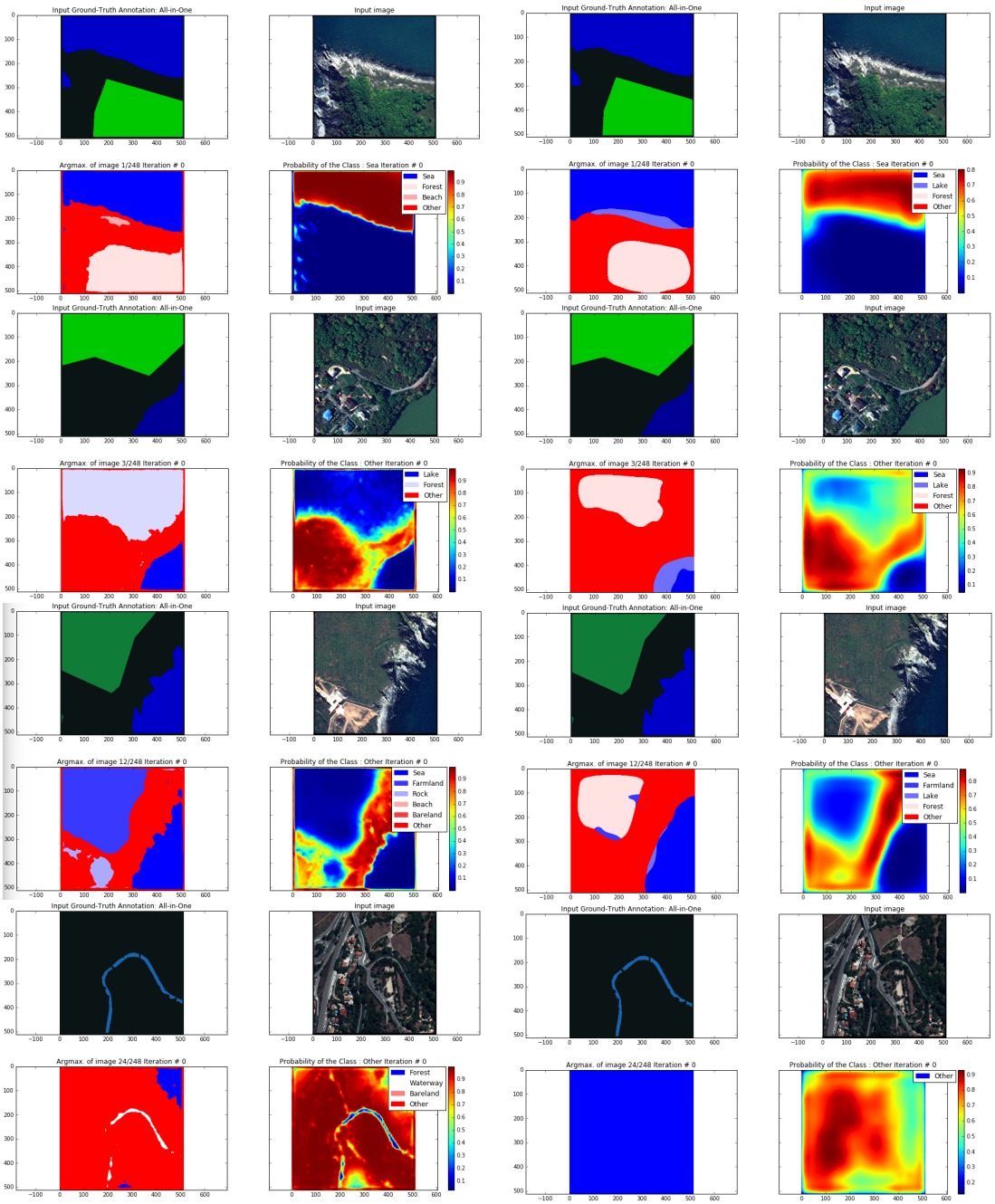


Figure 4.10: Sample outputs. Left - original FCN8s with ImageNet pretrained weights, Right - original FCN8s trained from scratch with seg2248 dataset

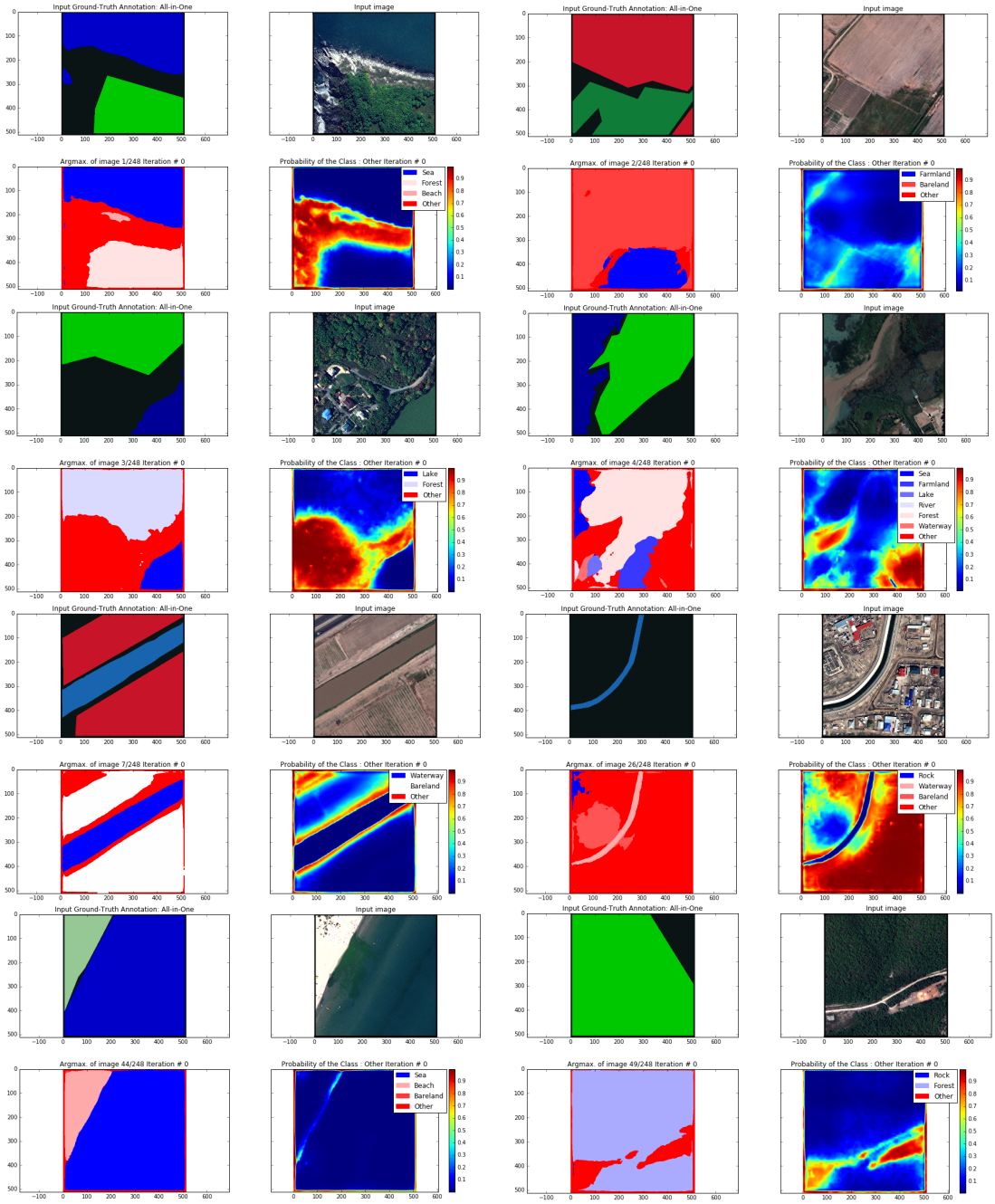


Figure 4.11: Sample output of FCN8s network fine-tuned with seg2248 dataset

CHAPTER 5

GEOSPATIAL OBJECT ANALYSIS USING DEEP NETWORKS

Small object detection from satellite images is one of the primary tasks of remote sensing image analysis. In general, man-made objects, such as airplanes, buildings, roads, ships, and vehicles are the main objects of interest in geospatial target detection. Although ImageNet competitions helped a rapid advance in general object detection research, the main difference and cause of difficulty of the object detection in satellite images is due to the relatively small size of the targets in remotely sensed images; i.e., typically about 20-50 pixels width and height for the satellites whose ground sampling distance is between 2.5m to 50cm. However on the positive side, there is not only a small variation in the observation angle, allowing to estimate the pixel sizes of the objects of interest in advance, but also no affine deformation of the objects due to oblique viewing of the scene.

As discussed in Chapter 3, a CNN based network named "You Only Look Once (YOLO)", a learning based detection technique, gives quite promising results and hence, is selected for object detection problem and tested using multispectral satellite images. The examined architecture and its corresponding experimental results are presented in this chapter.

5.1 Proposed Network

During object detection experiments, a modified version of YOLOv2 is utilized to analyze the limits of the performance of geospatial object detection. For this purpose, original YOLOv2 network is trained first for classification task with ImageNet1000

Table 5.1: Networks trained to detect Building and Aircraft Objects in Remote Sensing Images. B is the number of color bands used as input

	t-yolo7-2c	**t-yolo13-2c**	**t-yolo19-2c**	**t-yolo23-2c**
input layer	224x224xB	416x416xB	608x608xB	736x736xB
c1-(3,16)	224x224x16	416x416x16	608x608x16	736x736x16
m1-(2,2)	112x112x16	208x208x16	304x304x16	368x368x16
c2-(3,32)	112x112x32	208x208x32	304x304x32	368x368x32
m2-(2,2)	56x56x32	104x104x32	152x152x32	184x184x32
c3-(3,64)	56x56x64	104x104x64	152x152x64	184x184x64
m3-(2,2)	28x28x64	52x52x64	76x76x64	92x92x64
c4-(3,128)	28x28x128	52x52x128	76x76x128	92x92x128
m4-(2,2)	14x14x128	26x26x128	38x38x128	46x46x128
c5-(3,256)	14x14x256	26x26x256	38x38x256	46x46x256
m5-(2,2)	7x7x256	13x13x256	19x19x256	46x46x256
c6-(3,512)	7x7x512	13x13x512	19x19x512	46x46x512
m6-(2,1)	7x7x512	13x13x512	19x19x512	46x46x512
c7-(3,1024)	7x7x1024	13x13x1024	19x19x1024	23x23x1024
c8-(3,1024)	7x7x1024	13x13x1024	19x19x1024	23x23x1024
output layer	7x7x21	13x13x21	19x19x21	23x23x21

dataset, then modified and trained for detection with PASCAL VOC dataset. The classification network, namely *Darknet-19*, is used as baseline for YOLOv2, which has total 19 convolutional, 5 maxpool operations with stride 2 and 1 global average pooling layer.

The final version of YOLOv2 is capable of detecting 9,000 objects in general (non-aerial) images and it has a low-complexity to be used as real-time object detector. On the other hand, a smaller version of YOLOv2 called as tiny-YOLO network is also available that is trained for 80 class object detection task. This smaller network has 416x416x3 input layer, 8 convolutional layers with kernel size 3, 5 maxpooling layers with kernel size 2 and stride 2, 1 maxpooling layer with kernel size 2 but stride 1, and 13x13x(N*(C+5)) output map, where N is the number of anchor boxes used and C indicates the number of classes. The output of this network is 5 dimensional where these 5 dimensions represent the prediction box parameters, which are height, width, x and y values of the center point of the boxes and a confidence value.

Since tiny-YOLOv2 network is designed to detect PASCAL VOC classes and per-

Table 5.2: Proposed smaller network trained to detect only Aircraft in Remote Sensing Images

	verytiny-yolo23-1c
input layer	368x368xB
c1-(3,16)	368x368x16
m1-(2,2)	184x184x16
c2-(3,32)	184x184x32
m2-(2,2)	92x92x32
c3-(3,64)	92x92x64
m3-(2,2)	46x46x64
c4-(3,128)	46x46x128
m4-(2,2)	23x23x128
c5-(3,256)	23x23x256
c6-(3,256)	23x23x256
output layer	23x23x18

forms excellent in that dataset, it struggles when the image contains numerous smaller objects, such as buildings and airplanes, that are quite close to each other. In order to solve this issue, the output map size should be increased. This extension can be achieved by either increasing the input size of the network, or decreasing the depth of the network by removing one or more pooling layer. Moreover, the number of the utilized anchor boxes were reduced from default number 5 to 3 in the tests, while predicting the bounding boxes, since the size of the objects to be detected in this study does not vary significantly. The trained tiny-YOLOv2 with different output map sizes are given in Table 5.1. In order to show that a network with larger output map size performs more accurate than a network which is deeper but has smaller output map size, the network in Table 5.2 is proposed.

5.2 Experiments and Results

In this section, training a CNN based object detection method (YOLOv2) for geospatial object analysis is discussed in detail.

5.2.1 Dataset Generation for Object Detection

A new dataset is generated from WorldView-2 and GeoEye-1 satellite images and aircraft, building and ship ground-truth information is provided by manual labeling. For this purpose, several 4-band raster images with sizes around 20,000x20,000x4 are used in order to extract the image patches to be used as input to proposed networks. These satellite images have 0.46m x 0.46m spatial resolution with Red, Green, Blue and Near-InfraRed wavelengths. 3 datasets (named as *Aircraft*,*Ship*, and *Urban*, respectively) are generated for the task.

Image patches containing more than 10 buildings are assigned to Urban dataset. In addition, Aircraft and Ship datasets are generated with images containing at least 1 aircraft, or 1 ship, but there is no upper limit. Since the sizes objects to be detected vary from several meters up to 90 meters maximum, the image sizes are selected to be 250 meter per 250 meter, which corresponds to 500x500x4 in pixels.

Urban training set contains 1,254 images of size 500x500x4 with 19,715 annotated building objects whereas Aircraft training set contains 1,535 images with 8,313 annotated airplanes and Ship training set contains 2,096 images with 10,887 annotated ships in total. Two other datasets are obtained after combining Aircraft + Urban training sets first and then Aircraft + Ship training sets to be used in 2-class detection network training.

In order to observe the effect of the NIR band to the detection performance, RGB versions of the above 3 single class datasets are generated by excluding N-IR band. Sample training images are presented in Figure 5.1.

5.2.2 Experimental Setup

In order to show that increasing output map size also increases the performance of detecting smaller objects, several tiny YOLOv2 networks with different output map sizes were trained.

Tensorflow implementation of YOLO (i.e. Darkflow) is used for this study and modified for this purpose. Two separate computers with nVidia GTX 1080 (8 GB) and

Figure 5.1: Sample training set images of classes 'aircraft' on the left, 'building' in the middle, and 'ship' on the right column

GTX 1060 (6 GB) are used during the experiments. The batch sizes are kept constant at the maximum number that GPUs could support with slowly varying learning rate from 1E-3 to 1E-6. Table 5.3 summarizes the training properties of YOLOv2 used in our experiments while training in Darkflow framework.

5.2.3 Experimental Results for Object Detection

The networks in Table 5.1 are trained with Aircraft-Urban combined dataset and the results show that as the output size of the network increases, the performance of detecting smaller objects is getting better, as can be observed in Table 5.4. Sample detection results of this Aircraft-Urban network is given in Figure 5.2 F1 score is calculated as in Equation 5.1.

Table 5.3: Training properties of YOLOv2 used in this study

Properties	**YOLOv2**	**tiny-YOLOv2**
batch-norm	YES	YES
hi-res classifier	YES	NO
convolutional	YES	YES
anchor boxes	YES	YES
dimension priors	YES	YES
location prediction	YES	YES
passthrough	YES	NO
multi-scale	YES	NO
hi-res detector	YES	NO
adaptive learning rate	YES	YES

$$F1 = \frac{2(Recall)(Precision)}{(Recall + Precision)} \tag{5.1}$$

Both networks of 3-band input and 4-band input increases their capacity of detecting true positive significantly around 30 percent by slightly reducing the number of false positive with the increased resolution in the output map. Moreover, the networks that are fine-tuned with pretrained weights using only RGB data provide better results when compared to the networks trained from the scratch using all RGB and NIR data in the input.

In addition, Aircraft-Ship combined two-class dataset is used to train 23x23 output size tiny-YOLOv2 network to observe if two objects having low probability to happen

Table 5.4: Results of the network trained with Aircraft-Urban dataset. The network naming is (M)(N)(O)-(C)-(T): M is 3B/4B: num. of color band, N is T/VT: tiny or verytiny YOLOv2, O is 7/13/19/23: output size, C is 2: num. of class and T is the threshold of confidence

Network	**Recall**	**Precision**	**F1**
4BT7-2-.5	0.1272	0.1286	0.1279
4BT13-2-.5	0.2297	0.4010	0.2921
4BT19-2-.2	0.3094	0.4160	0.3549
3BT7-2-.5	0.1730	0.1960	0.1838
3BT13-2-.5	0.3056	0.5069	0.3813
3BT19-2-.2	0.4538	0.5126	**0.4814**

Table 5.5: Results of the network trained with Aircraft-Ship dataset and tested only single class detection

Network	Test Set	Recall	Precision	F1
4BT23-2-.3	aircraft only	0.7921	0.8364	0.8134
4BT23-2-.1	ship only	0.3874	0.3783	0.3824
4BT23-2-.3	both	0.4732	0.6592	0.5514

to be in the same scene could be detected as accurate as a single class network. Results in Table 5.5 shows the two-class detection test performance of the network. Sample detection results of Aircraft-Ship trained network is given in Figure 5.3.

Table 5.5 also gives the recall, precision, and F1 score of the above trained network tested only a single class detection. It is observed that ship object has lower recall and precision rate than aircraft detection.

The proposed network in Table 5.2 is trained for a single class only by using Aircraft dataset in order to observe whether decreasing the number of layers in the network but increasing the input and output map size improves the recall and precision or not.

The results in Table 5.6 shows that keeping the depth constant but increasing the resolution of the network increases the F1 score as expected. On the other hand, the proposed larger output map network with smaller number of layers, 4BVT23-1, has higher F1 score when compared to deeper but smaller output map 4BT19-1 network. This result means that by spending less memory, the performance of detection can be increased by using larger output map size.

Another result of these experiments is that 3-band networks with pretrained weights

Table 5.6: Results of the networks for only Aircraft detection

Network	Recall	Precision	F1
3BT19-2-.2	0.7394	0.6892	0.7134
4BT19-2-.2	0.6865	0.6263	0.6550
3BT19-1-.2	0.7964	0.7071	0.7491
4BT19-1-.2	0.7583	0.6526	0.7015
3BT23-1-.2	0.8273	0.7661	0.7955
4BT23-1-.2	0.8149	0.8638	**0.8386**
4BVT23-1-.2	0.6979	0.7439	0.7202

Table 5.7: Single detection result of the network trained with Ship dataset

Network	**Recall**	**Precision**	**F1**
4BT23-1-.3	0.4142	0.4933	0.4486

outperforms the 4-band networks which are trained from scratch. At last, 2-class trained networks 3BT19-2 and 4BT19-2 are less accurate to detect only aircrafts when compared to the single class trained networks 3BT19-1 and 4BT19-1. One result worth to note is that 3BT19-2 network has higher F1 score than 4BT19-1, which indicates the generalization ability of using pretrained weights. Besides, it is confirmed that aircraft only trained and tested network is more accurate to detect objects than aircraft-building and aircraft-ship trained but only aircraft object tested networks.

Finally, ship-only network is trained and tested for single class detection. The results are presented in Table 5.7. The reason of low F1 score for both building and ship detection in one or two-class networks is that these objects are found in a scene that has high textural complexity with many shapes and other objects, whereas the aircrafts are found in places where the terrain and scene does not have complex other objects or shapes.

Figure 5.2: Sample output images of 3BT19-2-.2 network, 'aircraft' on the left and 'building' on the right

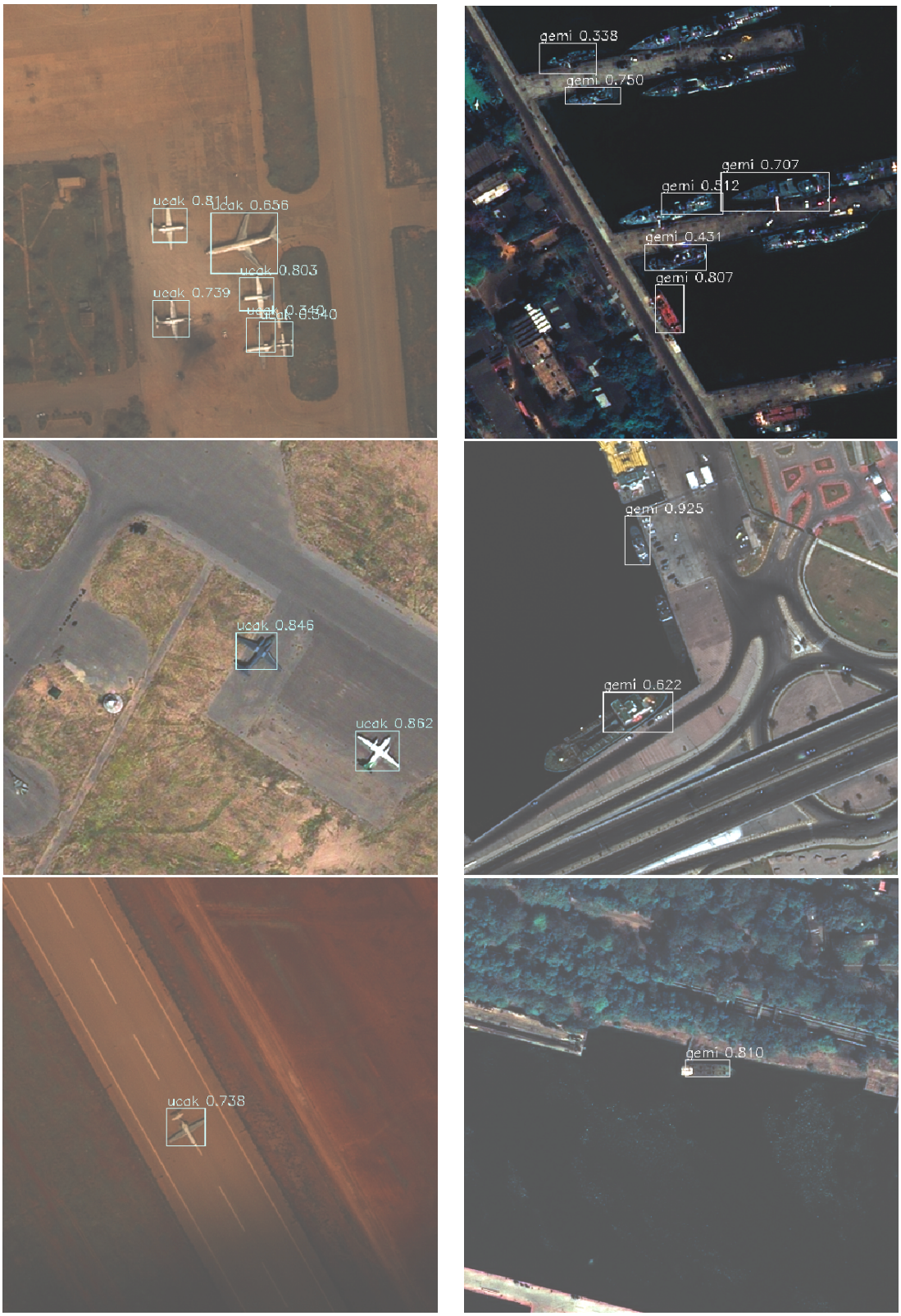


Figure 5.3: Sample output images of 4BT23-2-.2 network, 'aircraft' on the left and 'ship' on the right

CHAPTER 6

CONCLUSIONS

6.1 Summary

Convolutional neural networks are proven to outperform the traditional techniques in many applications, including remote sensing image analysis. Training a CNN requires a proper selection of hyperparameter set, which can be obtained using hyperparameter optimization methods such as grid search, random search or Bayesian hyperparameter optimization. Even though the deeper networks are believed to perform better, improper selection of hyperparameters and optimization methods could yield a worse performance when compared to a shallower but properly trained network. Moreover, increasing the depth of the network requires increasing amount of data in the training set. Therefore, using pretrained networks for remote sensing image analysis is a good idea because it is very difficult to obtain huge amount of labeled remote sensing dataset.

In this study, image scene classification architectures are examined for satellite images. Several parameters of CNN architectures including depth of network and CNN architecture are tuned to gain insight about training a CNN. Later, semantic image segmentation task is performed for the same topic. It is shown one more time that pretrained weights boost the performance of remote sensing image analysis. At last, geospatial object detection network is trained to detect 'airplane', 'building' and 'ship' classes. It is noted one more time that fine-tuning an ImageNet1000 pretrained network results in to be more accurate than a network trained from scratch, even though an additional information is available as 4th channel (Near Infra-Red band).

6.2 Conclusion

The conclusions that can be drawn from the experiments in this thesis can be stated as follows:

CNN architecture are powerful classifiers, detectors, and segmentation algorithms even for satellite images with relatively small resolutions.

One of the most important conclusions of this study is that if one has a relatively small labeled dataset for a learning problem, such as remote sensing image analysis, using pretrained networks significantly and consistently performs better than its shallower variants that are trained from scratch.

During this thesis, it has been observed that special attention should be given to the training stage of such deep networks.

Based on the experimental results, learning rate should be selected properly to train a network to the global optimum. Smaller learning rates resulted in longer training time, while larger ones resulted the network got stuck on a local minimum. Therefore, gradually decreasing the learning rate is usually the best option.

Dropout could be tried with 0.5 probability to avoid overfitting; however it cannot be stated that 0.5 dropout ratio is the default choice for remote sensing applications. Moreover, keeping the convolutional filter size as small as possible while increasing the depth of the network always gave the best results in all experiments.

Max pooling is the most useful selection of activation function rather than average pooling for deep learning in remote sensing applications since neurons do not suffer from saturating unlike other functions such as sigmoid or hyperbolic tangent.

Fully connected layers bring network a heavy burden of trainable parameters whereas convolutional layer filters bring significantly less number of trainable parameter burden to the network. Moreover, deeper networks can extract more semantic information. Therefore, increasing the depth of the network should be performed by adding more convolutional layers with pooling operation afterwards rather than adding fully connected layers before the output layer.

In addition to those, fully connected layers can be replaced with 1x1 convolutional layers, enabling the network perform pixel-wise classification with deconvolutional networks at the output map. Moreover, fusing higher resolution features into deconvolutional network outputs refines the boundary shapes of the segmented areas.

On the other hand, for object detection using bounding boxes task, multiclass networks cannot learn as accurate as single class networks. Detection of objects (aircrafts and ships) that have low probability to be found in the same image can be performed more accurately with two separate networks trained for single class. However, this costs 2 network training time and 2 network parameter storage concern. Therefore, one network for these two classes can be trained and used with slightly accuracy trade-off.

6.3 Future Works

Deep learning algorithms such as CNNs result in remarkable results in computer vision and remote sensing tasks. The attractive parts of such algorithms is that the pretrained networks such as ImageNet1000 trained AlexNet, VGGNet, GoogLeNet etc. are capable of generalizing for other domains such as remote sensing. As future studies, first of all, handmade ground truth information for classification (ekilialan), segmentation (seg2248), and detection (Aircraft, Ship, Urban) datasets should be refined and corrected. Training the classification base network of both semantic segmentation and detection networks could be trained beforehand using a handmade generated remote sensing classification dataset such as UCML. CRF post-processing could be applied just after the segmentation output to have better boundary shapes. Similar to the approach used in segmentation task, higher resolution pooling features could be fused in the detection output map as heavily utilized in SSD network.

REFERENCES

[1] Hai Wang, Yingfeng Cai, and Long Chen, "A vehicle detection algorithm based on deep belief network," vol. 2014, pp. 647380, 05 2014.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., pp. 1097–1105. Curran Associates, Inc., 2012.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[4] X. Yao, J. Han, G. Cheng, X. Qian, and L. Guo, "Semantic annotation of high-resolution satellite images via weakly supervised learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 6, pp. 3660–3671, June 2016.

[5] Q. Zou, L. Ni, T. Zhang, and Q. Wang, "Deep learning based feature selection for remote sensing scene classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 11, pp. 2321–2325, Nov 2015.

[6] G. Cheng, J. Han, L. Guo, Z. Liu, S. Bu, and J. Ren, "Effective and efficient midlevel visual elements-oriented land-use classification using vhr remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 8, pp. 4238–4249, Aug 2015.

[7] A. M. Cheriyadat, "Unsupervised Feature Learning for Aerial Scene Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, pp. 439–451, Jan. 2014.

[8] Gong Cheng, Junwei Han, Peicheng Zhou, and Lei Guo, "Multi-class geospatial object detection and geographic image classification based on collection of part detectors," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 98, pp. 119 – 132, 2014.

[9] X. Zheng, X. Sun, K. Fu, and H. Wang, "Automatic annotation of satellite images via multifeature joint sparse coding with spatial relation constraint," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 4, pp. 652–656, July 2013.

[10] Guofeng Sheng, Wen Yang, Tao Xu, and Hong Sun, "High-resolution satellite scene classification using a sparse coding based multiple feature combination," *International Journal of Remote Sensing*, vol. 33, no. 8, pp. 2395–2412, 2012.

[11] Yi Yang and Shawn Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, New York, NY, USA, 2010, GIS '10, pp. 270–279, ACM.

[12] Xiaoxiao Li and Guofan Shao, "Object-based urban vegetation mapping with high-resolution aerial photography as a single data source," *International Journal of Remote Sensing*, vol. 34, no. 3, pp. 771–789, 2013.

[13] Jason Walker and John Briggs, "An object-oriented approach to urban forest mapping in phoenix," vol. 73, pp. 577–583, 05 2007.

[14] J. A. dos Santos, O. A. B. Penatti, and R. da S. Torres, "Evaluating the potential of texture and color descriptors for remote sensing image retrieval and classification," in *VISAPP*, Angers, France, May 2010.

[15] Weixun Zhou, Zhenfeng Shao, Chunyuan Diao, and Qimin Cheng, "High-resolution remote-sensing imagery retrieval using sparse features by auto-encoder," *Remote Sensing Letters*, vol. 6, no. 10, pp. 775–783, 2015.

[16] G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 12, pp. 7405–7415, Dec 2016.

[17] Peicheng Zhou, Gong Cheng, Zhenbao Liu, Shuhui Bu, and Xintao Hu, "Weakly supervised target detection in remote sensing images based on transferred deep features and negative bootstrapping," *Multidimensional Syst. Signal Process.*, vol. 27, no. 4, pp. 925–944, Oct. 2016.

[18] Z. Shi, X. Yu, Z. Jiang, and B. Li, "Ship detection in high-resolution optical imagery based on anomaly detector and local shape feature," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 8, pp. 4511–4523, Aug 2014.

[19] T. R. Martha, N. Kerle, C. J. van Westen, V. Jetten, and K. V. Kumar, "Segment optimization and data-driven thresholding for knowledge-based landslide detection by object-based image analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 12, pp. 4928–4943, Dec 2011.

[20] André Stumpf and Norman Kerle, "Object-oriented mapping of landslides using random forests," *Remote Sensing of Environment*, vol. 115, no. 10, pp. 2564 – 2577, 2011.

[21] Thomas Blaschke and Josef Strobl, "Whats wrong with pixels? some recent developments interfacing remote sensing and gis," vol. 14, pp. 12 – 17, 06 2001.

[22] T. Blaschke, "Object based image analysis for remote sensing," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 65, no. 1, pp. 2 – 16, 2010.

[23] Thomas Blaschke, Geoffrey J. Hay, Maggi Kelly, Stefan Lang, Peter Hofmann, Elisabeth Addink, Raul Queiroz Feitosa, Freek van der Meer, Harald van der Werff, Frieke van Coillie, and Dirk Tiede, "Geographic object-based image analysis towards a new paradigm," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 87, pp. 180 – 191, 2014.

[24] Otávio Augusto Bizetto Penatti, Keiller Nogueira, and Jefersson Alex dos Santos, "Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?," in *CVPR Workshops*. 2015, pp. 44–51, IEEE Computer Society.

[25] Michael J. Swain and Dana H. Ballard, "Color indexing," *Int. J. Comput. Vision*, vol. 7, no. 1, pp. 11–32, Nov. 1991.

[26] Yi Yang and Shawn D. Newsam, "Comparing sift descriptors and gabor texture features for classification of remote sensed imagery," *2008 15th IEEE International Conference on Image Processing*, pp. 1852–1855, 2008.

[27] Xin Huang, Liangpei Zhang, and Le Wang, "Evaluation of morphological texture features for mangrove forest mapping and species discrimination using multispectral IKONOS imagery," *IEEE Geosci. Remote Sensing Lett.*, vol. 6, no. 3, pp. 393–397, 2009.

[28] G. Cheng, J. Han, L. Guo, and T. Liu, "Learning coarse-to-fine sparselets for efficient object detection and scene classification," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1173–1181.

[29] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[30] Aude Oliva and Antonio Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vision*, vol. 42, no. 3, pp. 145–175, May 2001.

[31] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection," in *In CVPR*, 2005, pp. 886–893.

[32] R. Haralick, K. Shanmugam, and I. Dinstein, "Texture features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, no. 6, 1973.

[33] Anil K. Jain, Nalini K. Ratha, and Sridhar Lakshmanan, "Object detection using gabor filters," *Pattern Recognition*, vol. 30, no. 2, pp. 295 – 309, 1997.

[34] Yan Ke and Rahul Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, 2004, CVPR'04, pp. 506–513, IEEE Computer Society.

[35] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, "Surf: Speeded up robust features," in *In ECCV*, 2006, pp. 404–417.

[36] Zuyu Zhang Yonglin Shen Bin Zhang, Yueyan Liu, "Land use and land cover classification for rural residential areas in china using soft-probability cascading of multifeatures," *Journal of Applied Remote Sensing*, vol. 11, pp. 11 – 11 – 17, 2017.

[37] Z. Li and L. Itti, "Saliency and gist features for target detection in satellite images," *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 2017–2029, July 2011.

[38] J. Yin, H. Li, and X. Jia, "Crater detection based on gist features," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 1, pp. 23–29, Jan 2015.

[39] Wanceng Zhang, Xian Sun, Kun Fu, Chenyuan Wang, and Hongqi Wang, "Object detection in high-resolution remote sensing images using rotation invariant parts based model," *IEEE Geosci. Remote Sensing Lett.*, vol. 11, no. 1, pp. 74–78, 2014.

[40] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, Jul 2002.

[41] Jianfeng Ren, Xudong Jiang, and Junsong Yuan, "Learning lbp structure by maximizing the conditional mutual information," *Pattern Recogn.*, vol. 48, no. 10, pp. 3180–3190, Oct. 2015.

[42] S. Chaib, Y. Gu, and H. Yao, "An informative feature selection method based on sparse pca for vhr scene classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 2, pp. 147–151, Feb 2016.

[43] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao, "Stacked convolutional denoising auto-encoders for feature representation," *IEEE Transactions on Cybernetics*, vol. 47, no. 4, pp. 1017–1027, April 2017.

[44] I.T. Jolliffe, *Principal Component Analysis*, Springer Verlag, 1986.

[45] Bruno A. Olshausen and David J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?," *Vision Research*, vol. 37, no. 23, pp. 3311 – 3325, 1997.

[46] G E Hinton and R R Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, July 2006.

[47] Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma, "Pcanet: A simple deep learning baseline for image classification?," *CoRR*, vol. abs/1404.3606, 2014.

[48] K. Qi, H. Wu, C. Shen, and J. Gong, "Land-use scene classification in high-resolution remote sensing images using improved correlatons," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 12, pp. 2403–2407, Dec 2015.

[49] Y. Zhang, X. Sun, H. Wang, and K. Fu, "High-resolution remote-sensing image classification via an approximate earth mover's distance-based bag-of-features model," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 5, pp. 1055–1059, Sept 2013.

[50] Bei Zhao, Yanfei Zhong, and Liangpei Zhang, "A spectralstructural bag-of-features scene classifier for very high spatial resolution remote sensing imagery," vol. 116, pp. 73–85, 06 2016.

[51] Kunlun Qi, Zhang Xiaochun, Wu Baiyan, and Wu Huayi, "Sparse coding-based correlaton model for land-use scene classification in high-resolution remote-sensing images," vol. 10, pp. 042005, 06 2016.

[52] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 3360–3367.

[53] F. Zhang, B. Du, and L. Zhang, "Saliency-guided unsupervised feature learning for scene classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 4, pp. 2175–2184, April 2015.

[54] Fan Hu, Gui-Song Xia, Jingwen Hu, and Liangpei Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sensing*, vol. 7, no. 11, pp. 14680–14707, 2015.

[55] F. Zhang, B. Du, and L. Zhang, "Scene classification via a gradient boosting random convolutional network framework," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 3, pp. 1793–1802, March 2016.

[56] Wenzhi Zhao and Shihong Du, "Scene classification using multi-scale deeply described visual words," *International Journal of Remote Sensing*, vol. 37, no. 17, pp. 4119–4131, 2016.

[57] Francois P. S. Luus, Brian P. Salmon, Frans van den Bergh, and Bodhaswar T. Maharaj, "Multiview deep learning for land-use classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, pp. 2448–2452, 2015.

[58] Dimitrios Marmanis, Mihai Datcu, Thomas Esch, and Uwe Stilla, "Deep learning earth observation classification using imagenet pretrained networks.," *IEEE Geosci. Remote Sensing Lett.*, vol. 13, no. 1, pp. 105–109, 2016.

[59] Martin Längkvist, Andrey Kiselev, Marjan Alirezaie, and Amy Loutfi, "Classification and segmentation of satellite orthoimagery using convolutional neural networks," *Remote Sensing*, vol. 8, no. 4, 2016.

[60] Marco Castelluccio, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva, "Land use classification in remote sensing images by convolutional neural networks," *CoRR*, vol. abs/1508.00092, 2015.

[61] Keiller Nogueira, Otávio Augusto Bizetto Penatti, and Jefersson Alex dos Santos, "Towards better exploiting convolutional neural networks for remote sensing scene classification," *CoRR*, vol. abs/1602.01517, 2016.

[62] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, July 2006.

[63] Ruslan Salakhutdinov and Geoffrey Hinton, "An efficient learning procedure for deep boltzmann machines," *Neural Comput.*, vol. 24, no. 8, pp. 1967–2006, Aug. 2012.

[64] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.

[65] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[66] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[68] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle, "Greedy layer-wise training of deep networks," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, Cambridge, MA, USA, 2006, NIPS'06, pp. 153–160, MIT Press.

[69] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *CoRR*, vol. abs/1406.4729, 2014.

[70] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[71] Lucas Assirati, Alexandre Souto Martinez, and Odemir Martinez Bruno, "Satellite image classification and segmentation using non-additive entropy," *CoRR*, vol. abs/1401.2416, 2014.

[72] N. P. Deepika and K. Vishnu, "Different techniques for satellite image segmentation," in *2015 Online International Conference on Green Engineering and Technologies (IC-GET)*, Nov 2015, pp. 1–6.

[73] Jamie Sherrah, "Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery," *CoRR*, vol. abs/1606.02585, 2016.

[74] D. Marmanis, J. D. Wegner, S. Galliani, K. Schindler, M. Datcu, and U. Stilla, "Semantic Segmentation of Aerial Images with AN Ensemble of Cnns," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 473–480, June 2016.

[75] Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefèvre, "Semantic segmentation of earth observation data using multimodal and multi-scale deep networks," *CoRR*, vol. abs/1609.06846, 2016.

[76] S. Paisitkriangkrai, J. Sherrah, P. Janney, and A. van den Hengel, "Semantic labeling of aerial and satellite imagery," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 7, pp. 2868–2881, July 2016.

[77] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Convolutional neural networks for large-scale remote-sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 645–657, Feb 2017.

[78] Pascal Kaiser, Jan Dirk Wegner, Aurélien Lucchi, Martin Jaggi, Thomas Hofmann, and Konrad Schindler, "Learning aerial image segmentation from online maps," *CoRR*, vol. abs/1707.06879, 2017.

[79] J. Zuo, G. Xu, K. Fu, X. Sun, and H. Sun, "Aircraft type recognition based on segmentation with deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 2, pp. 282–286, Feb 2018.

[80] K. Chen, K. Fu, M. Yan, X. Gao, X. Sun, and X. Wei, "Semantic segmentation of aerial images with shuffling convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 2, pp. 173–177, Feb 2018.

[81] Wenkai Zhang, Hai Huang, Matthias Schmitz, Xian Sun, Hongqi Wang, and Helmut Mayer, "Effective fusion of multi-modal remote sensing data in a fully convolutional network for semantic labeling," *Remote Sensing*, vol. 10, no. 1, 2018.

[82] I. evo and A. Avramovi, "Convolutional neural network based automatic object detection on aerial images," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 5, pp. 740–744, May 2016.

[83] Saikat Basu, Sangram Ganguly, Supratik Mukhopadhyay, Robert DiBiano, Manohar Karki, and Ramakrishna R. Nemani, "Deepsat - A learning framework for satellite imagery," *CoRR*, vol. abs/1509.03602, 2015.

[84] Yanfei Zhong, Feng Fei, Yanfei Liu, Bei Zhao, Hongzan Jiao, and Liangpei Zhang, "Satcnn: satellite image dataset classification using agile convolutional neural networks," *Remote Sensing Letters*, vol. 8, no. 2, pp. 136–145, 2017.

[85] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[86] Zhang H. Zhang J. Han, Z. and X. Hu, "Fast aircraft detection based on region locating network in large-scale remote sensing images," in *2017 IEEE Conference on Image Processing (ICIP)*, 17-20 September 2017.

[87] Tomonori Yamamoto and Yoriko Kazama, "Ship detection leveraging deep neural networks in worldview-2 images," p. 39, 10 2017.

[88] Y. Long, Y. Gong, Z. Xiao, and Q. Liu, "Accurate object localization in remote sensing images based on convolutional neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 5, pp. 2486–2498, May 2017.

[89] Zhaozhuo Xu, Xin Xu, Lei Wang, Rui Yang, and Fangling Pu, "Deformable convnet with aspect ratio constrained nms for object detection in remote sensing imagery," *Remote Sensing*, vol. 9, no. 12, 2017.

[90] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[91] Matija Radovic, Offei Adarkwa, and Qiaosong Wang, "Object recognition in aerial images using convolutional neural networks," *Journal of Imaging*, vol. 3, no. 2, 2017.

[92] Jennifer Carlet and Bernard Abayowa, "Fast vehicle detection in aerial imagery," *CoRR*, vol. abs/1709.08666, 2017.

[93] Joseph Redmon and Ali Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016.

[94] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg, "Ssd: Single shot multibox detector," 2016, To appear.

[95] Zhong Chen, Ting Zhang, and Chao Ouyang, "End-to-end airplane detection using transfer learning in remote sensing images," *Remote Sensing*, vol. 10, no. 1, 2018.

[96] Tianyu Tang, Shilin Zhou, Zhipeng Deng, Lin Lei, and Huanxin Zou, "Arbitrary-oriented vehicle detection in aerial imagery with single convolutional neural networks," *Remote Sensing*, vol. 9, no. 11, 2017.

[97] Kurt Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251 – 257, 1991.

[98] Y. LeCun, "A theoretical framework for back-propagation," in *Proceedings of the 1988 Connectionist Models Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds., CMU, Pittsburgh, Pa, 1988, pp. 21–28, Morgan Kaufmann.

[99] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[100] Léon Bottou, "Large-scale machine learning with stochastic gradient descent," in *in COMPSTAT*, 2010.

[101] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. 2013, ICML'13, pp. III–1139–III–1147, JMLR.org.

[102] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[103] Abdel-rahman Mohamed, Tara N. Sainath, George E. Dahl, Bhuvana Ramabhadran, Geoffrey E. Hinton, and Michael A. Picheny, "Deep belief networks using discriminative features for phone recognition," in *ICASSP*. 2011, pp. 5060–5063, IEEE.

[104] Ronan Collobert and Jason Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, New York, NY, USA, 2008, ICML '08, pp. 160–167, ACM.

[105] Thomas Serre, Gabriel Kreiman, Minjoon Kouh, Charles Cadieu, Ulf Knoblich, and Tomaso Poggio, "A quantitative theory of immediate visual recognition," vol. 165, pp. 33–56, 02 2007.

[106] Yoav Freund and David Haussler, "Unsupervised learning of distributions on binary vectors using two layer networks," in *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds., pp. 912–919. Morgan-Kaufmann, 1992.

[107] Y. Chen, X. Zhao, and X. Jia, ," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*.

[108] T. Li, J. Zhang, and Y. Zhang, "Classification of hyperspectral image based on deep belief networks," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 5132–5136.

[109] Diao Wenhui, Xian Sun, Fangzheng Dou, Menglong Yan, Hongqi Wang, and Kun Fu, "Object recognition in remote sensing images using sparse deep belief networks," vol. 6, pp. 745–754, 10 2015.

[110] Q. Zou, L. Ni, T. Zhang, and Q. Wang, "Deep learning based feature selection for remote sensing scene classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 11, pp. 2321–2325, Nov 2015.

[111] J. Tang, C. Deng, G. B. Huang, and B. Zhao, "Compressed-domain ship detection on spaceborne optical image using deep neural network and extreme learning machine," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 3, pp. 1174–1185, March 2015.

[112] Xiaorui Ma, Jie Geng, and Hongyu Wang, "Hyperspectral image classification via contextual deep learning," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 1, pp. 20, Jul 2015.

[113] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015.

[114] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014.

[115] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.