

ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

**SEMANTIC LAND COVER AND LAND USE CLASSIFICATION USING DEEP
CONVOLUTIONAL NEURAL NETWORKS**



M.Sc. THESIS

Berk GÜNEY

Department of Communication Systems

Satellite Communication and Remote Sensing Programme

Thesis Advisor: Prof. Dr. Elif SERTEL

JUNE 2019

ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

**SEMANTIC LAND COVER AND LAND USE CLASSIFICATION USING DEEP
CONVOLUTIONAL NEURAL NETWORKS**



M.Sc. THESIS

**Berk Güney
(705161004)**

Department of Communication Systems

Satellite Communication and Remote Sensing Programme

Thesis Advisor: Prof. Dr. Elif Sertel

JUNE 2019

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

**DERİN EVRİŞİMSEL SINIR AĞLARI İLE ARAZİ KULLANIMI VE ARAZİ
ÖRTÜSÜNÜN ANLAMSAL SINIFLANDIRILMASI**

YÜKSEK LİSANS TEZİ

**Berk GÜNEY
(705161004)**

İletişim Sistemleri Anabilim Dalı

Uydu Haberleşmesi ve Uzaktan Algılama Programı

Tez Danışmanı: Prof. Dr. Elif SERTEL

HAZİRAN 2019

Berk Güney, a **M.Sc.** student of ITU Informatics Institute student ID **705161004** successfully defended the thesis entitled “**Semantic Land Cover and Land Use Classification using Deep Convolutional Neural Networks**” which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor : **Prof. Dr. Elif SERTEL**
İstanbul Technical University

Jury Members : **Prof. Dr. Şinasi KAYA**
İstanbul Technical University

Prof. Dr. Bülent BAYRAM
Yıldız Technical University

Date of Submission :

Date of Defense : **11 June 2019**





To my family,



FOREWORD

I would like to express my gratitude to my supervisor Prof. Dr. Elif SERTEL for her precious guidance and insistence for this study. She also provided hardware to be used in this thesis work. I also would like to thank Prof. Dr. Markus GERKE for his contribution and collaboration with my time in Technical University Braunschweig for the ERASMUS+ programme.

I am also grateful to Mehmet Soydaş for his generous support and help with his experience and knowledge in deep learning.

Finally, I would like to thank my family for their endless support throughout my study.

May 2019

Berk Güney



TABLE OF CONTENTS

	<u>Page</u>
FOREWORD	ix
TABLE OF CONTENTS	xi
ABBREVIATIONS	xiii
LIST OF TABLES	xv
LIST OF FIGURES	xvii
SUMMARY	xix
ÖZET	xxi
1. INTRODUCTION	1
1.1 Purpose of Thesis.....	3
1.2 Scope and the Organization of the Thesis.....	3
2. LITERATURE REVIEW ON LAND COVER AND LAND USE	
CLASSIFICATION	5
2.1 Pixel-Based Image Classification	5
2.2 Sub-Pixel Based Image Classification	6
2.3 Object-Based Image Classification.....	6
2.4 Spatio-Contextual Image Classification.....	7
2.5 Machine Learning in Remote Sensing	7
2.5.1 Supervised learning.....	8
2.5.2 Support vector machines	8
2.5.3 Decision tree classifier	10
2.5.4 Random forest classifier	12
2.5.5 Artificial neural networks.....	12
3. DEEP NEURAL NETWORKS	15
3.1 Autoencoders.....	15
3.1.1 Stacked autoencoders	16
3.1.2 Sparse autoencoders	17
3.2 Deep Belief Networks and Restricted Boltzmann Machine.....	17
3.3 Recurrent Neural Networks.....	18
3.4 Convolutional Neural Networks.....	20
3.4.1 Convolutional layer	21
3.4.2 Pooling layer	23
3.4.3 Fully connected layer	23
3.5 CNN Architectures.....	24
3.5.1 LeNet-5	24
3.5.2 AlexNet.....	25
3.5.3 VGGNet.....	27
3.5.4 GoogLeNet.....	28
3.5.5 ResNet	31
3.5.6 Inception-ResNet.....	33
3.6 Training Convolutional Neural Networks.....	35
3.6.1 Hyperparameter selection	35

3.6.1.1 Loss function	35
3.6.1.2 Learning rate.....	35
3.6.1.3 Mini-batch size.....	36
3.6.2 Optimization algorithms.....	36
3.6.2.1 Gradient Descent.....	36
3.6.2.2 Stochastic Gradient Descent(SGD).....	37
3.6.2.3 AdaGrad	38
3.6.2.4 RMSProp	39
3.6.2.5 Adam	39
3.6.3 Regularization.....	40
3.6.3.1 L2 regularization.....	40
3.6.3.2 L1 regularization.....	41
3.6.3.3 Dropout regularization	41
3.6.3.4 Batch normalization	42
3.6.3.5 Data augmentation	42
3.7 Transfer Learning	43
4. EXPERIMENTS AND RESULTS.....	45
4.1 Image Scene Classification for Land Cover and Land Use Analysis.....	45
4.1.1 Proposed classification networks.....	45
4.1.2 Training dataset for the classification network	46
4.1.3 Validation dataset for the classification network	50
4.1.4 Pre-processing	54
4.1.5 Training setup.....	54
4.1.6 Results.....	54
4.1.7 Discussion	70
5. CONCLUSIONS.....	71
5.1 Summary	71
5.2 Conclusions.....	72
5.3 Future Works.....	73
REFERENCES.....	75
CURRICULUM VITAE	81

ABBREVIATIONS

AE	: Auto-encoder
ANN	: Artificial Neural Network
AOI	: Area of interest
BN	: Batch Normalization
CV	: Computer vision
CNN	: Convolutional Neural Network
CRF	: Conditional Random Field
DT	: Decision Tree
DBN	: Deep Belief Network
DBM	: Deep Boltzmann Machine
LTU	: Linear Threshold Unit
MRF	: Markov Random Field
MLP	: Multi-Layer Perceptron
NN	: Neural Network
RBM	: Restricted Boltzmann Machine
ReLU	: Rectified Linear Unit
RF	: Random Forest
SAE	: Stacked Auto-Encoder
SGD	: Stochastic Gradient
SVM	: Support Vector Machine
UAV	: Unmanned Aerial Vehicle



LIST OF TABLES

	<u>Page</u>
Table 2.1 : The differences between pixel, sub-pixel and object based approaches. . .	6
Table 3.1 : LeNet-5 architecture	25
Table 3.2 : AlexNet architecture.....	26
Table 3.3 : VGGNet architecture.....	27
Table 3.4 : Number of parameters in VGGNet in millions.....	28
Table 3.5 : Shows the top-1 and top-5 error rates of ResNet models based on the validation set of ILSVRC 2014.....	33
Table 3.6 : The top-1 and top-5 error rates of Inception models based on the validation set of ILSVRC 2012.....	34
Table 4.3 : Results of the trained networks.....	54
Table 4.4 : Error types for classification.....	55
Table 4.5 : Precision recall and f1-scores for the Inception-ResNet-v2	56
Table 4.6 : Precision recall and f1-scores for the Inception-v4.....	56



LIST OF FIGURES

	<u>Page</u>
Figure 2.1 : Difference between classical programming and machine learning.....	8
Figure 2.2 : Difference between classification and regression	8
Figure 2.3 : Non-linear transform and optimal hyperplane for SVM.....	10
Figure 2.4 : Sample decision tree	11
Figure 2.5 : Linear threshold unit.....	13
Figure 2.6 : Multi-Layer Perceptron.....	14
Figure 3.1 : Autoencoder	16
Figure 3.2 : Stacked Autoencoder	16
Figure 3.3 : A DBN with two RBM's	18
Figure 3.4 : Backward connection of RNN	18
Figure 3.5 : LSTM cell	19
Figure 3.6 : LeNet architecture	20
Figure 3.7 : Comparison of input layers: Fully connected layer vs convolutional layer. Size of the local receptive field is 5x5	21
Figure 3.8 : 2-dimensional convolutional example with filter size 3x3 and stride 1 with zero padding.....	22
Figure 3.9 : Activation functions.....	22
Figure 3.10 : Max pooling and average pooling operations with a filter size 2x2 and stride 2.....	23
Figure 3.11 : Typical CNN architecture	24
Figure 3.12 : Inception Module.....	28
Figure 3.13 : Difference between Fully Connected Layer and Global Average Pooling.	28
Figure 3.14 : GoogleNet architecture	29
Figure 3.15 : Inception modules used in Inception-v2	30
Figure 3.16 : From left Inception modules A, B, C used in Inception-v4.....	31
Figure 3.17 : Residual Learning.....	32
Figure 3.18 : Regular deep neural network(left) and deep residual learning(right)..	32
Figure 3.19 : Residual unit.....	33
Figure 3.20 : Inception modules A,B,C in an Inception-ResNet-v1. Pooling layer was replaced by the residual connection.....	34
Figure 3.21 : Inception-ResNet-v2 architecture	34
Figure 3.22 : The effect of learning rate on training loss.	36
Figure 3.23 : Weight updates in the opposite direction of the gradient.....	37
Figure 3.24 : SGD fluctuates to find a newer and better local minima.	38
Figure 3.25 : Graph showing underfitting and overfitting in the network.....	40
Figure 3.26 : Dropout Regularization. Standart network(left) network with dropout(right)	41
Figure 3.27 : Transfer learning.....	43
Figure 4.1 : Sample patches for the training dataset.....	47
Figure 4.2 : Code for automated script to convert 16-bit imagery to 8-bit.....	50

Figure 4.3 : Sample patches for the validation dataset.....	51
Figure 4.4 : Confusion matrix for the Inception-ResNet-v2 network.....	57
Figure 4.5 : Confusion matrix for the Inception-v4 network.....	58
Figure 4.6 : Error instances of the trained networks	61
Figure 4.7 : Loss function for Inception-ResNet-v2.	62
Figure 4.8 : Loss function for Inception-v4.....	62
Figure 4.9 : Training accuracy of the networks	62
Figure 4.10 : Loss function for Inception-ResNet-v2 with %15 training ratio and no varying batch size.....	66
Figure 4.11 : Loss function for Inception-v4 with %15 training ratio and no varying batch size.....	66
Figure 4.12 : Confusion matrix for the Inception-ResNet-v2 with %15 training ratio of NWPU-RESISC45 dataset.....	67
Figure 4.13 : Confusion matrix for the Inception-v4 with %15 training ratio of NWPU-RESISC45 dataset.....	68



LAND COVER AND LAND USE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS

SUMMARY

In recent years, deep learning (DL), the successor of neural networks (NNs), has become the state-of-the-art approach in areas particularly, computer vision (CV), speech recognition and natural language processing. (NN) is an established branch of artificial intelligence that has been brought to life due to factors such as high-performance computing, algorithmic improvements and big data. In the field of remote sensing big data has also become the norm. Remote sensing is obtaining information about an object or phenomenon without making physical contact, especially the Earth. The definition includes the conventional areas of remote sensing, e.g. satellite and aerial photography. However, remote sensing also covers areas such as unmanned aerial vehicles (UAVs) and crowdsourcing (telephone images, tweets, etc.). Several satellites were launched in the last five years with high spatial resolution such as Sentinel-1A/B and Sentinel-2A within the European Copernicus program, and Landsat-8 within the U.S. Geological Survey (USGS) and the National Aeronautics and Space Administration. All of these data sets are free to access on operational basis.

Land use and land cover classification is a standard remote sensing task where each image pixel is either associated with a class label indicating the physical material of the surface(land cover) or each object describing the socio-economic function of the land(land use). Therefore, land use objects are complex structures consist of many different land cover elements. Due to its complex nature, both spectral and spatial features need to be incorporated for a successful land use/land cover mapping. Experiments to combine both of these features based on the Conditional Random Field (CRF) model, Markov Random Field model and Composite Kernel (CK) method have been carried out. Nevertheless, in most cases, the process of extracting extensive number of features for the intent of supervised classification is time consuming and requires comprehensive knowledge to extract useful features. In addition to that, hand-crafted methods that are used for classification mainly relies on low-level features and produce inadequate classification results. With the increasing amount of accessible data, application of deep learning for overcoming these challenges has become prominent. Compared to machine learning approaches such as Support Vector Machine (SVM) and Random Forest (RF) deep learning shows great promise with the use of big data. Current deep learning models are Deep Belief Net (DBN), Stacked Auto-Encoder (SAE), and Convolutional Neural Network's (CNN). Most well-known deep learning model (CNN) shows great progress for processing of remote sensing imagery. (CNN's) outperform shallow-structured machine learning tools in remote sensing applications such as object detection, segmentation and classification.

In this thesis, two pre-trained CNN models namely Inception-ResNet-V2 and Inception-v4 are used to classify scenes from satellite imagery. There are 20 classes with 700 images each such as airport, chaparral, dense residential, forest, freeway, golf course, ground track field, industrial area, intersection, meadow, medium residential, overpass, parking lot, rectangular farmland, river, runway, sparse residential, storage tank, tennis court and terrace. Scenes acquired from Worldview-3 satellite sensor are used to evaluate the performance of the network. Suggested networks reached %91.2 and %87.2 accuracy over the 1000 test image.



DERİN EVRİŞİMSEL SİNİR AĞLARI İLE ARAZİ KULLANIMI VE ARAZİ ÖRTÜSÜNÜN SINIFLANDIRILMASI

ÖZET

Son yıllarda, sinir ağlarının halefi olan derin öğrenme, özellikle bilgisayar görüşü, konuşma tanıma ve doğal dil işleme gibi alanlarda son teknoloji bir yaklaşım haline gelmiştir. Sinir ağları yüksek performanslı bilgi işlem, algoritmik iyileştirmeler ve büyük veriler gibi faktörler ile hayata geçirilen yerleşik bir yapay zeka dalıdır. Geçtiğimiz yıllarda büyük veri yapıları uzaktan algılama konusunda da büyük önem kazanmıştır. Uzaktan algılama, özellikle Dünya olmak üzere fiziksel temas kurmadan bir nesne veya fenomen hakkında bilgi edinmektir. Bu tanım, geleneksel uzaktan algılama alanlarını, örn. uydu ve hava fotoğrafçılığını kapsamakla birlikte, insansız hava araçları (İHA) ve kitle kaynak kullanımı (telefon görüntüleri, tweetler, vb.) alanlarını da içerir. Son yıllarda yüksek çözünürlüklü gözlem uydularının sayısı giderek artmıştır. Avrupa Kopernik programında geliştirilen Sentinel uyduları ve ABD Jeolojik Etütleri (USGS) ile Ulusal Havacılık ve Uzay İdaresi bünyesindeki Landsat uydularının elde ettiği verilerin hepsine operasyonel olarak erişim serbesttir. Elde edilen bu büyük verilerin incelenmesi ve analiz edilmesi uzaktan algılama konuları için önem arz etmektedir. Özellikle şehir planlama, tarım rekoltesi hesaplama, iklim değişikliğinin incelenmesi, arazi kullanımı ve arazi örtüsünün sınıflandırılması konularında kullanılır.

Uzaktan algılanmış verilerin yeryüzüne ait bilgiye dönüştürülmesinde kullanılan en önemli yöntemlerden biri görüntülerin sınıflandırılmasıdır. Sınıflandırma işlemi, benzer spektral özellikleri taşıyan nesnelerin gruplandırılmasıdır. Sınıflandırma işlemi için genellikle iki farklı yaklaşım kullanılır. Bu yaklaşımlar kontrollü ve kontrolsüz sınıflandırma olmak üzere ikiye ayrılır. Kontrollü sınıflandırma metodu eğitim veri seti kullanılarak sınıflandırmayı içerir. Bu yaklaşım ile daha yüksek doğruluklar elde edildiğinden en çok tercih edilen yöntemdir. Bunun yanı sıra geleneksel sınıflandırma yöntemleri olarak en çok benzerlik sınıflandırıcısı ve histogram eşitleme yöntemi örnek olarak verilebilir. Bu yöntemler el becerisi ile elde edilen özellikler içerdiğinden üzerinde çalışılmamış görüntüler ile iyi sonuç vermemekle birlikte sonuçların oluşturulması uzun zaman alabilir. Literatürde bugüne kadar uzaktan algılanmış görüntülerin sınıflandırılmasına yönelik daha karmaşık çeşitli algoritmalar geliştirilmiştir. Bu yöntemlerden bazıları destek vektör makineleri, karar ağaçları, markov rastgele alanı, koşullu rastgele alan ve bulanık mantık sınıflandırıcıdır. Fakat bu metotların büyük boyutlu eğitim verilerinden faydalanamadığı, kısıtlı eğitim verisi ile üzerinde çalışılmış görüntüler üzerinde etkili olduğu araştırmalarla ortaya konmuştur.

Arazi örtüsü ve arazi kullanımı sınıflandırması, her görüntü pikselinin ya yüzeyin fiziksel malzemesini (arazi örtüsü) veya sınıfın sosyo-ekonomik işlevini tanımlayan her bir nesneyi (arazi kullanımı) gösteren bir sınıf etiketi ile ilişkilendirildiği standart

bir uzaktan algılama problemidir. Bu nedenle, arazi kullanım nesnelere birçok farklı arazi örtüsü elemanından oluşan karmaşık yapılardır. Karmaşık doğası nedeniyle, başarılı bir arazi örtüsü arazi kullanımı haritalaması için hem spektral hem de mekansal özelliklerin dahil edilmesi gerekir. Derin öğrenme algoritmaları bu iki özelliği de kullanarak sınıflandırma yapabilmeleri açısından arazi örtüsü arazi kullanımı haritalaması için kullanılan en gelişmiş modellerdir. Artan erişilebilir veri miktarıyla birlikte, derin öğrenme uygulamaları öne çıkmıştır. Destek vektör makinesi ve karar ağacı gibi makine öğrenme yaklaşımlarıyla karşılaştırıldığında, derin öğrenme uygulamaları büyük verilerin kullanımı ile büyük umut vaat etmektedir. Mevcut derin öğrenme modellerinden Derin İnanç Ağları, Yığınlaşmış Otomatik Kodlayıcı ve Evrişimsel Sinir Ağları uzaktan algılama problemlerinde etkin olarak kullanılmaktadır. Görüntü sınıflandırmada en iyi bilinen derin öğrenme modeli olan Evrişimsel Sinir Ağları uzaktan algılama görüntülerinin işlenmesi için de büyük ilerleme göstermektedir. Evrişimsel Sinir Ağları nesne algılama, segmentasyon ve sınıflandırma gibi uzaktan algılama uygulamalarında sık yapıları makine öğrenme araçlarından daha iyi performans göstermektedir.

Bir derin öğrenme mimarisi olan Evrişimsel Sinir Ağları, özellikle görüntü sınıflandırmada kullanılır. Evrişimsel sinir ağları, eğitilebilen birçok katmandan oluşmaktadır. Çok katmanlı mimarisi sayesinde görüntülerden öznitelik çıkarma konusunda oldukça başarılıdır. Her katmanın kendine ait öznitelik havuzlama katmanı, filtre banka katmanı ve doğrusal olmayan katmanı bulunmaktadır. Filtre banka katmanı farklı öznitelikler çıkarılması için birçok çekirdek bulundurur. Havuzlama katmanında elde edilen öznitelik haritaları tek tek ele alınır. Her harita maksimum değerinin veya komşu değerinin ortalamasının elde edilmesini sağlamaktadır. Görüntü önce parçalara ayrılır ve her parçaya filtre uygulanır. Filtre işleminden sonra görüntüde küçülme meydana gelir. Bu işlem sonucunda elde edilen pikseller anlamlandırılarak sınıflandırma problemi çözülmeye çalışılır.

Evrişimsel Sinir Ağları mimarileri giderek daha karmaşık ve derin bir yapıya evrilmiştir. Yann Lecun tarafından geliştirilen LeNet modern anlamda görüntü işlemede kullanılan ilk derin mimariye sahip evrişimsel sinir ağıdır. Akabinde 2012 yılındaki ILSVRC (ImageNet Large-Scale Visual Recognition Challenge) yarışmasında Alex Krizhevsky tarafından geliştirilen AlexNet görüntü sınıflandırma ve tanıma alanında büyük bir başarı sağlamıştır. Bu başarının ardından Evrişimsel Sinir Ağları görüntü işlemede sıkça kullanılmaya başlamıştır. İzleyen yıllarda ILSVRC yarışmasında önde gelen mimariler VGGNet, ResNet ve GoogleNet olmuştur.

Oldukça derin mimariye sahip modern evrişimsel sinir ağlarını spesifik bir sınıflandırma problemi için sıfırdan eğitmek uzun hesaplamalar gerektirir. Fakat çok katmanlı yapısı sayesinde farklı verilerle eğitilmiş ağlar başka bir sınıflandırma problemi için kullanılabilir. Nesnelere oluşturduğu çizgiler ve köşeler gibi kavramlar ağların aşağı katmanlarında öğrenilir. Yukarı katmanları ise yeniden eğitilerek istenilen sınıflandırma problemine uyarlanır. Böylece etiketli veri bulmanın zahmetli olduğu alanlarda başarılı sonuçlar elde edilebilir. Günümüzde açık kaynak olarak kullanılabilen ağlar ImageNet veriseti ile eğitilmiş olup bir çok farklı alanda kullanılmak üzere ince ayar yapılabilir. Uzaktan algılama problemlerinde literatüre bakıldığında önceden eğitilmiş ağların nesne tanıma ve sınıflandırma gibi konularda başarılı sonuçlar verdiği görülmüştür.

Bu tez çalışmasında, Inception-ResNet-V2 ve Inception-v4 adlı iki önceden eğitilmiş Evrişimsel Sinir Ağı modeli, uydu görüntülerini sınıflandırmak için kullanılmıştır. Sınıflar havaalanı, yoğun yerleşim alanı, orman, çevre yolu, golf sahası, arazi yolu

alanı, sanayi bölgesi, kavşak, çayır, orta ölçekli yerleşim alanı, üst geçit, otopark, dikdörtgen tarım arazileri, nehir, pist, seyrek yerleşim alanı, depolama tankı, tenis kortu ve teras olmak üzere 20 adettir. Eğitim verisi olarak her sınıf için 700 görüntü kullanılmıştır. Worldview-3 uydu sensöründen elde edilen sahneler ağın performansını değerlendirmek için test seti olarak kullanılmıştır. Önerilen ağlar, 1000 test görüntüsünde %91.2 ve %87.2 doğruluğa ulaşmıştır.





1. INTRODUCTION

Recent years in remote sensing field can be named as the era of big data. Volume and the availability of the remote sensing data has increased immensely. Due to large scale of these data sets new challenges have risen. Land use land cover classification has always been an important task in remote sensing field, providing crucial information for applications such as urban planning and precision agriculture. This can be done by analyzing remote sensing imagery.

The task of classification in the context of remote sensing imagery is utilizing labeled samples to determine which class does each pixel belong to. There has been wide range of studies on analyzing each individual pixel of the images and classify them based on their spectrum. In this framework, mainly used approaches are support vector machines(SVMs)[1] and decision trees[2]. However, these techniques are not effective in a large-scale due to the fact that majority of satellite imagery does not use high spectral resolution sensors. Without understanding the shape of the objects, separating classes entirely by their spectrum is difficult. On the other hand, more advanced techniques incorporate information from a several neighboring pixels to boost the classifiers' performance, specified as spectral-spatial classification. In this context studies based on the Conditional Random Field (CRF)[3] model, Markov Random Field[4] model and Composite Kernel (CK)[5] method have been carried out. However these methods only show promise with the data being analyzed. Generalization of the proposed solutions are questionable. Also the process of extracting extensive number of features for the intent of supervised classification is time consuming and requires comprehensive knowledge to extract useful features. On the other hand, deep learning approaches learn from the data itself, thus replacing the expertise of feature engineering. Deep learning models outperform shallow-structured approaches in remote sensing applications such as object detection, segmentation and classification. Convolutional neural networks (CNNs) are accepted deep learning models that extract contextual image features by utilizing stack of learned convolution filters. Inspired by the human visual cortex CNNs consist of multiple layers. First part of the CNN is

usually referred as feature extractor and last part is called as multilayer perceptron(MLP). Final layer assigns class labels and compute probabilities of a given class. Other layers are mostly convolutional filters. For instance, to analyze grayscale imagery, CNNs utilize two dimensional (2-D) convolutional filters and as for red-green-blue RGB imagery (3-D) convolutional filters are used. After training phase, filters learn to elicit hierarchical features straight from the input data, in contrast to machine learning models that use “hand-crafted” features. Architecture and the details of the CNN models will be discussed in the chapters later on.

CNNs prevailed in tasks such as classification and object detection. Beginning in the 1990’s, LeCun et al[6]. designed LeNet for reading zip codes. It created an impact in the image processing community. Krizhevsky et al.[7] created a deep CNN which won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. Next year Zeiler and Fergus[8] developed ZFNet, which was based on AlexNet architecture with tweaked parameters and won (ILSVRC). In 2014, Szegedy et al[9] created GoogLeNet also known as (Inception-v1) which had only 4 million parameters compared to AlexNet(60 million) won (ILSVRC). ResNet designed by He et al.[10] introduced residual connections which improved the training speed remarkably. In 2016, Szegedy et al introduced residual connections in conjunction of Inception network called as Inception-ResNet which significantly improved recognition performance[11].

Transfer learning is the practice of reusing a trained algorithm on a comparable dataset. As humans we don’t learn to recognize new images by analyzing thousands of similar ones. Concepts of lines and curves comes first. Idea behind transfer learning is quite similar. Transferring low-level features from a pre-trained model and tune the filter weights to identify other different patterns. Thus, eliminating the heavy work of training from the scratch with thousands of images.

In the remote sensing field, Marmanis et al[12] utilized this concept by using a CNN pre-trained on the ImageNet dataset and extracted features of orthoimagery from the last layer. Donahue et al.[13] displayed that the crucial information is obtained from the deeper layers. With a similar approach Salberg[14] detected seal pups in aerial imagery with high accuracy. Other transfer learning applications of remote sensing are as follows; Othman et al[15] trained ILSVRC-12 challenge data set and used transfer learning on UC Merced Land Use[16] dataset. Iftene et al.[17] used ImageNet data set on CaffeNet and GoogLeNet models then applied results to WHU-RS[18] data set

consist of very high resolution imagery. Ghazi et al.[19] and Lee et al.[20] used a combination of pre-trained networks such as GoogLeNet, VGGNet and AlexNet on plant identification.

1.1 Purpose of Thesis

In recent years expansion in the satellite industry led to a vast amount of available satellite imagery of the earth. These imagery provide critical information for many of remote sensing tasks including land cover and land use analysis. Traditional methods for the classification of image scenes are no longer applicable for the analysis of big data. These methods often rely on handcrafted features and therefore on the feature engineer. Also it takes significant amount of time to create features that can be generalized over the unseen data.

One of the subcategories of machine learning discipline, deep learning became a hot topic for computer vision and remote sensing field. Stacked auto-encoders (SAEs) deep Boltzmann machines (DBMs), deep belief networks (DBNs), and convolutional neural networks (CNNs) are the most popular deep neural network architectures used in remote sensing applications. These networks can extract reliable features directly from data without any need for feature engineering. Research shows that CNNs are the most capable feature extractors for classification problems. Although the most popular networks are designed and trained to recognize daily internet images, these networks are also capable of recognizing geospatial objects, land cover and land use classes in the satellite images.

Purpose of this thesis is to investigate state-of-the-art CNN models for land cover and land use classification and produce accurate results that can be generalized over the unseen data.

1.2 Scope and the Organization of the Thesis

The thesis is organized as follows: A detailed literature overview on land cover land use classification with both conventional techniques and machine learning methods is discussed in Chapter 2. Theory and details of deep learning as well as popular deep networks are presented in Chapter 3. The experiments conducted on scene classification are given in Chapter 4 and conclusions are presented in Chapter 5.



2. LITERATURE REVIEW ON LAND COVER AND LAND USE CLASSIFICATION

Land cover and land use information is a crucial aspect of remote sensing. Information derived from remote sensing imagery is fundamental to numerous environmental and socio-economic applications such as urban and regional planning and natural resource management. Beginning in the 1980's, various methods have been developed to generate information from remote sensing imagery. Processes of classification and image interpretation have been introduced. Between 1980's and 1990's, almost all classification methods used image pixel as a primary unit, labeling each pixel with a single land cover and land use class. However, pixel based classification methods brought challenges as the pixel may contain more than one land cover land use type. Thus, in late 1990's object based classification methods have been developed. This method groups several pixels with homogeneous attributes into an object and each object is then considered as the basic unit rather than pixels. As the number of very high resolution sensors (i.e. IKONOS, Quick bird) increased, images started to have more intra-class spectral variability. This resulted in unsatisfactory results with classifiers that mainly utilize spectral variables. Therefore, spatial component of the image also needed to be used. Term "spatio-contextual" image classification is then addressed to describe the relationship between target pixel and its neighboring pixels.

2.1 Pixel-Based Image Classification

Pixel-based approaches assume that each pixel belongs to single land cover and land use type [21]. Pixel based classifiers can be grouped as supervised and unsupervised classifiers. Unsupervised classifiers divide remote sensing imagery into a number of classes based on their pixel values without using any training data. K-means algorithm [22] and Iterative Self-Organizing Data Analysis (ISODATA) technique are examples of widely used unsupervised classifiers. On the other hand, for supervised classification, image analyst has to select training samples and compare those samples to the spectral properties of the target image. Then, analyst labels pixels to the

appropriate class type according to decision rules. Maximum Likelihood Classifier (MLC) [23], Minimum Distance-to-Means Classifier [24], K-Nearest Neighbors Classifier [25] are commonly used supervised classifiers.

2.2 Sub-Pixel Based Image Classification

Assumption of each pixel belonging to a certain class is often leads to poor performance in classification accuracy with medium resolution imagery. As a better alternative, sub-pixel based approach gives each pixel partial memberships to all classes so that the corresponding areal distribution of each class can be predicted respectively. Major sub-pixel based models are fuzzy classification [26], regression modeling [27] and spectral mixture analysis [28].

2.3 Object-Based Image Classification

In comparison to pixel and sub-pixel based approaches object-based models consider the objects as the basic unit of analysis. Objects are comprised of several individual pixels that have homogeneous attributes. These image objects are generated with a process addressed as image segmentation. With image segmentation objects are formed using spatial, contextual and spectral information. The differences between pixel, sub-pixel and object based approaches are given in (Table 2.1).

Table 2.1 : The differences between pixel, sub-pixel and object based approaches.

Classification of Techniques	Attributes	Examples of Classifiers
Pixel-based Techniques	Each pixel is labeled as a single land use land cover type.	Unsupervised (e.g. k-means, ISODATA) Supervised (e.g. Maximum likelihood)
Sub-pixel based Techniques	Each pixel is considered mixed, and the areal distribution of each class is predicted.	Fuzzy classification, spectral mixture analysis, regression modeling
Object-based Techniques	Objects, instead of individual pixels, are considered as the basic unit.	E-cognition, ArcGIS Feature Analyst

2.4 Spatio-Contextual Image Classification

Spectral classifiers have advantages in terms of simplicity and computational load. However not all land cover land use types can be classified using spectral information [29]. In order to overcome these challenges, spatial and contextual information has to be utilized as well. Markov Random Field (MRF) model [30] is one of the spatio-contextual remote sensing image analysis techniques. MRF is a graphical model that has been applied in a wide range of fields from computer vision to physics. MRF's can be used to analyze the local and global properties of a remote sensing imagery, and evaluate the spatial autocorrelation between pixels through mathematical means. Increasing number of studies shows that MRF-based classification methods produce substantial results compared to conventional non-contextual classifiers [31]. Variety of MRF-based classification techniques have been used in land cover and land use classifications and showed promising results, however according to many remote sensing scientists, the concepts of MRF are considered cumbersome and their implementations include challenging computational difficulties.

Traditional classification methods require high level of expertise and usually work well with the data being analyzed but produces poor results with the unseen data [32]. Thus, the generalization of the extracted features is questionable. However, with machine learning approaches, contribution of the image analyst is reduced as the features are extracted directly from the data itself. The classification problem with machine learning approaches is discussed in the next section.

2.5 Machine Learning in Remote Sensing

Machine learning is one of many sub-fields of artificial intelligence (AI) and has become very well-known in the last decade. Although AI has other sub-fields aside from machine learning, the two are used interchangeably. Machine learning systems are created by analyzing lots of examples and devise rules to predict outcomes for unseen data(Figure 2.1). Machine-learning technology has laid the foundation of numerous developments of modern society such as tailoring advertisements, relevant web searches and content filtering on social media. It is getting more and more available in consumer products such as smart phones and cameras. Machine learning systems can be classified under three sub-categories; systems that depend on human

supervision or not (supervised, unsupervised), systems that learn incrementally as they go (online learning) and systems work solely by comparing data points to newer ones (instance-based learning).

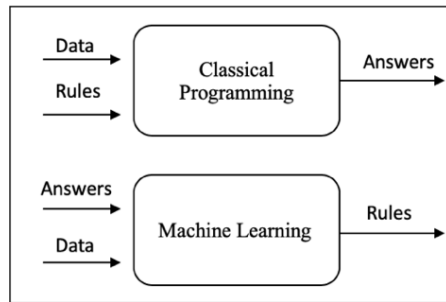


Figure 2.1 : Difference between classical programming and machine learning.

2.5.1 Supervised learning

Machine learning systems may require supervision to a certain extent. Idea is to feed labeled samples in order to generate features from the given data. Regression and classification are two major tasks that require supervision (Figure 2.2). Regression stands for predicting a target numeric value based on set of features and it is measured by root mean square error. However, in classification goal is to predict a label. Therefore classification is measured by accuracy. In the field of remote sensing, Support Vector Machines (SVM), Decision Trees (DT), Random Forests (RF) and Neural Networks (NN) are well established supervised algorithms that are used for classification problem.

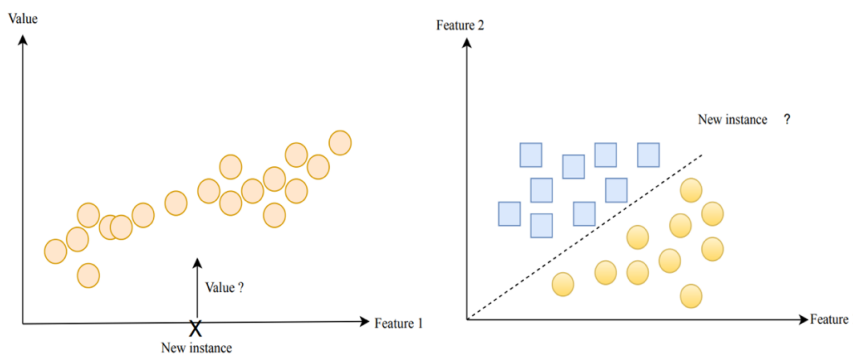
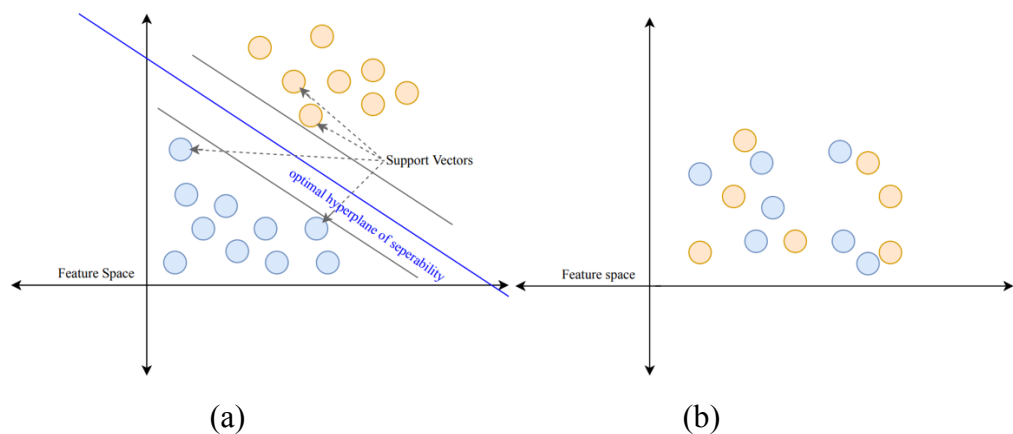


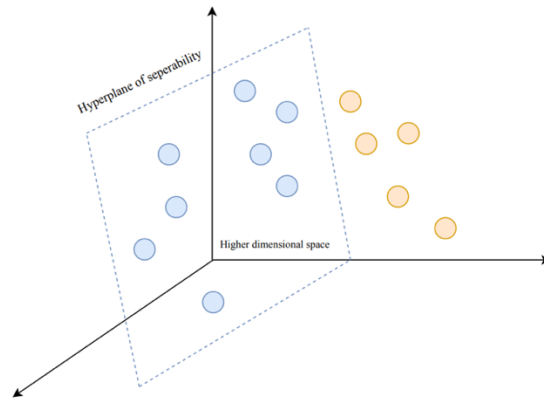
Figure 2.2 : Difference between classification and regression.

2.5.2 Support vector machines

For parametric classification, goal is to analyze feature space values and their distribution of each class. On the contrary, SVM focuses solely on the training samples

and the optimal boundary between classes. However, not all the available training samples can be used to describe and specify the separating hyperplane. The optimal hyperplane is determined by a subset of feature vectors that lie on the margin named as support vectors. Main objective in SVM is to find the optimal boundary, which increases the margin, or separation between the support vectors (Figure 2.3a). When the separability is nonlinear (Figure 2.3b), a nonlinear transform can be made to a newer space with greater dimension in order to achieve linear separability (Figure 2.3c). This operation is called as kernel trick. For this operation transform function is not required. Merely kernel function k is needed. However, choosing the right kernel function presents challenges in terms of optimal results. Studies show that different kernels such as polynomial and radial basis function applied on SVM-based classification produced different results in satellite images [33]. For classes that are not separable, parameter value C is specified by the user to create a soft margin for the decision boundary. Higher C values often lead to poor results in terms of algorithms ability generalize. Also, SVM classifier is naturally binary, therefore it can only identify a single boundary between two classes. This problem can be tackled by applying classifier to each possible combination of classes. By doing so, computational time is expected to increase exponentially as the number of classes increase. Additionally, SVMs are highly affected by noisy data; which are commonly encountered in remotely sensed imagery.





(c)

Figure 2.3 : Non-linear transform and optimal hyperplane for SVM.

There are numerous studies regarding the use of SVM's in the field of remote sensing especially for the problem of land cover and land use classification. Huang et al. [34] used Landsat TM and Landsat ETM+ images to detect forest cover change. The classification is carried out using SVM and produced approximately 90% accuracy. Li et al. [35] proposed an SVM-based classifier using high resolution imagery from the QuickBird sensor. A scene segmentation algorithm was incorporated with the SVM object classifier yielded better results. It is also shown that the SVM classifier is highly reliant on the segmentation process, a typical disadvantage of object-based classifiers. Another study carried out by Brenning, [36] used eleven different classifiers to detect rock glacier using Landsat and SRTM. SVM-based method did not show promising results compared to other methods. In conclusion, SVM classifiers can show decent results with limited amount of data due to support vector concept relies on small number of data points to define a classifier's hyperplane. However, selection of parameters and kernel functions present challenges that often lead to "trial and error" approach.

2.5.3 Decision tree classifier

Conventional classifiers employ neural and statistical approaches to the classification problem. All available features are used to assign each pixel to an appropriate class. However, DT uses a sequential approach for label assignment. Chain of simple decisions is made based on the results of sequential tests instead of one complex decision. The data can be split depending on the threshold value. Iteration through

nodes is decided depending on the value of a certain band is above or below of the threshold value. Thus, the model logic can be described as a set of if-then rules given in (Figure 2.4). Once the model is constructed, classification is swift due to no further complex mathematics is needed. Decision tree classification methods have been used successfully for a wide range of classification problems including the remote sensing field. Otukei and Blaschke [37] compared support vector machine, maximum likelihood and decision tree based techniques for the assessment of land cover change using Landsat TM and ETM+ data and found decision tree based methods produced the best results. Punia et al. [38] classified IRS-P6 AWiFS data using decision tree classifier and obtained very high accuracy. Challenges with DT's include over fitting and the possibility of generating a non-optimal solution. The former problem can be tackled by a process called as pruning the tree which means removing one or more layers of splits (i.e. branches). However, according to Pal and Mather [39] pruning reduces the accuracy of classifying the training data but often increases the accuracy of unseen data. Also, they've reported that when hyperspectral data are used, the performance of DT classifiers declines as the number of features increases.

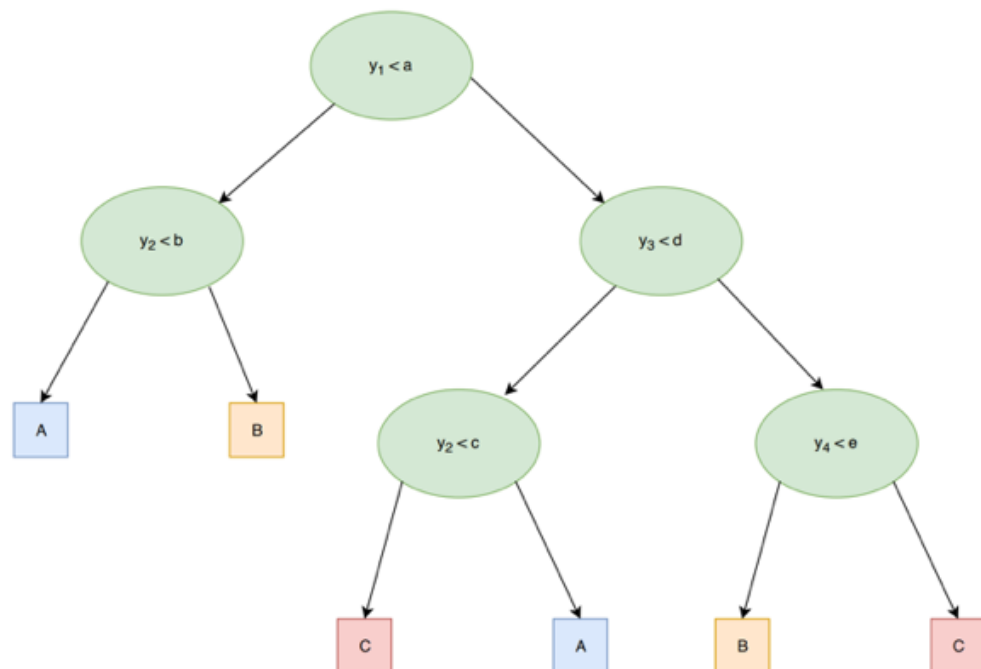


Figure 2.4 : Sample decision tree.

2.5.4 Random forest classifier

Random forest is comprised of many DT's in order to achieve better results than a single DT. A method of "voting" is applied to all trees to obtain the label for each instance. Due to difference of error correlation for each tree final result is more accurate. This idea can be extended to each tree having own subset of training data thus, minimizing correlation and making the ensemble more reliable. This technique is increasingly being applied in the field of remote sensing especially in land-cover classification using multispectral and hyperspectral satellite sensor imagery[40] [41] [42]. However, most studies that have used random forests have few land-cover classes and focused on small study areas [43][44]. Lawrence and Moran [45] compared the performance of a variety of machine-learning classification algorithms, using 30 different data sets. They have reported that RF had the highest average classification accuracy of 73.19%.

2.5.5 Artificial neural networks

Inspired by the human brain, concept of ANN's were first introduced back in 1940's by a neurophysiologist Warren McCulloch and a mathematician Walter Pitts. They have mathematically modeled biological neurons to perform intricate computations. A neuron is essentially an input/output device transmitting binary coded information. In 1957, Frank Rosenblatt introduced perceptrons as the foundation of modern ANN architectures. Instead of binary values, perceptrons use numbers as input and output. It is based on linear threshold unit (LTU) which computes weighted (w) sum of inputs (x) and utilizes a step function to that sum and outputs the result, given in equation 2.1, equation 2.2 and (Figure 2.5);

$$z = w_1x_1 + w_2x_2 + \dots + w_nx_n = w^t \cdot x \text{ (sum of the inputs),} \quad (2.1)$$

$$h_w(x) = \text{step}(z) = \text{step}(w^t \cdot x) \text{ (step function of the sum and the output)} \quad (2.2)$$

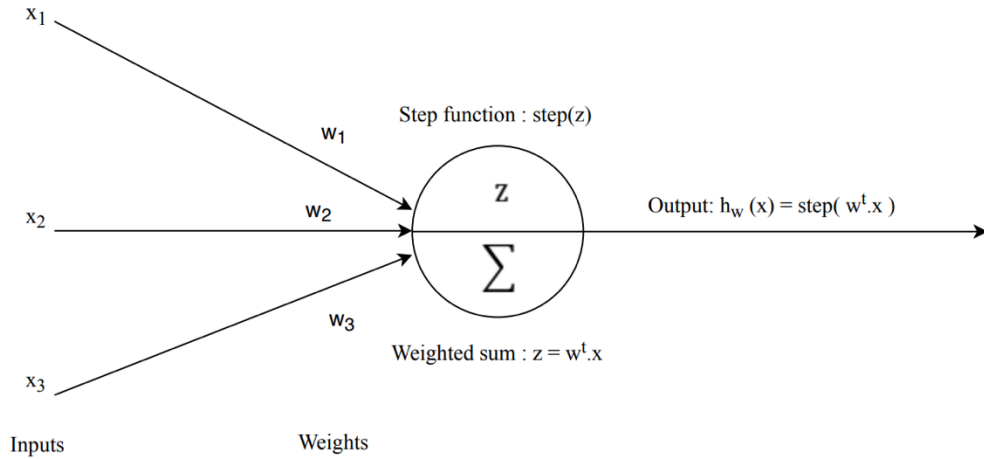


Figure 2.5 : Linear threshold unit.

For a simple linear binary classification problem, a single LTU can be utilized. Combination of inputs is computed and depending on the threshold value output can be a positive or a negative class similar to a linear-SVM. Appropriate values for weights w_1 , w_2 and w_3 are calculated by training the algorithm. Perceptrons are trained simply by reinforcing connection weights that lead to correct output. Each training instance is fed through the network and a prediction is made. For every output neuron that contributed to the right prediction weights are updated;

$$w_{i,j}^{k+1} = w_{i,j}^k + \lambda(\hat{y}_j - y_j)x_i, \quad (2.3)$$

where;

$w_{i,j}$ is the connection weight between i^{th} input neuron and the j^{th} output neuron,

k is the step number,

\hat{y}_j is the output of the j^{th} output neuron of the ongoing training instance,

y_j is the target output of the j^{th} output neuron for the ongoing training instance,

x_i is the value of i^{th} input of ongoing training instance,

λ is the learning rate.

Although perceptrons showed great promise, they failed in XOR classification problem. This occurs when network tries to predict XOR logic gates given two binary inputs. An XOR function needs to return false if two inputs are equal and true if they are different. However this problem is tackled by stacking multiple perceptrons.

Multi-Layer Perceptrons (MLP) are consist of one input layer, multiple LTU's referred as hidden layers and one layer of LTU's addressed as output layer. Input layer and hidden layers have a fully connected bias neuron (Figure 2.6). MLP is trained with an algorithm called back propagation. First, algorithm feeds each training instance to the network and output of each neuron is computed. This process is called as forward pass. After forward pass, output error of the network is calculated. Then algorithm tracks error contributions of each neuron until it reaches to the input layer. By propagating backwards in the network, error gradient of all connection weights are effectively measured so that tweaks can be made on the weights. This final step of the back propagation is referred as Gradient Descent. MLP's used in variety of remote sensing challenges including land cover and land use classification [46][47].

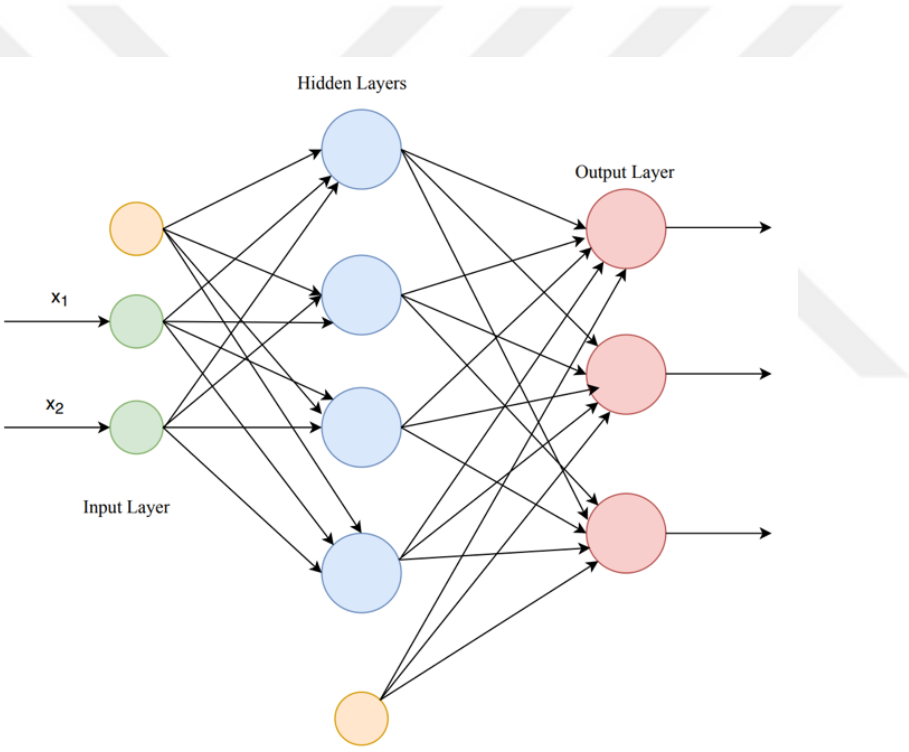


Figure 2.6 : Multi-Layer Perceptron.

3. DEEP NEURAL NETWORKS

In recent years, deep learning is one of the fastest growing areas of research. There are numerous implementations in fields such as computer vision, speech recognition, language processing and remote sensing. As the successor of neural networks, deep learning models share the fundamental concepts with NN's, however to be called deep, network has to have more than two hidden layers. As the number of hidden layers increase, higher-level of features can be extracted. However, to make use of this deep architecture number of training samples has to increase. Thus, computational cost of training a deep network from scratch is too high and could take several months. Although, recent advances in big data and GPU technology has helped deep learning approaches to be more practical.

Multi-layered architecture of deep networks can extract efficient features from raw data without the need of significant feature engineering. Thus, deep learning models became the state-of-the-art when it comes to classification and object detection. Various studies have been conducted in remote sensing applications using deep learning based models [48][49][50][51][52][53][54][55]. As of today, there are four major deep learning architectures. These are the deep belief networks (DBNs), and recurrent neural networks (RNN) autoencoders (AE) and convolutional neural networks (CNN). Following sections discuss these architectures in detail.

3.1 Autoencoders

Autoencoder is an unsupervised neural network meaning that it can extract features from unlabeled data. Autoencoders obtain compact representations of the input data referred as codings. Codings are reduced in dimension compared to input data, making autoencoders a dimensionality reduction tool which is needed for many remote sensing applications. They also can be used as generating new data from the training data. This is called a generative model. Autoencoders comprised of two parts; an encoder and a decoder (Figure 3.1). Encoder also referred as the recognition network, transforms input data to latent representations. Decoder, referred as the generative network

generates outputs from these representations. Autoencoder architecture is very similar to MLP's but the number of inputs is equal to number of neurons in the output layer. Outputs are also called as reconstructions and when outputs are different from the inputs, reconstruction loss is employed to tune the model. An encoder with one hidden layer can be represented as;

$$h = f(x) = g(Wx + \beta), \tag{3.1}$$

$$x' = (W'x + \beta'), \tag{3.2}$$

$$\mathcal{L} = f'[f(x)], \tag{3.3}$$

where f is the encoder function, h is the latent representation or code, g is the decoder function that maps the output x' , W' is decoding weight, β' is the decoding bias and \mathcal{L} is the loss function of x and x' .

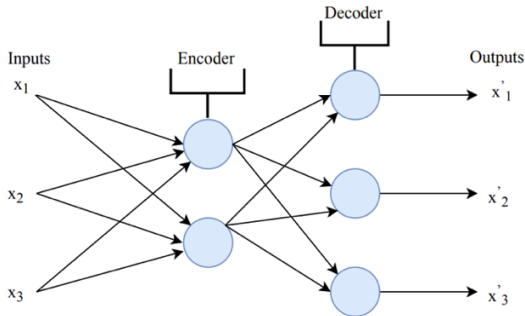


Figure 3.1 : Autoencoder.

3.1.1 Stacked autoencoders

Autoencoders that have multiple hidden layers are called stacked autoencoders or deep autoencoders given in (Figure 3.2). By adding more layers autoencoder can learn intricate codings. Stacked autoencoders are especially used for image analysis in remote sensing [56][57].

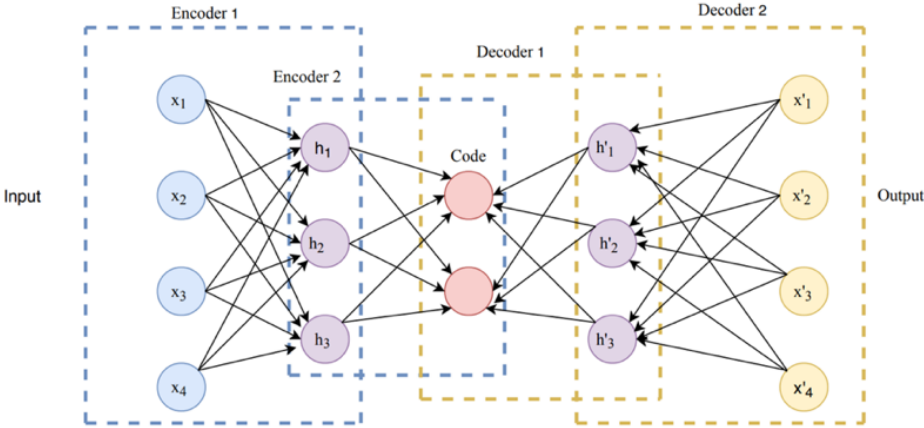


Figure 3.2 : Stacked Autoencoder.

3.1.2 Sparse autoencoders

For more complex structures, sparsity constraint can be added to the hidden units leading to better feature extraction. This reduces the number of active neurons in the coding layer and forces autoencoder to represent each input with less activation. Mean average activity of each neuron in the coding layer is calculated and neurons that are active above mean average are penalized by adding sparsity loss. General approach is to use Kullback–Leibler divergence. The following equations show the divergence between two discrete distributions P and Q ;

$$D_{KL} = (P | Q) = \sum P (i) \log \frac{P(i)}{Q(i)}, \quad (3.4)$$

$$D_{KL} (p|q) = p \log \frac{p}{q} + (1 - p) \log \frac{1-p}{1-q}, \quad (3.5)$$

where p is the target average activation value and q is the mean activation of all neurons. The equations show the divergence between target sparsity p and the actual sparsity q.

3.2 Deep Belief Networks and Restricted Boltzmann Machine

Restricted Boltzmann machine (RBM) is an undirected graphical model comprised of a hidden layer and a visible layer. In contrast to autoencoders or sparse autoencoders, there are no connections in hidden or visible layers. Energy function of the RBM is given by

$$\mathbb{E}(v, h) = -\sum a_i v_i - \sum b_j h_j - \sum_{i,j} v_i h_j w_{i,j}, \quad (3.6)$$

where $w_{i,j}$ is the weight between visible unit i and hidden unit j , a_i, b_j are their biases and v_i, h_j shows the states of i and j .

Two or more RBM's are stacked together forming DBN (Figure 3.3). DBNs have been used successfully in remote sensing problems such as scene classification[58], object recognition[59] and change detection[60].

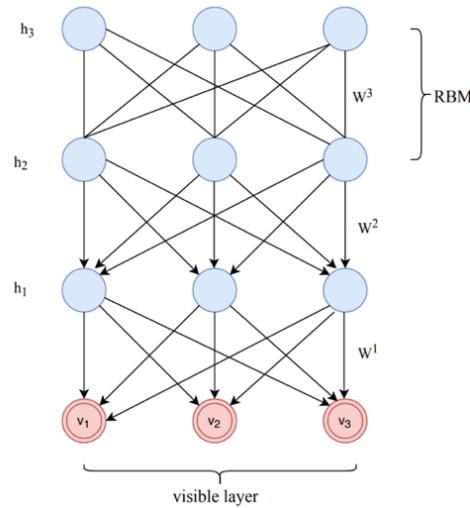


Figure 3.3 : A DBN with two RBM's.

3.3 Recurrent Neural Networks

In contrast to feed-forward networks discussed earlier on, RNN's have connections pointing backward. RNN's have a recurrent hidden state that activates at each step depending on the previous step (Figure 3.4). Each recurrent neuron has one weight for the inputs $x(t)$ and another for the outputs of the previous time step, $y(t-1)$. If we represent these weights with W_x and W_y , output of a single recurrent neuron can be shown as in equation 3.7.

$$y_{(t)} = \Phi (x_{(t)}^T \cdot W_x + y_{(t-1)}^T \cdot W_y + b), \quad (3.7)$$

where Φ the activation function and b is is the bias term. Output of a recurrent neuron is a function of all the inputs from previous time steps. A single recurrent neuron, or a layer of recurrent neurons can be called as a basic memory cell.

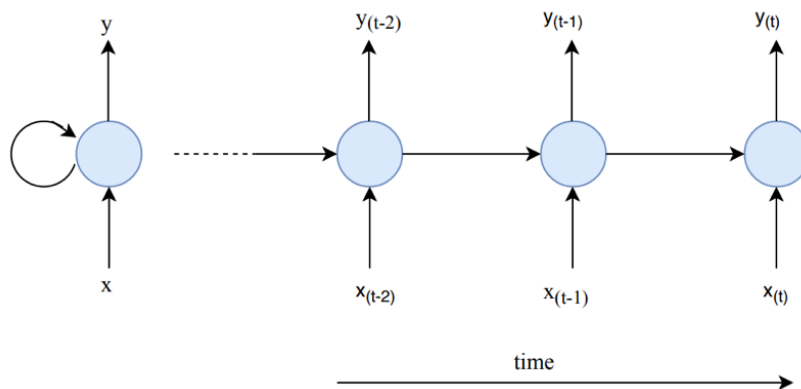


Figure 3.4 : Backward connection of RNN.

As the network propagates, weights are applied on top of itself and causing memory inputs to gradually fade away. When traversing an RNN, data goes through transformations and after each step some part of the information is lost. Subsequently, the RNN's state does not contain any of the first inputs. To address this issue, Sepp Hochreiter and Jürgen Schmidhuber introduced LSTM (Long Short-Term Memory) cell [61]. LSTM is a recurrent cell and its state is described with two vectors $c_{(t)}$ and $h_{(t)}$. Main idea is that the network needs to learn which memories to store and which memories to throw away. As shown in (Figure 3.5), the long-term state $c_{(t-1)}$ traverses the network and goes through the forget gate, drops some memories, and then it adds some new memories that were selected by an input gate. Following the addition operation, the long-term state is fed to the tanh function, and the result is filtered by the output gate. Remaining layers are gate controllers. Gate controllers use the sigmoid logistic activation function therefore outputs range from 0 to 1. Their outputs are fed to element-wise multiplication operations, so if they output 0's, gate is closed, and if they output 1's, gate is opened. Each gate serves different purposes. Forget gate controlled by $f_{(t)}$ determines which elements of the long-term state are to be removed. Input gate controlled by $i_{(t)}$ determines which elements of $g_{(t)}$ should be added to the long-term state. Finally, the output gate controlled by $o_{(t)}$ determines which elements of the long-term state should be read.

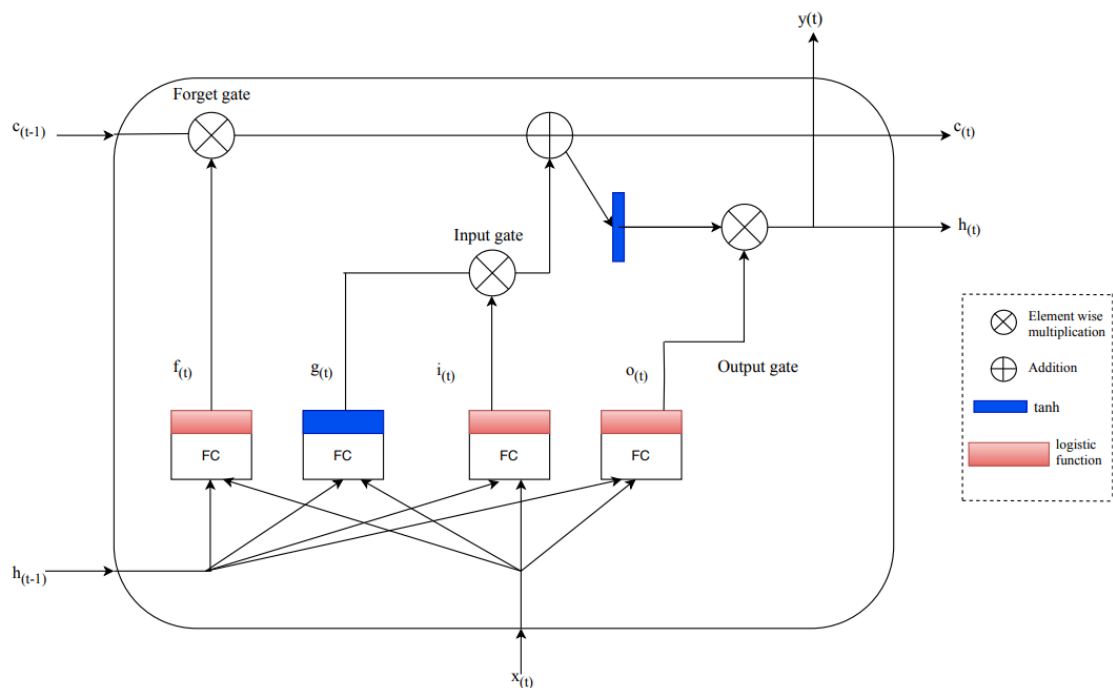


Figure 3.5 : LSTM cell.

Computations of LSTM elements (Figure 3.5) are given below,

$$i_{(t)} = \sigma(W_{xi}^T \cdot x_{(t)} + W_{hi}^T \cdot h_{(t-1)} + b_i) \quad (3.8)$$

$$f_{(t)} = \sigma(W_{xf}^T \cdot x_{(t)} + W_{hf}^T \cdot h_{(t-1)} + b_f) \quad (3.9)$$

$$o_{(t)} = \sigma(W_{xo}^T \cdot x_{(t)} + W_{ho}^T \cdot h_{(t-1)} + b_o) \quad (3.10)$$

$$g_{(t)} = \tanh(W_{xg}^T \cdot x_{(t)} + W_{hg}^T \cdot h_{(t-1)} + b_g) \quad (3.11)$$

$$c_{(t)} = f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)} \quad (3.12)$$

$$y_{(t)} = h_{(t)} = o_{(t)} \otimes \tanh(c_{(t)}) \quad (3.13)$$

RNN's are best suitable for sequential data like time series. RNN's are generally used for stock price prediction and natural language processing. In remote sensing, it is used for hyperspectral and multi-temporal image classification [61][62].

3.4 Convolutional Neural Networks

Convolutional neural networks (CNNs) are developed from studying brain's visual cortex, and they have been implemented in various image recognition applications such as autonomous cars, image search services, automatic video classification systems. Furthermore, CNNs are not limited to visual tasks; they also show promise at natural language processing and voice recognition. Recent advances in computational power and the amount of available training data, CNNs have become a hot topic in the deep learning community. LeNet-5 architecture designed by LeCun et al. has laid the foundation of many different architectures we use today. It has been used to recognize handwritten numbers.

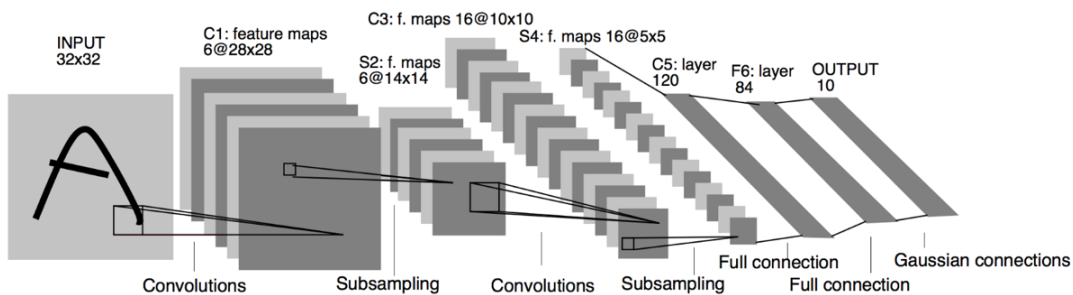


Figure 3.6 : LeNet architecture.

Convolution neural networks share the fundamental operations with any other neural network such as receiving an input, making dot productions and follow that up with a non-linear activation function in order to learn weights and biases. When it comes to classifying images fully-connected networks(MLP's) present challenges. In order to classify an image with an input size of 64x64x3 fully connected layers need 12288 weights in the first hidden layer. Parameters will increase as the input size gets bigger. Networks having large number of parameters likely to train slower and chances of overfitting are increased. CNN's exploit input images by localizing the reception of features. These features in image are spatially close to each other and non-dynamic. This process capitalize on the spatially-local correlated neighboring fields addressed as receptive fields by implementing a local connectivity pattern between neurons of adjacent layers shown in (Figure 3.7). Furthermore, except for their receptive fields, all neurons of a layer are identical to one another. Thus, they share the same weights. This reduces the number of weights to be learned, leading to reduced number of parameters, lower computational cost and lesser amount of training data required to train the neural network. CNN is composed of three different layers referred as convolutional layer, pooling layer and the fully connected layer.

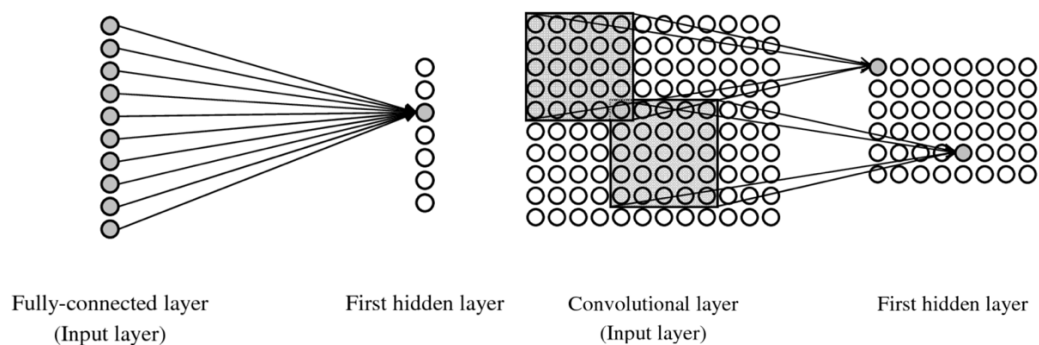


Figure 3.7 : Comparison of input layers: Fully connected layer vs convolutional layer. Size of the local receptive field is 5x5.

3.4.1 Convolutional layer

Convolutional layer is the most crucial aspect of the CNN. In the first convolutional layer, each neuron is connected to the pixels in their receptive fields (also referred as convolutional filter or kernel) of the input image. In the second convolutional layer, each neuron is connected to neurons located within a small rectangle in the first layer.

By using this architecture, network focuses on low-level features in the first hidden layer, then compiles them into higher-level features in the following hidden layers. In real world images, this hierarchical structure is prevalent, which is why CNNs are accurate in image recognition tasks. Convolution of the input image matrix and the filter matrix gives the feature map (Figure 3.8). When the filter does not perfectly fit the input image, padding is used. In order to fit the input image, image matrix is padded with zeros(zero-padding). Input image matrix is often times larger than the filter matrix. Therefore, filter has to be shifted over the image matrix. Number of pixels shifts over the input matrix is called the stride. Convolution of an image with different filters can perform operations such as blur, edge detection and sharpening.

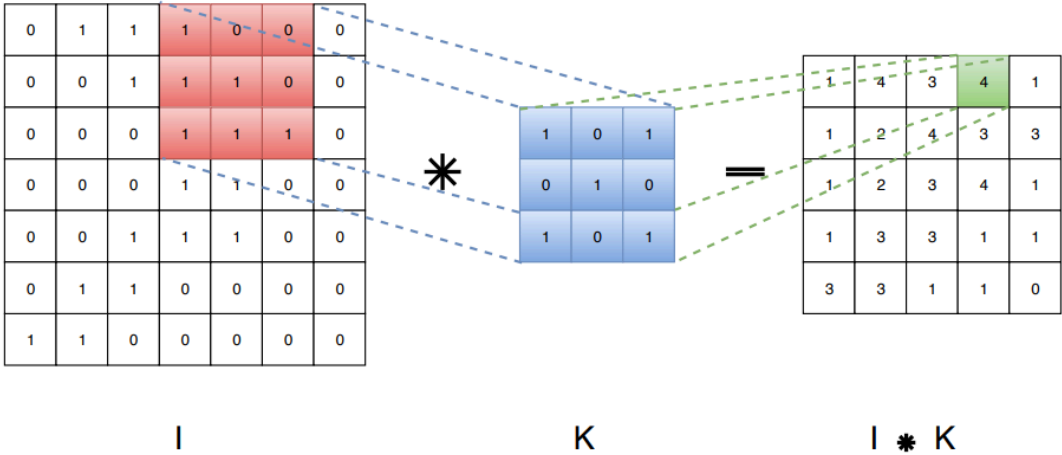


Figure 3.8 : 2-dimensional convolutional example with filter size 3x3 and stride 1 with zero padding.

After each convolution layer, an activation function adds non-linearity to the model and decides which neuron will be fired. There are various activation functions (Figure 3.9). Most commonly used one is rectified linear unit (ReLU).

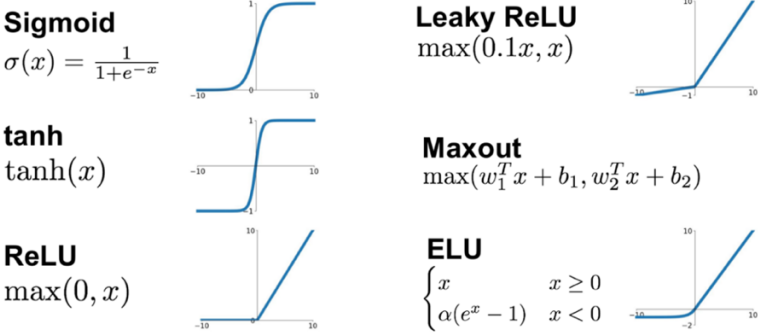


Figure 3.9 : Activation functions.

3.4.2 Pooling layer

Pooling layer reduces spatial size of the network by sub-sampling the input image. Sub-sampling reduces the number of parameters, computational load and the memory usage. This leads to lower risk of over fitting in the network. In a pooling layer, each neuron located within a filter is connected to the outputs of the neurons in the previous layer, similar to convolutional layer. However, a pooling neuron has no weights. It aggregates the inputs using an aggregation function such as the maximum or average pooling (Figure 3.10). Max-pooling takes out the largest element from a pool. On the other hand, average pooling takes out the average of the pool. By sliding the filters through the input; the maximum or the average parameter is taken out at every stride, and the rest is dropped. This leads to a down-sampled network.

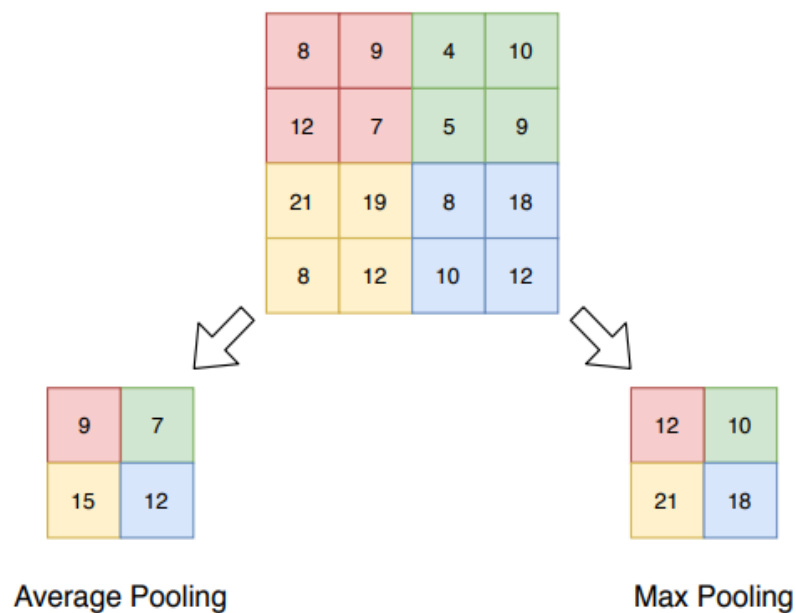


Figure 3.10 : Max pooling and average pooling operations with a filter size 2x2 and stride 2.

3.4.3 Fully connected layer

In this layer, feature map matrix will be converted to vectors and combined together to create a model. Fully connected layers connect every neuron in one layer to every neuron in another layer. The last fully-connected layer uses an activation function for

classifying the generated features of the input image into various classes based on the training dataset.

3.5 CNN Architectures

A typical CNN architectures consist of several stacked convolutional layers with each one accompanied by a ReLU layer and then several pooling layers (Figure 3.11). As the image progresses through the network it gets smaller and smaller but it also typically gets deeper and deeper due to increase in feature maps. Feedforward neural network is added at the top of the stack, and the final layer outputs the class predictions (softmax layer). Throughout the years, derivatives of this fundamental architecture have been developed, leading to significant advances in the field. Measure of this progress is the error rate in competitions. ImageNet project is a large visual database created for deep learning research. The ImageNet project runs an annual software contest called as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), where the goal is to design the highest performing classifier algorithm. It is the measure of state-of-the-art deep networks. Various visual recognition tasks such as semantic labelling, object recognition and scene classification are carried out using ImageNet dataset as benchmark. In this section, most widely used CNN architectures are introduced with their novel approaches.

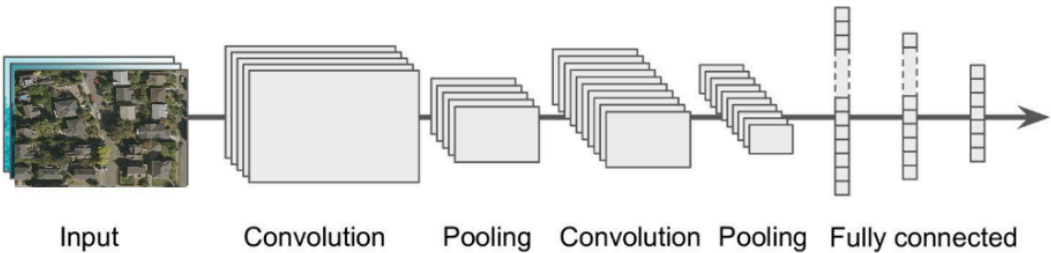


Figure 3.11 : Typical CNN architecture.

3.5.1 LeNet-5

LeNet-5 architecture[6] is the most widely known CNN architecture. LeNet-5 laid the foundation for the deeper architectures that came after. However, at the time deep neural networks were not easy to implement due to hardware restrictions and shortage of vast amount of labeled training samples. LeNet-5 commonly used for recognizing

hand written digits like MNIST dataset. MNIST dataset consists of 28×28 pixel images but they are zero-padded to 32×32 pixels. Therefore, rest of the network does not require any padding. Novelty of the LeNet-5 is in the output layer. Earlier neural networks compute the dot product of the inputs and the weight vector in the output layer. However, in LeNet-5 each neuron in the output layer computes the square of the Euclidian distance between the corresponding input and weight vector. Each output measures the probability of an image belonging to a particular digit class. Nowadays, cross entropy cost function is preferred, due to penalization of bad predictions are much more efficient, leading to larger gradients and faster convergence. LeNet-5 architecture is given below (Table 3.1).

Table 3.1 : LeNet-5 architecture.

Layer	Type	Maps	Size	Kernel Size	Stride	Activation
Out	Fully Connected	-	10	-	-	Radial Basis
F6	Fully Connected	-	84	-	-	tanh
C5	Convolution	120	1x1	5x5	1	tanh
S4	Average Pooling	16	5x5	2x2	2	tanh
C3	Convolution	16	10x10	5x5	1	tanh
S2	Average Pooling	6	14x14	2x2	2	tanh
C1	Convolution	6	28x28	5x5	1	tanh
In	Input	1	32x32	-	-	-

3.5.2 AlexNet

AlexNet introduced by Alex Krizhevsky et al [7] and won the ImageNet ILSVRC challenge at 2012. While having a similar structure with LeNet-5, AlexNet is much more deeper and larger. Also, instead of having pooling layer on top of every convolutional layer, AlexNet has stacked convolutional layers and it is the first network to employ ReLU as activation function. Main contribution of AlexNet is using

a normalization step immediately after the ReLU step of layers C1 and C3, referred as local response normalization. This form of normalization forces significantly active neurons to suppress neurons in the adjacent feature maps. This reinforces feature maps to specialize, leading to wider range of features and higher rate of generalization. LRN can be shown as,

$$b_i = a_i (k + \alpha \sum_{j=j_{low}}^{j_{high}} a_j^2)^{-\beta} \quad \text{with} \quad \begin{cases} j_{high} = \min(i + \frac{r}{2}, f_n - 1) \\ j_{low} = \max(0, i - \frac{r}{2}) \end{cases}, \quad (3.14)$$

where a_i is the activation of the neuron after the ReLU step and b_i is the normalized output of the neuron belongs to feature map i . k , α , β , and r are called as hyperparameters. k is the bias and r represents the depth radius. Finally, f_n shows the number of feature maps. AlexNet architecture is given in (Table 3.2).

Table 3.2 : AlexNet architecture.

Layer	Type	Maps	Size	Kernel Size	Stride	Padding	Activation
Out	Fully Connected	-	1000	-	-	-	Softmax
F9	Fully Connected	-	4096	-	-	-	ReLU
F8	Fully Connected	-	4096	-	-	-	ReLU
C7	Convolution	256	13x13	3x3	1	SAME	ReLU
C6	Convolution	384	13x13	3x3	1	SAME	ReLU
C5	Convolution	384	13x13	3x3	1	SAME	ReLU
S4	Max Pooling	256	13x13	3x3	2	VALID	-
C3	Convolution	256	27x27	5x5	1	SAME	ReLU
S2	Max Pooling	96	27x27	3x3	2	VALID	-
C1	Convolution	96	55x55	11x11	4	SAME	ReLU
In	Input	3 (RGB)	224x224	-	-	-	-

3.5.3 VGGNet

VGGNet introduced by Zisserman et al. [63]. VGG stands for Visual Geometry Group from University of Oxford. VGGNet uses small filters and deeper architecture compared to AlexNet. VGGNet achieved a second place in classification and first in localization in ILSVRC challenge at 2014. VGGNet has two different versions referred as VGG16 and VGG19 respectively. VGGNet has huge number of parameters which increased learning power but training this network was demanding so it is divided into smaller networks with layers added one at a time. VGGNet architecture given in (Table 3.3, Table 3.4)

Table 3.3 : VGGNet architecture.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
Input (224x224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256, conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
softmax					

Table 3.4 : Number of parameters in VGGNet in millions.

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

3.5.4 GoogLeNet

Developed by Szegedy et al. [9] from Google Research, GoogLeNet won the ILSVRC 2014 challenge by top-5 error rate below 7%. They introduced a novel sub-network referred as Inception module (Figure 3.12). This module copies the input signal and feeds it to three convolutional layers that use the ReLU activation function and also to a pooling layer. By employing convolutional layers with different kernel sizes such as 1×1 , 3×3 , and 5×5 , patterns with different dimensions can be captured. Additionally, all layers use SAME padding and a stride of 1. This makes the outputs of every layer have the same height and width as their inputs. Thus, it is possible to concatenate all the outputs in the final layer by stacking the feature maps from each convolutional layer. Convolutional layers with kernel size 1×1 serves as the bottleneck layer by reducing dimensionality. This leads to significant improvement in computing speed. Compared to AlexNet, GoogLeNet has only 6 million parameters where AlexNet has 60 million.

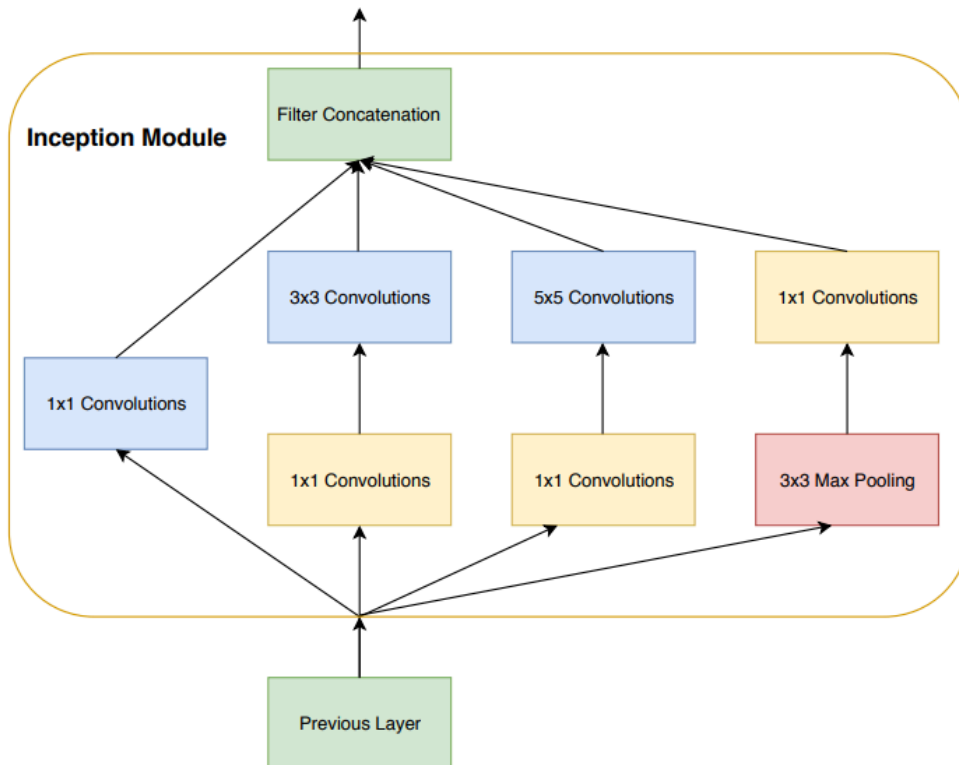


Figure 3.12 : Inception Module.

Another novelty brought by GoogleNet is the global average pooling. Previous networks used fully connected layers where all inputs are connected to each output. However, in GoogleNet global average pooling is employed nearly at the end of network by averaging each feature map from 7×7 to 1×1 (Figure 3.13).

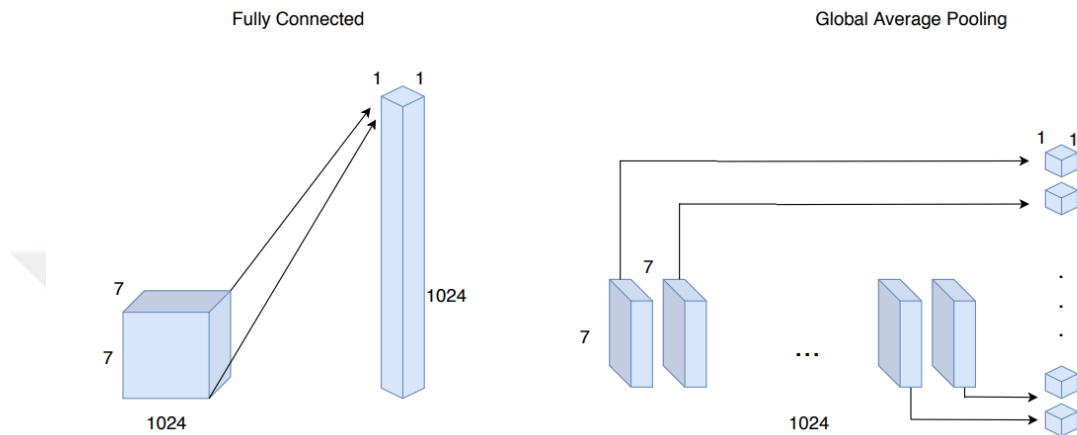


Figure 3.13 : Difference between Fully Connected Layer and Global Average Pooling.

GoogLeNet (Figure 3.14) has global average pooling layers at the end of 9 inception modules that are stacked linearly with total of 27 layers including the pooling layers.

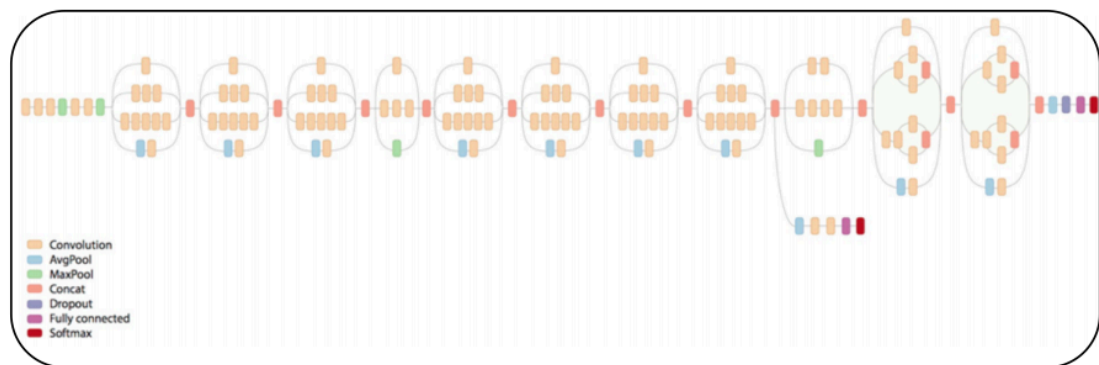


Figure 3.14 : GoogleNet architecture.

GoogLeNet is often referred as Inception-v1 and there are three more versions with upgrades which increased the accuracy of the model while reducing complexity. Inception-v2 swaps 5×5 convolutional layer with two 3×3 convolutional operations. A 5×5 convolution is computationally 2.78 times more expensive than a 3×3

convolution. Thus, factorizing 5x5 convolutions brings significant boost in performance (Figure 3.15a). Furthermore, they found out that convolutions with filter size $n \times n$ can be factorized to a combination of $1 \times n$ and $n \times 1$ convolutions in order to reduce computational complexity (Figure 3.15b). For example, instead of computing 3x3 convolution, a 1x3 convolution followed by a 3x1 convolution is performed. They have reported that this method is 33% more efficient than the single 3x3 convolution. Inception-v3 incorporated these upgrades and also added factorization of 7x7 convolutions and RMSProp Optimizer.

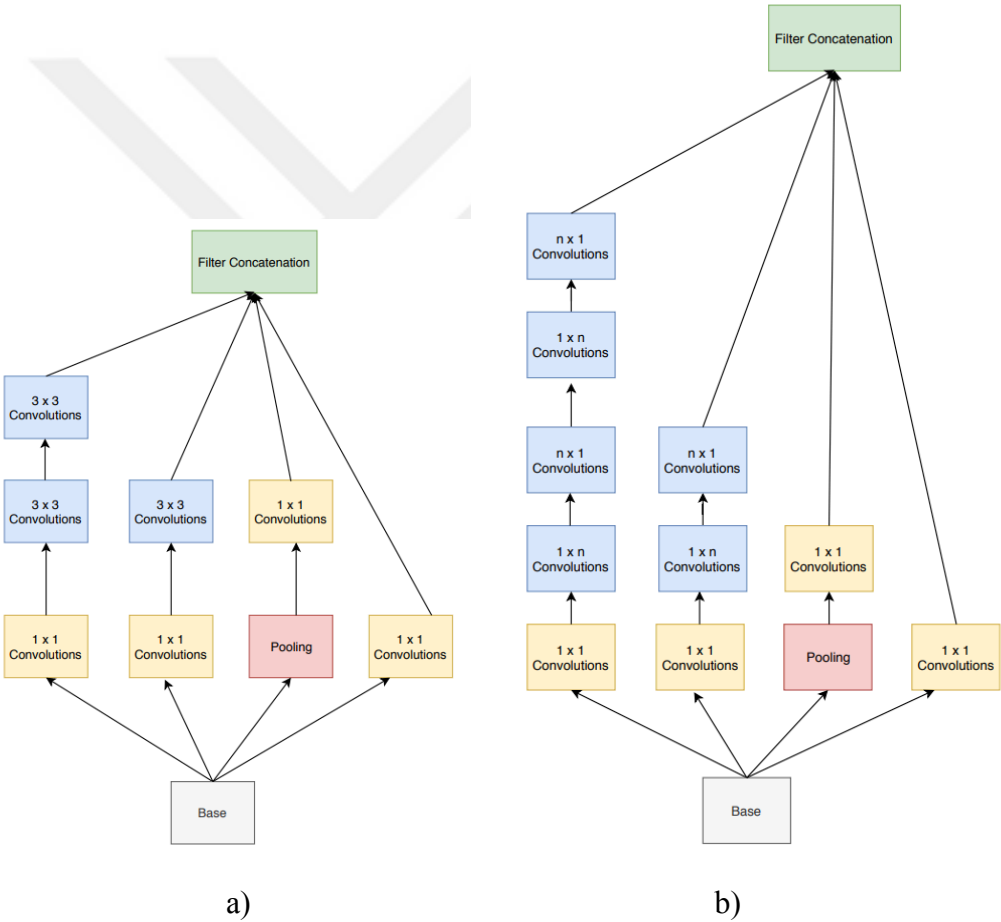


Figure 3.15 : Inception modules used in Inception-v2.

Inception-v4 introduced three different inception modules named A, B and C (Figure 3.16). Their concept is similar to Inception-v2 and Inception-v3 although modules are more uniform, leading to increase in performance.

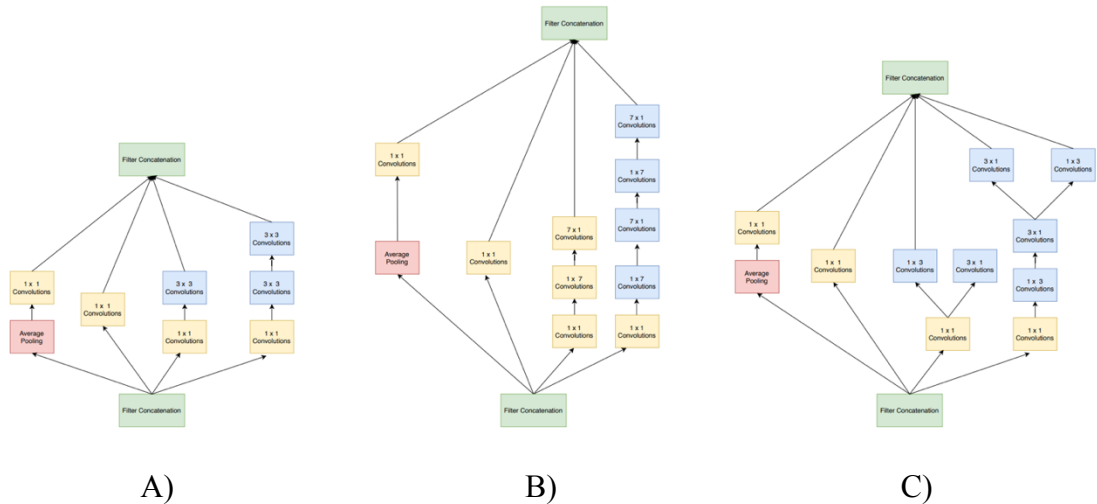


Figure 3.16 : From left Inception modules A, B, C used in Inception-v4.

3.5.5 ResNet

Residual Network (or ResNet), developed by He et al. [10] won the ILSVRC 2015 challenge with an exceptional top-5 error rate of 3.6% which set new records in detection localization and classification. They have proposed an extremely deep network with 152 layers. In theory, networks should perform better as the architecture gets deeper. However, stacking many convolutional layers create problems in terms of optimization. ResNet solves this problem by introducing residual connections which gives the network its name. Also called as the shortcut or skip connections, idea behind residual connections is to feed the input signal to the output of the layer as well. If we represent target function to model as $h(x)$, adding residual connections will force network to model $h(x) = f(x) + x$ where x is the signal added to the input and the output of the layer (Figure 3.17). When the neural network is initialized, values of its weights are near zero, therefore output values of the network are near zero. If a residual connection is added, network outputs a copy of its inputs; namely, modeling the identity function. Often times the identity function is moderately close to the target function which improves training speed significantly. Furthermore, if several residual connections are added, the network will be able to start making progress even though some layers have not begun learning yet (Figure 3.18). Because of residual connections, the signal can travel along the whole network. ResNet consists of stacked residual units, where every residual unit is a minor neural network with a residual connection. Each residual unit consists of two convolutional layers, with ReLU activation and Batch Normalization (BN) using 3×3 kernels with stride 1 and SAME

padding (Figure 3.19). Model has four different versions with 18, 34, 50, 101, 152 convolutional layers (Table 3.6).

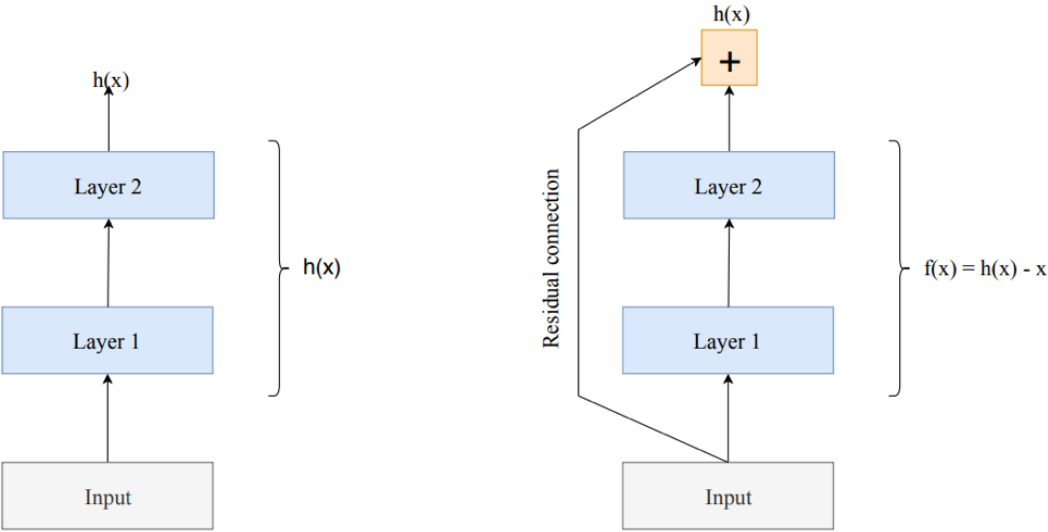


Figure 3.17 : Residual Learning.

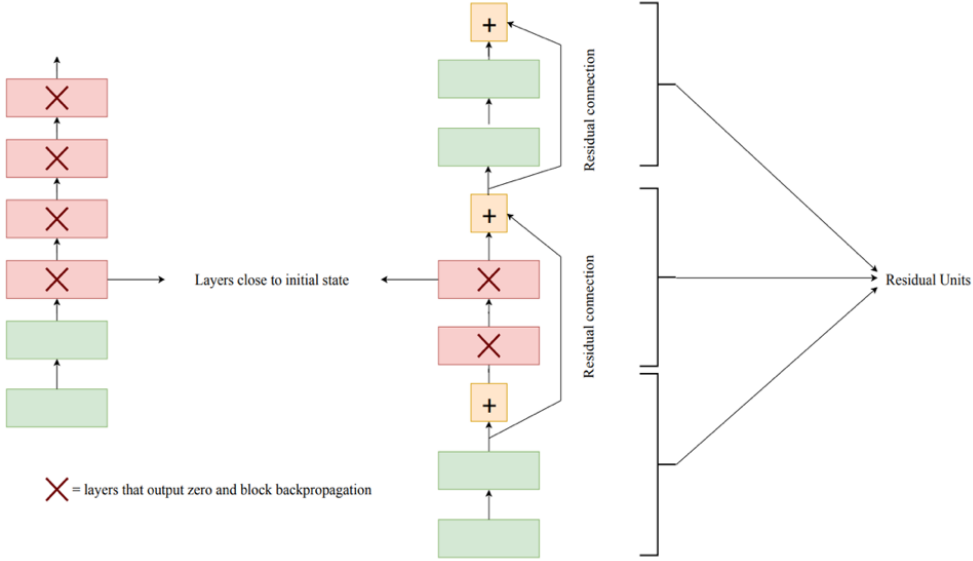


Figure 3.18 : Regular deep neural network(left) and deep residual learning(right).

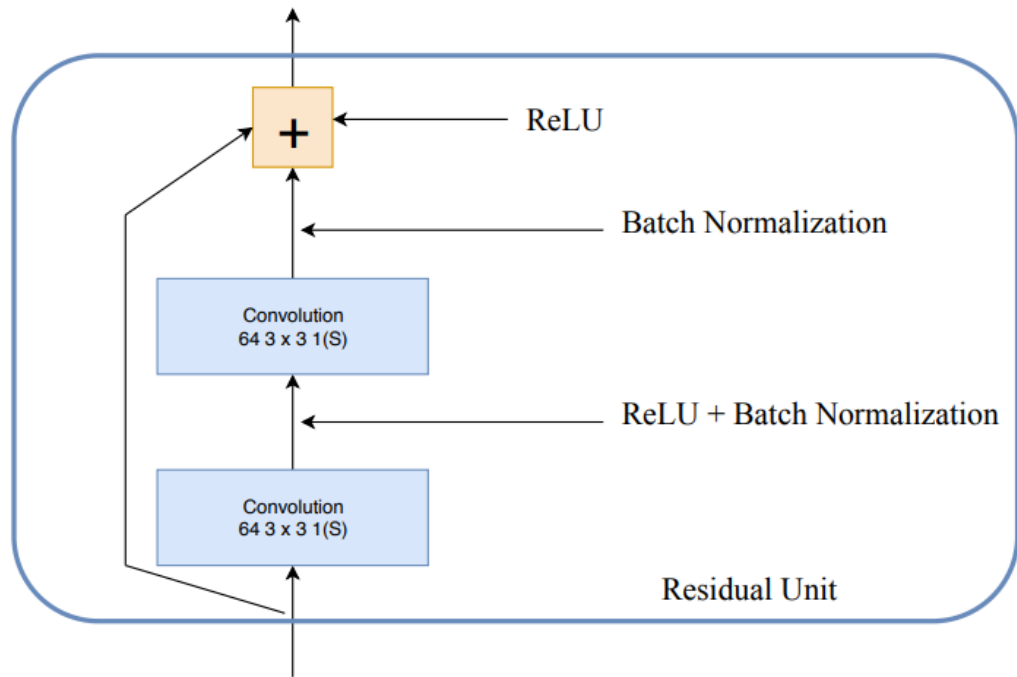


Figure 3.19 : Residual unit

Error rates of single-model results on the ILSVRC'14 validation set shows significant improvement over the VGG and GoogLeNet models (Table 3.6).

Table 3.5 : Shows the top-1 and top-5 error rates of ResNet models based on the validation set of ILSVRC 2014 [10].

Model	Top-1 Error (%)	Top-5 Error (%)
VGG-16	24.4	8.43
GoogLeNet	-	7.89
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	19.38	4.49

3.5.6 Inception-ResNet

Inspired by the performance of the ResNet, a hybrid inception module was proposed [11]. There are two sub-versions of Inception ResNet, namely v1 and v2. Both versions have the same structure for the inception modules A, B, C (Figure 3.20). However, differences are the hyper-parameter settings and the computational cost. Inception-ResNet-v1 has a computational cost that is comparable to Inception-v3 whereas Inception-ResNet-v2 has a computational cost that is comparable to Inception v4.

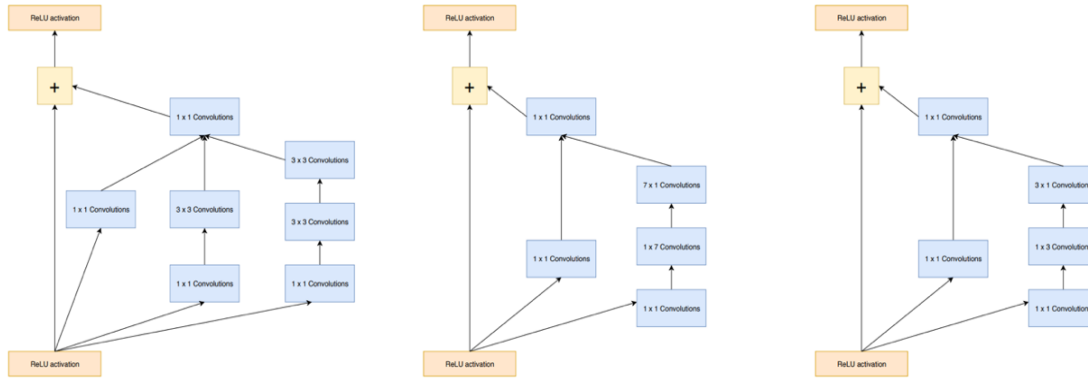


Figure 3.20 : Inception modules A,B,C in an Inception-ResNet-v1. Pooling layer was replaced by the residual connection.

Szegedy et al. report that the Inception-ResNet-v2 architecture produces more accurate results than previous state of the art models.(Table 3.7) shows the Top-1 and Top-5 validation errors on the ILSVRC 2012 image classification benchmark based on a single crop of the image. Inception-ResNet-v2 given in Figure 3.21

Table 3.6 : The top-1 and top-5 error rates of Inception models based on the validation set of ILSVRC 2012.

Model	Top-1 Error (%)	Top-5 Error (%)
BN-Inception	25.2	7.8
Inception-v3	21.2	5.6
Inception-ResNet-v1	21.3	5.5
Inception-v4	20.0	5.0
Inception-ResNet-v2	19.9	4.9

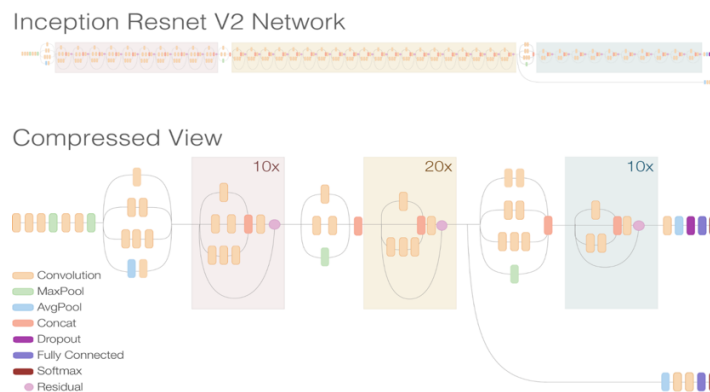


Figure 3.21 : Inception-ResNet-v2 architecture.

3.6 Training Convolutional Neural Networks

CNN's are powerful feature extractors that can produce superior performance compared to their predecessors. However, they are complex mathematical models and training a CNN model for optimum results have intricacies. This section provides the details of training a successful CNN model.

3.6.1 Hyperparameter selection

Deep learning algorithms involve “hyperparameters” which are variables set before starting the training process. CNN's can have many hyperparameters which identifies the structure of the network and governs how the network is trained. Certain critical parameters are listed and discussed below.

3.6.1.1 Loss function

Loss function is defined to compare the output of the training instance against the desired ground truth output. Ideally, loss function is minimized with respect to the connection of the weights. It is calculated after each time network makes a pass through the entire training dataset. This is also referred as an epoch. A typical loss function is the squared Euclidian distance given as,

$$L = \frac{1}{2} \sum_i (y_i - z_i)^2, \quad (3.15)$$

where y_i is the i^{th} network output and z_i is the i^{th} value of the target output. Output of the CNN's usually treated as a probability distribution where the final layer consists of the softmax function. Therefore it is more common to use cross-entropy loss defined as,

$$L = - \sum_i y_i \log z_i \quad (3.16)$$

3.6.1.2 Learning rate

The learning rate determines step size of the gradient updates. If the learning rate is set too small, the model will go through many iterations to converge. If the learning rate is set too large, the model will diverge (Figure 3.22).

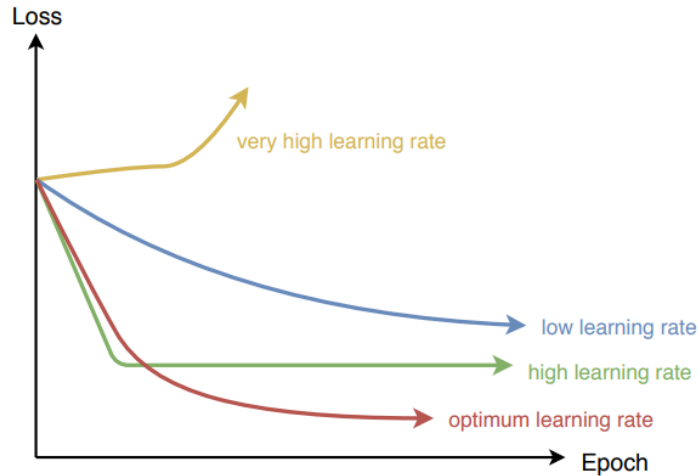


Figure 3.22 : The effect of learning rate on training loss.

The learning rate is typically decreased over time. The general approach is to find a proper parameter and it should be fine-tuned later on. It is also common to set an adaptive learning rate which regulates depending on the loss function.

3.6.1.3 Mini-batch size

Due to hardware considerations, it is not practical to train the whole training dataset at once. Usually deep networks consist of vast number of weights. Therefore, training such Big Data requires substantial amount of memory. Mini-batch training includes feeding small part of the training data to the network and computes the local gradient. However, selecting a small batch size could lead to a noisy loss function due to high variance in the gradient estimation. Mini-batch size should be selected by considering memory capacity and the training data.

3.6.2 Optimization algorithms

Optimization algorithms are used to minimize the loss function of the network by updating weights and biases. Finding optimum values for these internal parameters have a key role in training an effective model that produces accurate results. Most widely used optimization algorithms are given below.

3.6.2.1 Gradient Descent

Gradient Descent algorithm is one of the common optimization algorithms used in neural networks. Gradient Descent algorithm calculates the gradient of error function (E),

$$E = \frac{1}{2}(y - f(\sum w_i x_i))^2, \quad (3.17)$$

and updates parameters in the opposite direction of the gradient vector of error as shown in (Figure 3.23). Error rate increases if the value of the weights is too small or too large. Thus, weights need to be updated and optimized until reaching a local minima.

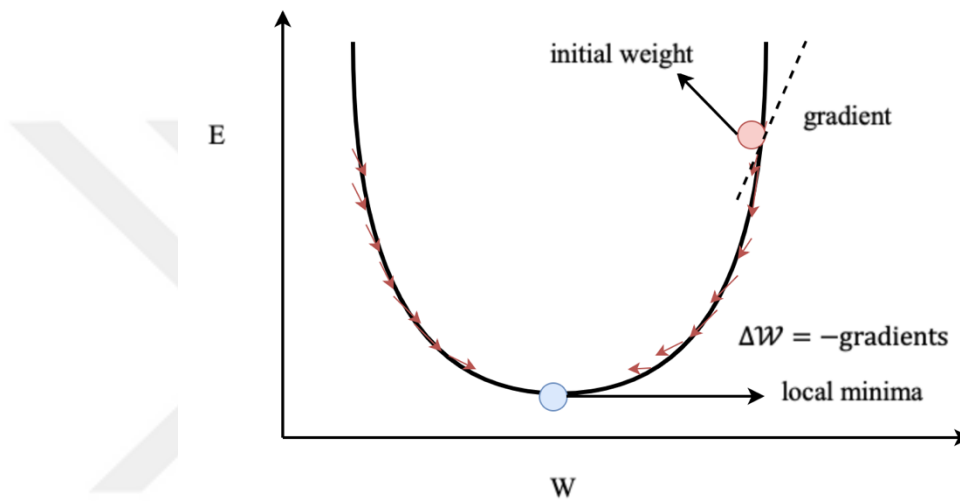


Figure 3.23 : Weight updates in the opposite direction of the gradient.

Standard Batch Gradient Descent algorithm updates parameters after calculating the gradient of the whole data set. This will cause model to converge slower and it is not applicable for large datasets. These issues are rectified in the following variants of the gradient descent algorithm.

3.6.2.2 Stochastic Gradient Descent(SGD)

Unlike standard gradient descent algorithm SGD updates parameters with each training instance. It is defined as,

$$\theta = \theta - \eta \cdot \nabla J(\theta; x(i); y(i)) \quad (3.18)$$

where θ shows the models parameters, $-\eta$ is the learning rate, $\nabla J(\theta)$ is the gradient of loss function J and $x(i), y(i)$ are the training instances.

As a result of frequent parameter updates, loss function oscillates to different amounts (Figure 3.24). This could lead to a newer and lower local minima, however it can also cause model to keep overshooting.

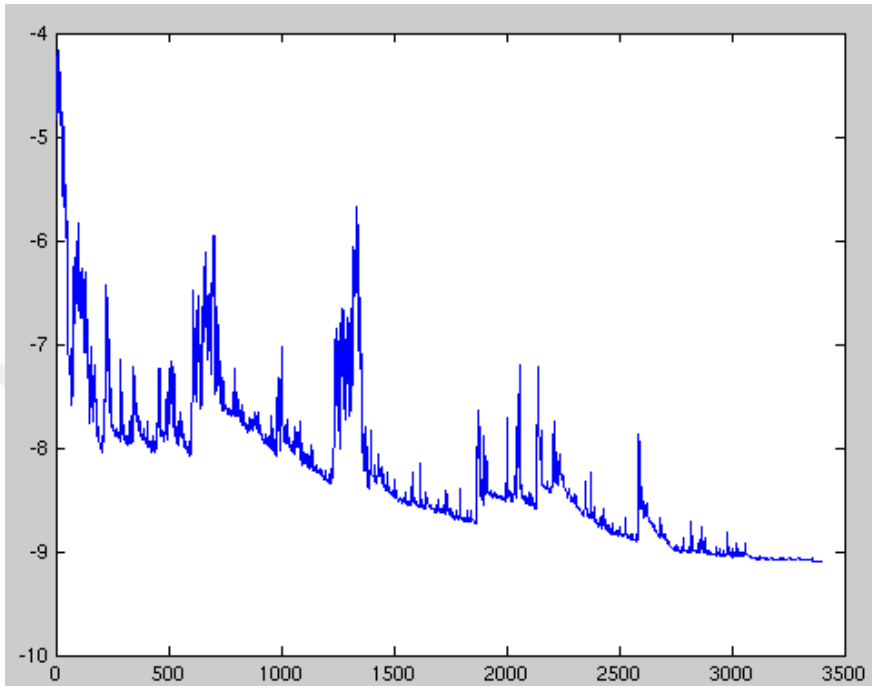


Figure 3.24 : SGD fluctuates to find a newer and better local minima.

In order to tackle problems such as fluctuations in loss function and slower convergence, a method called Momentum is employed which speeds up SGD by steering towards to suitable direction and reduce oscillations in unrelated directions. The momentum term γ controls parameter updates only for the relevant instances which lead to accelerated convergence and lessen oscillations. Momentum and the parameter updates are,

$$V(t) = \gamma V(t-1) + \eta \nabla J(\theta), \quad (3.19)$$

$$\theta = \theta - V(t). \quad (3.20)$$

3.6.2.3 AdaGrad

Adaptive Gradients (AdaGrad) is an optimization algorithm that adapts learning rate - η to the parameters [64]. Larger updates are carried out for infrequent parameters and smaller updates for frequent parameters. Therefore, it is suitable when using sparse

data. AdaGrad adapts the learning rate η for every parameter θ_i at each time step t based on the past gradients. AdaGrad's per-parameter update is shown as,

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}, \quad (3.21)$$

where $g_{t,i}$ is the gradient of the loss function to the parameter θ_i at time step t .

Main disadvantage of AdaGrad algorithm is learning rate is constantly decaying due to accumulation of the gradients. This causes learning ability of the model decrease leading to longer training time.

3.6.2.4 RMSProp

Problem of constant decaying learning rate in AdaGrad is rectified in Root Mean Square Propagation (RMSProp) algorithm[65] by changing the gradient accumulation into an exponentially weighted moving average. Exponential average weights the recent gradient updates more than the previous ones shown as,

$$V_t = \rho V_{t-1} + (1 - \rho) \cdot g_t^2, \quad (3.22)$$

$$\Delta W_t = -\frac{\eta}{\sqrt{V_t + \epsilon}} \cdot g^t, \quad (3.23)$$

$$W_{t+1} = W_t + \Delta W_t, \quad (3.24)$$

where V_t is the exponential average of the squares of gradients and ρ is the weight of the recent gradient update.

3.6.2.5 Adam

Adaptive Moment Estimation (Adam) is another adaptive optimization algorithm [66]. Adam keeps an exponential average of past squared gradients like RMSprop and also stores an exponentially decaying average of past gradients $M(t)$, similar to momentum. Parameter update for Adam is given as,

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{V_t + \epsilon}} \cdot M(t). \quad (3.25)$$

Adam outperforms other adaptive optimization algorithms as it converges faster. Furthermore, it rectifies problems that are present in other optimization methods such as vanishing learning rate, slow convergence and fluctuating loss function.

3.6.3 Regularization

Regularization methods are employed to reduce the generalization error of the model. Deep learning models may produce high rate of validation error even after training error drops, resulting in overfitting (Figure 3.25). However, a successfully trained model needs to produce accurate results with validation or test data. Regularization strategies carried out to reduce overfitting at the expense of increasing training error. These strategies include putting extra constraints on the parameter values or adding extra terms on objective functions such as loss function. Typically used regularization methods are listed and discussed below.

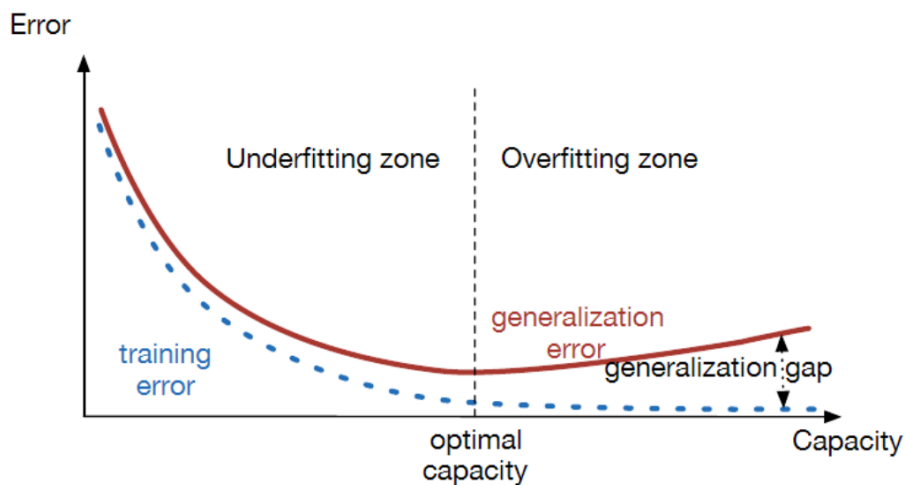


Figure 3.25 : Graph showing underfitting and overfitting in the network.

3.6.3.1 L2 regularization

L2 regularization technique involves adding a new term to the loss function in order to penalize large weights. Sum of the squared norms of the weight matrices multiplied by a constant $\frac{\lambda}{2m}$ is added to the loss function shown as,

$$loss + \left(\sum_{j=1}^n \|w^{[j]}\|^2 \right) \cdot \frac{\lambda}{2m}, \quad (3.26)$$

where n denotes the number of layers, m is the number of inputs, $w^{[j]}$ is the weight matrix of j th layer and λ is the regularization parameter.

3.6.3.2 L1 regularization

L1 regularization adds sum of the absolute values of the weights multiplied by the regularization parameter λ to the loss function shown as,

$$loss + \left(\sum_{j=1}^n \|w^{[j]}\| \right) \cdot \lambda, \quad (3.27)$$

L1 Regularization reduces weights by a fixed amount in every iteration, regardless of the value of the weight. Thus, weight of most of the connections inclines to zero and fewer connections left with larger weights. This increases sparsity of the weights in the model.

3.6.3.3 Dropout regularization

Dropout is an efficient regularization technique that includes randomly erasing neurons in the dropout layers. Thus, whole network can be represented as a sub-network with fewer connections required to update throughout back propagation. Dropout encourages the network to learn a sparse representation. Consequently, over fitting is reduced. (Figure 3.26) illustrates the dropout regularization.

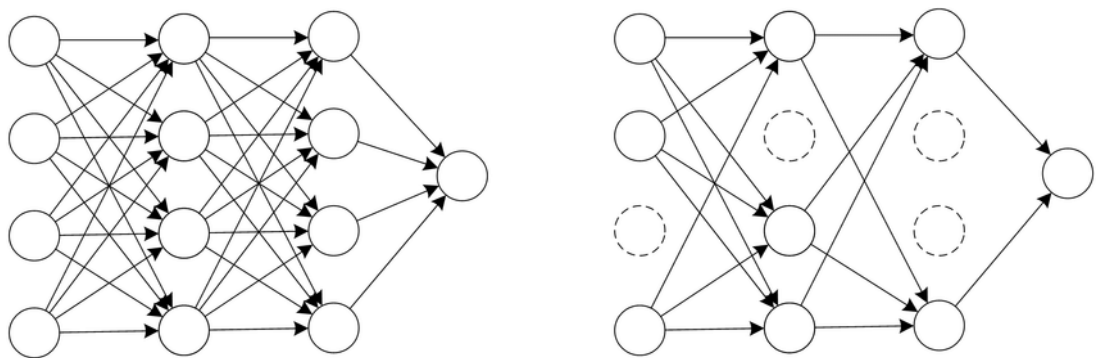


Figure 3.26 : Dropout Regularization. Standart network(left) network with dropout(right).

3.6.3.4 Batch normalization

Batch normalization is the regularization method of normalizing network's parameters in order to adjust and scale the activations [67]. For instance, certain values of neurons in the input layer can be set between 0 and 1 and other neurons could take values between 1 and 1000. Same approach can be applied for the hidden layers as well. Consequently, higher learning rates can be applied to the model since activation value of neurons can't exceed or fall beyond the given range. This leads to an increase in speed of training and stability of the network. Batch normalization works by normalizing the output of previous activation layer. In order to do so, batch mean is subtracted from the output and the resultant is divided by the batch standard deviation as given in following equations,

$$\frac{1}{m} \sum_{i=1}^m x_i = \mu_{\beta}, \quad (3.28)$$

$$\frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\beta})^2 = \theta_{\beta}^2, \quad (3.29)$$

$$\frac{x_i - \mu_{\beta}}{\sqrt{\theta_{\beta}^2 + \epsilon}} = \hat{x}_i, \quad (3.30)$$

where β denotes the values of x over a mini-batch $\{x_1 \dots m\}$, μ_{β} is the mini-batch mean, θ_{β}^2 is the mini-batch variance and \hat{x}_i is the normalized value.

3.6.3.5 Data augmentation

Insufficient amount of training data leads to over fitting of the network. Data augmentation or in other words regularization with data provides new data from existing data by performing different operations such as translation, rotation, reflection, skewing, scaling, or changing contrast or brightness of the input image data. Also, there are other augmentation techniques that can't be experienced with human eye such as adding random noise to the training data. In the field of remote sensing,

acquiring labeled data is not very easy due to the commercial restrictions and economical costs. Therefore, it is crucial to make the most with the data available.

3.7 Transfer Learning

State-of-the-art deep networks namely Inception, ResNet and VGGNet are really large networks such that training one from scratch takes several weeks and requires advanced computing resources (i.e. GPU's). However, each of these networks is already trained with ImageNet dataset that consists of millions of labeled images. So the weights in the different layers of the model already learned to identify useful low-level features such as shapes, edges and different intensities of light and dark pixels. By using transfer learning method, parameters learnt from a training model can be used for a different classification problem (Figure 3.27). Only, final layers of the network need to be fined tuned for the classification task. For remote sensing applications such as land cover land use classification, obtaining a huge training data similar to ImageNet data set is not very realistic. In these cases, using pre-trained networks are highly beneficial for reducing computing time and accurate classification results.

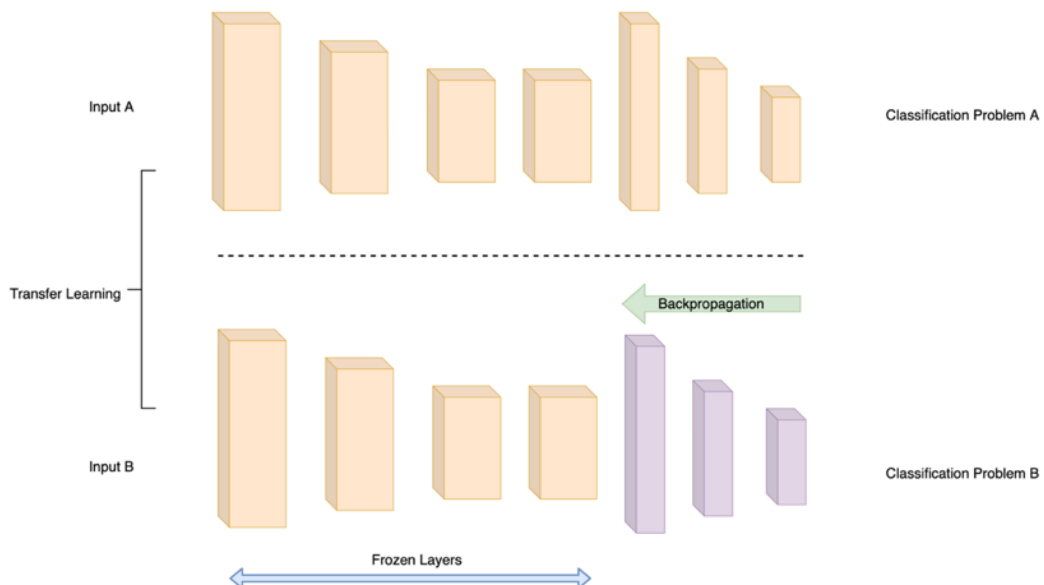


Figure 3.27 : Transfer learning.



4. EXPERIMENTS AND RESULTS

Accessibility of the remote sensing imagery have become increasingly available with the launch of new commercial satellite sensors, such as IKONOS, WorldView and GeoEye. Remote sensing applications for these imagery provides crucial analysis for various subjects such as urban planning, climate change observations, resource management, and land use monitoring. These satellites can deliver panchromatic or multispectral images. Panchromatic images have higher resolution compared to multispectral images however, they include only a single band. Contrarily, multispectral images can contain several bands such as Red, Green, Blue and Near-Infrared.

Land cover and land use classification using satellite imagery is a challenging task. Traditional methods require extensive expertise on extracting features for land cover and land use classes. However, deep learning models eliminate the need of feature engineering by learning from the data itself. Moreover, increase in the available remote sensing imagery, calls for an automatization for the process of land use analysis. In this regard, CNN-based deep learning experiments are carried out for the purpose of image scene classification are discussed in this chapter.

4.1 Image Scene Classification for Land Cover and Land Use Analysis

In this section, CNN-based deep learning classifiers are proposed in the context of land cover and land use classification. Details of training these networks are discussed below.

4.1.1 Proposed classification networks

For the classification of remote sensing imagery deep learning models provide far better performance compared to traditional methods. Variety of deep learning models especially CNN's, produce results that can be generalized over the unseen data. In this experiment, two state-of-the-art CNN models namely Inception-ResNet-v2 and Inception-v4 are trained using pretrained weights from the ImageNet dataset. These

architectures are proven to outperform existing CNN models in terms of classification accuracy and faster convergence according to ILSVRC challenge.

4.1.2 Training dataset for the classification network

Training dataset is extracted from the NWPU-RESISC45 dataset[68] which is publicly available benchmark for Remote Sensing Image Scene Classification (RESISC). Images are size of 256×256 pixels in the red green blue (RGB) color space. The spatial resolution varies from about 30 m to 0.2 m per pixel. Dataset is extracted from Google Earth by various experts in remote sensing. Compared to previous benchmark datasets NWPU-RESISC45 is large-scale and have rich image variations with different weathers, seasons, illumination conditions, imaging conditions, and scales. For each scene category, dataset possesses much rich variations in translation, viewpoint, object pose and appearance, spatial resolution, illumination, background, and occlusion. Also it provides high class diversity and between class similarity to truly test deep learning model's classification capability. NWPU-RESISC45 dataset consists of 31,500 remote sensing images divided into 45 scene classes. One of the crucial aspect of the experiment carried out in this thesis is to use independent dataset for the validation part. In order to do so, patches have to be extracted manually and the process can be cumbersome for generating test images for 45 classes. So the classes to be used in the training are narrowed down to 20 scene classes each containing 700 training images. These 20 scene classes include airport, chaparral, dense residential, forest, freeway, golf course, ground track field, industrial area, intersection, meadow, medium residential, overpass, parking lot, rectangular farmland, river, runway, sparse residential, storage tank, tennis court and terrace. These classes are selected because they are easy to obtain in any given AOI and have complex structure as well as similarities to each other that can be used to assess model's classification capability. Sample patches from the training dataset are given in (Figure 4.1).


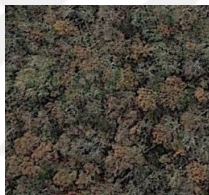








Airport				
Chaparral				
Dense Residential				
Forest				
Freeway				
Golf Course				
Ground Track Field				

Figure 4.1 : Sample patches for the training dataset.

Industrial Area				
Intersection				
Meadow				
Medium Residential				
Overpass				
Parking Lot				
Rectangular Farmland				

Figure 4.1 (continued) : Sample patches for the training dataset.

River				
Runway				
Sparse Residential				
Storage Tank				
Tennis Court				
Terrace				

Figure 4.1 (continued) : Sample patches for the training dataset.

4.1.3 Validation dataset for the classification network

Images for validation dataset is extracted from Worldview-3 satellite imagery that is made publicly available for SpaceNet challenge. Images are 3-band RGB with 16-bit and 30 cm resolution. Before extracting the patches pixel values of the images had to be normalized to 8-bit due to the fact that models used in the experiment only accept JPEG images. Pixel values of the images are normalized to 8-bit using an automated script with parallel programming and “GDAL” library. Code snippet about the script is given in (Figure 4.2).

Patches are extracted and labeled manually from three different AOI’s including Vegas, Paris and Shanghai. Area of raster belonging to the AOI’s are 216, 1030 and 1000 square kilometers. 50 patches are extracted for each class with a total of 1000 patches. Selected scenes usually have intra-class variability in order to truly test the networks capability with minimum bias. Patches have different ground sampling distances illumination and occlusion. Sample patches for the validation dataset are given in (Figure 4.3).

```
#!/bin/bash
tiftotalcount=$(find ~/vegas -type f -name "*.TIF" -printf "x" | wc -c)
tiftotalsize=$(find ~/vegas -type f -name "*.TIF" -exec du -b {} \; | awk '{total+=$1} END {totalGB=total/(1024*1024*1024); printf "%06.2f", totalGB}')
datestamp=$(date +"%Y%m%d_%H%M%S")
logfile=$(printf "CONVERT-16BIT28BIT-%s-%02dfiles-%sGB.log" $datestamp $pixtotalcount $piftotalsize)
tifdonefile=$(printf "CONVERTDONE-%s.log" $datestamp)
echo "CHECK LOGFILE $logfile"
date +"[%Y/%m/%d %H:%M:%S] START" >> $logfile
find ~/vegas -type f -name "*.TIF" | parallel --no-notice "gdal_translate -ot Byte -scale -b 1 -b 2 -b 3 {} ~/vegas8bitparallel/{/} >> $logfile 2>&1; date +"[%Y/%m/%d %H:%M:%S] PROCESSED {}\" >> $logfile; echo {} >> $tifdonefile"
date +"[%Y/%m/%d %H:%M:%S] END">> $logfile
```

Figure 4.2 : Code for automated script to convert 16-bit imagery to 8-bit.


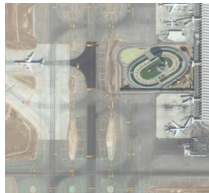



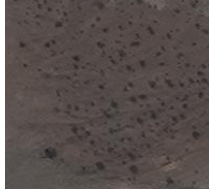

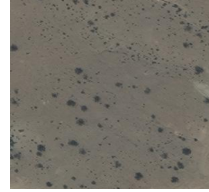
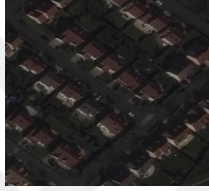
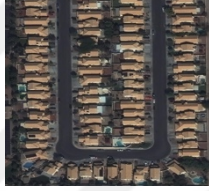
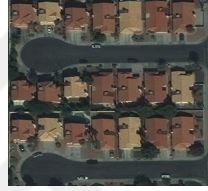
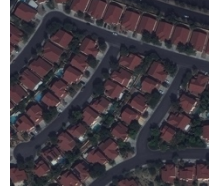
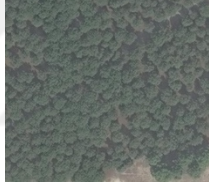

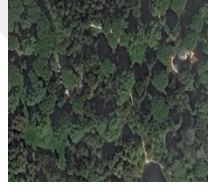
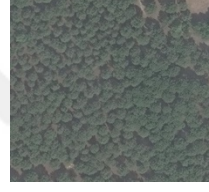
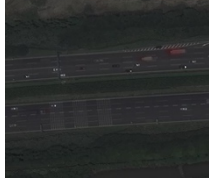


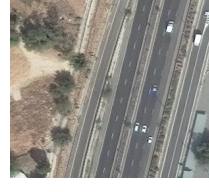

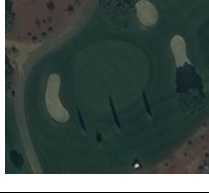

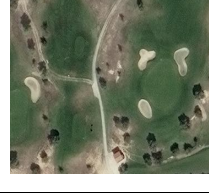
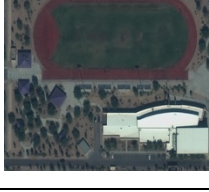
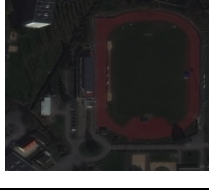

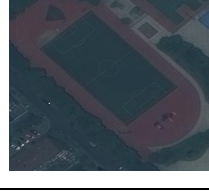
Airport				
Chaparral				
Dense Residential				
Forest				
Freeway				
Golf Course				
Ground Track Field				

Figure 4.3 : Sample patches for the validation dataset.



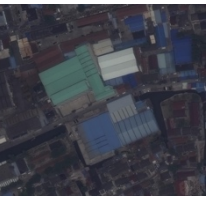





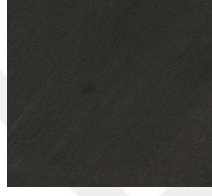

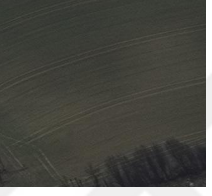
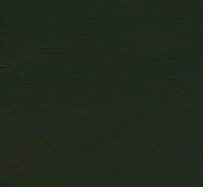


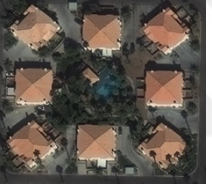
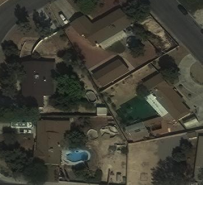
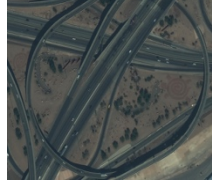

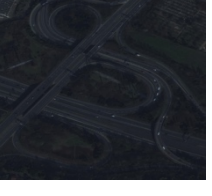
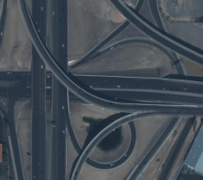
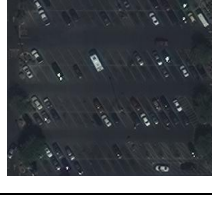

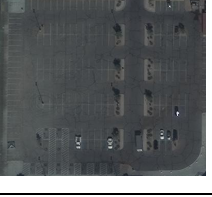
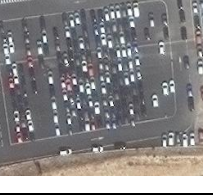
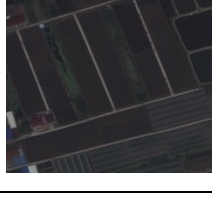
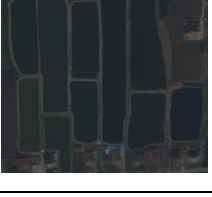
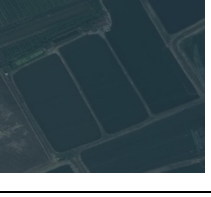
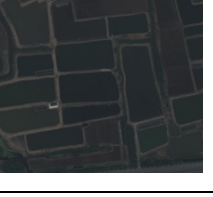
Industrial Area				
Intersection				
Meadow				
Medium Residential				
Overpass				
Parking Lot				
Rectangular Farmland				

Figure 4.3 (continued) : Sample patches for the validation dataset.







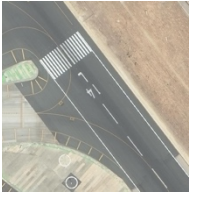

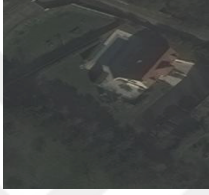

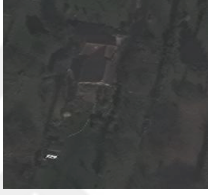


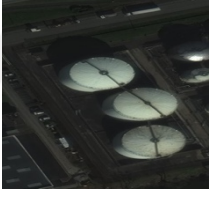






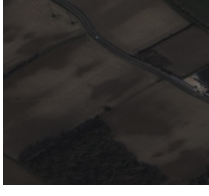


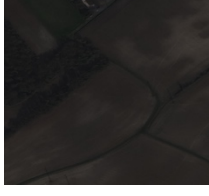
River				
Runway				
Sparse Residential				
Storage Tank				
Tennis Court				
Terrace				

Figure 4.3 (continued) : Sample patches for the validation dataset.

4.1.4 Pre-processing

Before feeding data to the network, datasets are converted to a file format referred as .tfrecords. Raw image data can be slower to read from the disk and take up significant space in the RAM. TFrecords are a binary file format storage that works as a buffer for loading data to the network. Also it is possible to sequence and shuffle the data to provide diversity in each batch. Data augmentation is also integrated to further make use of the data. All classes are coded with one-hot label and images are resized to 299 x 299 pixels as it's the default Inception size.

4.1.5 Training setup

Proposed CNN's were trained using tensorflow[68] framework and TF-Slim library. Training were conducted on a nVidia GTX 1060 6 GB GPU using cross-entropy loss function, decaying learning rate and ADAM optimizer. Learning rate has been set to $2.000e^{-4}$. Research has shown that increasing batch size throughout training leads to faster convergence with more accurate results[70]. Both networks trained for 100 epochs with increasing batch size from 12 to 16 at epoch 50 and 16 to 20 at epoch 75. Total training time for Inception-ResNet-v2 and Inception-v4 are around 10 hours and 12 hours respectively.

4.1.6 Results

Training loss, accuracy and validation accuracy are the crucial metrics for evaluation of the networks. These metrics are given in (Table 4.1).

Table 4.1 : Results of the trained networks

Proposed CNN Model	Training Loss	Training Accuracy	Validation Accuracy
Inception-ResNet-v2	0,471	0,967	0,828
Inception-v4	0,543	0,942	0,778

Benefits of having residual connections can be seen as the first network provides more accurate results with faster convergence. In order to further discuss the comparison between two networks, classification outcomes of each network needs to be

represented with certain metrics. Predictions for the classes can be divided into two groups whether the label matches the ground truth of the actual class or not. Positive classification outcome denotes the model predicted the desired label regardless of the ground truth of the class. If it matches the ground truth it is referred as the True Positive if not it is False Positive. Negative classification outcome means the model couldn't predict the desired label. However, if the ground truth for the negatively classified image is also negative, it means that the model is successful. This is referred as the True Negative. On the contrary, False Negative stands for the situation where the model did not predict the desired label for the specified ground truth. Explained in (Table 4.2).

Table 4.2 : Error types for classification.

		Actual Class	
		Positive	Negative
Classification Outcome	Classification Outcome Positive	True Positive(TP)	False Positive(FP)
	Classification Outcome Negative	False Negative(FN)	True Negative(TN)

Precision and recall are two very important model evaluation metrics. Precision refers to the percentage of the relevant results whereas recall refers to the percentage of total relevant results accurately classified by the model. For simplicity, there is another metric available, called F-1 score, which is a harmonic mean of precision and recall. Metrics are given in equation (4.1)

$$Precision = \frac{TP}{TP+FP}, \quad Recall = \frac{TP}{TP+FN}, \quad F1 = 2 \frac{P.R}{P+R} \quad (4.1)$$

Precision recall and F-1 Score of the trained networks are given in the following (Table 4.3) and (Table 4.4). Confusion matrix for the networks are given in (Figure 4.4) and (Figure 4.5). Examples of the false negatives and false positives of the networks are given with their prediction and ground truth respectively in (Figure 4.6)

Table 4.3 : Precision recall and f1-scores for the Inception-ResNet-v2.

Class	Airport	Chaparral	Dense Residential	Forest	Freeway	Golf Course	Ground Track Field	Industrial Area	Intersection
Precision	0,877	0,803	0,816	0,918	0,807	0,846	0,955	0,750	0,836
Recall	0,860	0,788	0,816	0,882	0,875	0,862	0,877	0,823	0,820
F1-Score	0,868	0,795	0,816	0,900	0,840	0,854	0,914	0,785	0,828

Class	Meadow	Medium Residential	Overpass	Parking Lot	Rectangular Farmland	River	Runway	Sparse Residential	Storage Tank
Precision	0,769	0,759	0,833	0,860	0,705	0,888	0,931	0,863	0,934
Recall	0,784	0,803	0,800	0,877	0,734	0,934	0,872	0,863	0,914
F1-Score	0,776	0,780	0,816	0,868	0,719	0,910	0,901	0,863	0,924

Class	Tennis Court	Terrace	Average
Precision	0,933	0,763	0,842
Recall	0,893	0,707	0,839
F1-Score	0,913	0,734	0,840

Table 4.4 : Precision recall and f1-scores for the Inception-v4.

Class	Airport	Chaparral	Dense Residential	Forest	Freeway	Golf Course	Ground Track Field	Industrial Area	Intersection
Precision	0,836	0,775	0,760	0,918	0,666	0,836	0,909	0,689	0,775
Recall	0,803	0,788	0,760	0,823	0,800	0,788	0,833	0,769	0,745
F1-Score	0,819	0,781	0,760	0,867	0,727	0,811	0,869	0,727	0,760

Class	Meadow	Medium Residential	Overpass	Parking Lot	Rectangular Farmland	River	Runway	Sparse Residential	Storage Tank
Precision	0,627	0,703	0,800	0,800	0,653	0,888	0,833	0,808	0,918
Recall	0,711	0,760	0,750	0,833	0,708	0,829	0,754	0,760	0,918
F1-Score	0,666	0,730	0,774	0,816	0,679	0,857	0,792	0,783	0,918

Class	Tennis Court	Terrace	Average
Precision	0,877	0,612	0,784
Recall	0,895	0,612	0,785
F1-Score	0,886	0,612	0,783

		Reference Data																						
Classified Image		Airport	Chaparral	Dense Residential	Forest	Freeway	Golf Course	Ground Track Field	Industrial Area	Intersection	Meadow	Medium Residential	Overpass	Parking Lot	Rectangular Farmland	River	Runway	Sparse Residential	Storage Tank	Tennis Court	Terrace	Classified Totals	Users' Accuracy	
	Airport	43							2								4						49	%87
	Chaparral		41								4							2				4	51	%80
	Dense Residential			40					2			4				2					1		49	%81
	Forest				45		2				2												49	%91
	Freeway					42				3			3	2			2						52	%84
	Golf Course				3		44				3					2							52	%84
	Ground Track Field							43													2		45	%95
	Industrial Area	3		3					42				2	2		2				2			56	%75
	Intersection		1			1				41						1	3	2					49	%83
	Meadow		2		3		3	2			40		2										52	%76
	Medium Residential			5					1			41	1					6					54	%75
	Overpass					3			1			1	40	2									48	%83
	Parking Lot					2			1	2		1	2	43									50	%86
	R. Farmland		2													36						13	51	%70
	River		2					2	1							40							45	%88
	Runway	4								2								40					46	%86
	Sparse Residential		2								2	5							40				49	%81
	Storage Tank								1							1	2			47			51	%92
	Tennis Court			1				4		2											45		52	%86
Terrace		2													13						35	50	%70	
Reference Totals	50	52	49	51	48	51	49	51	50	51	51	50	49	49	48	51	50	49	48	52	1000			
Producers' Accuracy	%86	%78	%81	%88	%87	%86	%87	%82	%82	%78	%80	%80	%87	%73	%83	%78	%80	%95	%93	%67				

Overall Accuracy : 828/1000 = %82.8

Figure 4.4 : Confusion matrix for the Inception-ResNet-v2 network.

		Reference Data																						
Classified Image		Airport	Chaparral	Dense Residential	Forest	Freeway	Golf Course	Ground Track Field	Industrial Area	Intersection	Meadow	Medium Residential	Overpass	Parking Lot	Rectangular Farmland	River	Runway	Sparse Residential	Storage Tank	Tennis Court	Terrace	Classified Totals	Users' Accuracy	
	Airport	41							3								5						49	%83
	Chaparral		38								5							2				4	49	%77
	Dense Residential			38					4			5				2					1		50	%76
	Forest				42		4																46	%91
	Freeway					40				5	5		4	3			3						60	%66
	Golf Course				3		41				3					2							49	%83
	Ground Track Field							40													4		44	%90
	Industrial Area	5		5					40				2			2				4			58	%68
	Intersection		1			1				38					3		1	3	2				49	%77
	Meadow		4		6		5	4	1		37		2										59	%62
	Medium Residential			6					1			38	2						7				54	%70
	Overpass					5			1			1	36	2									45	%80
	Parking Lot					4				3		1	2	40									50	%80
	R. Farmland		2						1							34						15	52	%65
	River		2					2	1							39							44	%88
	Runway	5								3								40					48	%83
	Sparse Residential		2								2	5							38				47	%80
	Storage Tank								1							1	2			45			49	%91
	Tennis Court							4		2											43		49	%87
Terrace		4	1											14							30	49	%61	
Reference Totals	51	53	50	51	50	52	48	52	51	52	50	48	48	48	47	53	50	49	48	49		1000		
Producers' Accuracy	%80	%71	%76	%82	%80	%78	%83	%76	%74	%71	%76	%75	%83	%70	%82	%75	%76	%91	%89	%61				

Overall Accuracy : 778/1000 = %77.8

Figure 4.5 : Confusion matrix for the Inception-v4 network.

	Inception-ResNet-v2		Inception-v4	
	False Negative	False Positive	False Negative	False Positive
Airport	 Industrial Area	 Runway	 Industrial Area	 Runway
Chaparral	 Meadow	 Sparse Residential	 Terrace	 Sparse Residential
Dense Residential	 Tennis Court	 Medium Residential	 Medium Residential	 Industrial Area
Forest	 Chaparral	 Meadow	 Golf Course	 Meadow
Freeway	 Overpass	 Overpass	 Overpass	 Parking Lot
Golf Course	 Meadow	 Forest	 Meadow	 River

Figure 4.6 : Error instances of the trained networks.







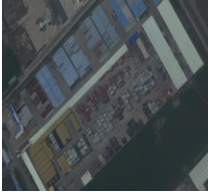


















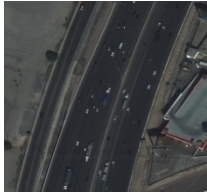
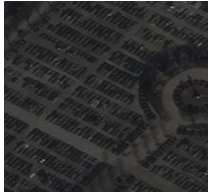

Ground Track Field	 Tennis Court	 Tennis Court	 Tennis Court	 Tennis Court
Industrial Area	 Dense Residential	 Airport	 Parking Lot	 Airport
Intersection	 Tennis Court	 Runway	 Chaparral	 Freeway
Meadow	 Forest	 Forest	 Forest	 Golf Course
Medium Residential	 Sparse Residential	 Dense Residential	 Parking Lot	 Dense Residential
Overpass	 Medium Residential	 Freeway	 Industrial Area	 Freeway
Parking Lot	 Freeway	 Freeway	 Industrial Area	 Overpass

Figure 4.6 (continued) : Error instances of the trained networks.







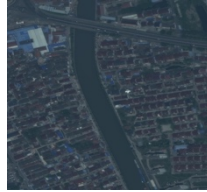
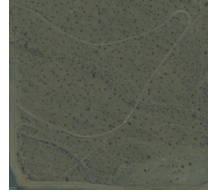







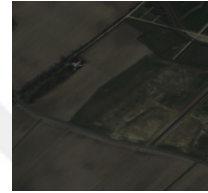

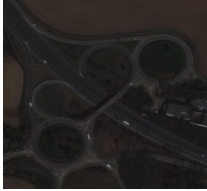


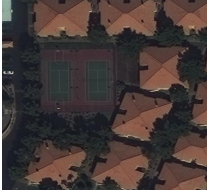




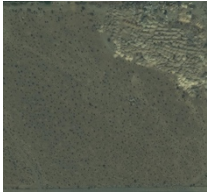


R. Farmland	 Terrace	 Terrace	 Terrace	 Terrace
River	 Dense Residential	 Chaparral	 Industrial Area	 Chaparral
Runway	 Airport	 Airport	 Intersection	 Airport
Sparse Residential	 Meadow	 Medium Residential	 Chaparral	 Terrace
Storage Tank	 Industrial Area	 Overpass	 Industrial Area	 Runway
Tennis Court	 Medium Residential	 Intersection	 Sparse Residential	 Ground Track Field
Terrace	 Farmland	 Chaparral	 River	 Farmland

Figure 4.6 (continued) : Error instances of the trained networks.

Loss function, test accuracy and the learning rate of the networks are given in (Figure 4.7), (Figure 4.8) and (Figure 4.9). X-axis shows the step size for all figures and Y-axis is the loss rate.

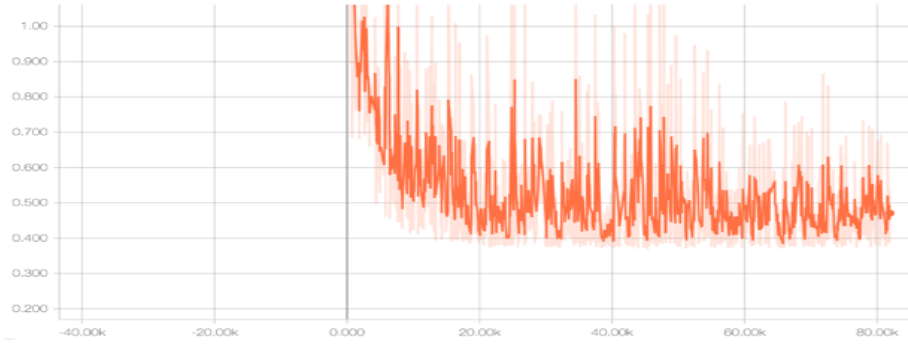


Figure 4.7 : Loss function for Inception-ResNet-v2.

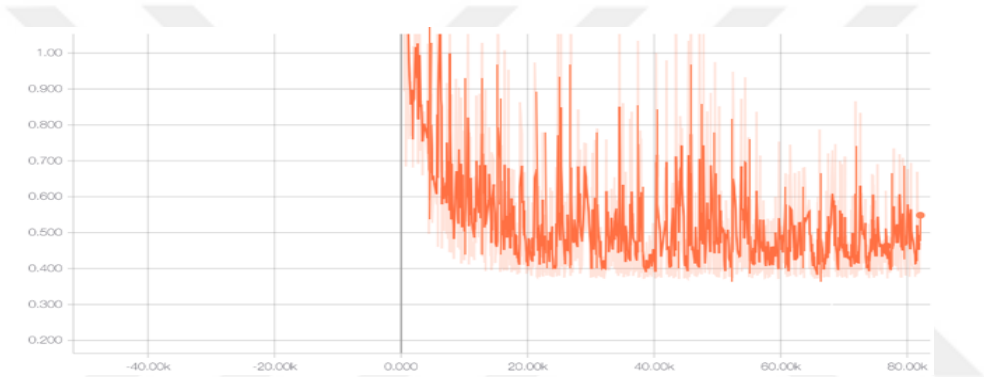


Figure 4.8 : Loss function for Inception-v4.

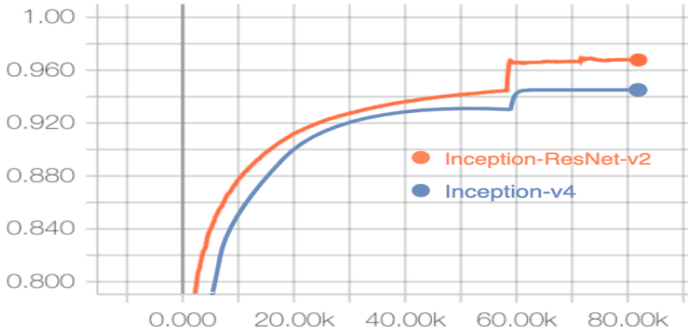


Figure 4.9 : Training accuracy of the networks.

Loss function of the networks are visualized with TensorBoard app built inside the Tensorflow library. X-axis shows the step size and Y-axis shows the value of the loss function. Loss functions show considerable amount of oscillations and many sub-optimal local minimas. However this fluctuation is expected because of the varying

batch sizes. During experimentation with different size of batches, any value exceeding 20 led to memory error and interruption in training process. Hardware limitations such as memory capacity of the GPU is most likely the reason behind the system crash. Experiments show that 6 GB of GPU ram is the bare minimum standard for using deep learning applications. Below 4 GB of GPU memory Inception-ResNet-v2 architecture can not be trained.

Comparable experiment regarding to using %15 of the training data as validation is carried out and mentioned later in this section. Varying batch sizes are not included and fluctuations are minor. Positive effects of using varying batch sizes in training accuracy are observed with %2 differential in both networks in different experiments, however this increase could be from using additional training data in the first experiment.

Training accuracy of the networks are quite similar however difference can be seen in the validation accuracy. Inception-ResNet-v2 network yields %5 greater overall accuracy compared to Inception-v4. This differential can be attributed to residual layers of the prior network since the rest of the architecture is quite similar. Class by class analysis also shows that Inception-ResNet-v2 has better performance in terms of accuracy.

Due to their spatial and spectral complexity following land use classes yielded lower accuracy compared to other classes in both models. Industrial area and medium residential classes achieved %75 user's accuracy with Inception-ResNet-v2 model. On the other hand, Inception-v4 model achieved %68 and %70 user's accuracy for the classes respectively. In depth analysis for the false negative and false positive instances show intra-class mixture was present. Airport areas contained hangars and depots that often can be identified as industrial areas. Ground sampling distance of the training images were high and these discrepancies were not present, making it harder for the model for recognizing the actual class. As for the "medium residential" class, error instances often labeled as dense or sparse residential. Empirical evidence suggests tiling material and the inconsistent spaces between buildings also contributed to error rate. On the other hand, benchmark datasets should include distinctive attributes for determining dense, medium and sparse residential areas. As for the types and the total number of buildings for each class.

Land cover classes such as chaparral and meadow indicate lower accuracy than expected since the spectral complexity of these classes are much lower compared to land use classes. Inception-ResNet-v2 model display %80 and %76 user's accuracy for chaparral and meadow classes respectively. For Inception-v4 model, user's accuracy for the classes mentioned above were %77 and %62 respectively. Certain classes contain intra-class mixtures with these land cover areas such as sparse residential and golf course, resulting in false positive instances.

Intra-class variability often creates problems for accurate labelling. For instance an "intersection" scene containing a "tennis court" is labeled as a "tennis court" or a "parking lot" scene with a "freeway" nearby is labeled as "freeway". This occurs for the instances that a certain class contain features that represent another class, not because of the poor generalization of the features.

"Storage tank" class is the most accurate class for both networks along with "forest" class. Results indicate that the error instaces belonging to those classes are not because of intra-class variability but rather spatial and spectral similarities to other classes. For example meadow class is often mislabeled as forest due to similar pixel values. For rare instances images belong to the "overpass" class labeled as "storage tank" due to similar round shape that is present in both of the classes. Another example of similar type of error instances are the mislabeled "river" and "chaparral" classes. Evidence suggests that factors contributing to these errors are stream or creek like shapes that can be found in both classes.

Results show that "terrace" and "farmland" have the worst accuracy rate compared to other classes. Reason being that the "terrace" and "farmland" classes share common spatial and spectral features that is almost indistinguishable for the human eye as well. Therefore, false negatives and false positives for these classes point each other.

On the other hand, experiments on the two networks display the importance of having residual connections for the CNN model. Results indicate residual connections improve training speed, achieving higher rate of accuracy with a lesser training time. As the model reaches close to a convergence, smaller learning rate is employed each time to further improve training accuracy.

There have been numerous studies regarding scene classification using NWPU-RESISC45 dataset. Experimental results carried out by Cheng et al[68] and Zhang et

al[71] with different networks on the same training dataset that is used in this thesis are given in (Table 4.5) and (Table 4.6). They have separated %10 and %20 of the training dataset to be used in validation respectively. Note that original NWPU-RESISC45 dataset consist of 45 classes.

Table 4.5 : Comparison of results with different networks.

CNN Models	%10 Training ratio	%20 Training ratio
Fine-tuned AlexNet	%81.22	%85.16
Fine-tuned VGGNet-16	%87.15	%90.36
Fine-tuned GoogLeNet	%82.57	%86.02

Table 4.6 : Comparison of the results with different networks.

CNN Models	%10 Training ratio	%20 Training ratio
AlexNet	%76.47	%79.79
VGGNet-16	%76.69	%79.85
GoogLeNet	%76.19	%78.48
VGG-16-CapsNet	%85.08	%89.18
Inception-v3-CapsNet	%89.03	%92.6

To further analyse the capability of the models used in this experiment in comparative manner and minimal bias, both of the networks are used in this thesis are trained with %15 training ratio of the NWPU-RESISC45 dataset. Which means using 11900 images for training and 2100 images for validation. Loss functions for the networks are given in (Figure 4.10) and (Figure 4.11).

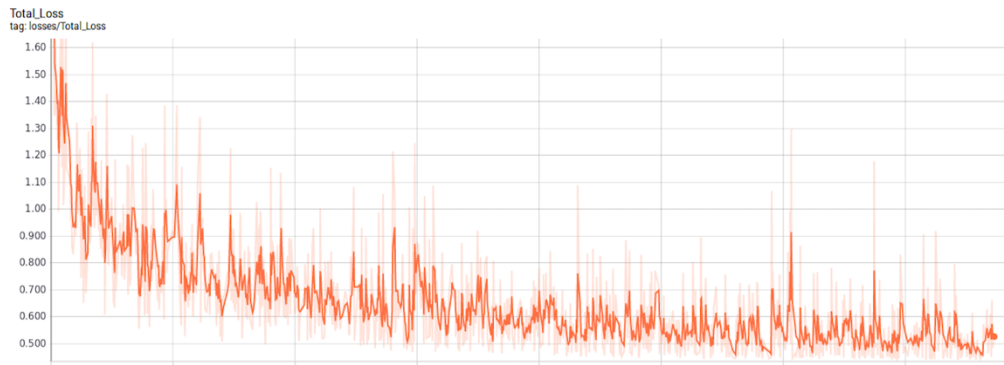


Figure 4.10 : Loss function for Inception-ResNet-v2 with %15 training ratio and no varying batch size.

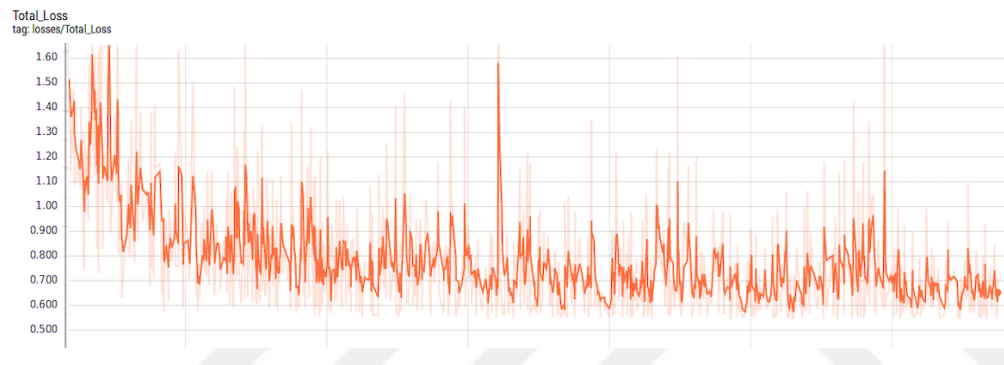


Figure 4.11: Loss function for Inception-v4 with %15 training ratio and no varying batch size.

Loss functions exhibit lesser oscillations due to the fact that varying batch size is not included in this training. However, final loss rate is higher for both networks. Training accuracy is %94 and %92 for Inception-ResNet-v2 and Inception-v4 respectively. Confusion matrices are given in (Figure 4.12) and (Figure 4.13). Results suggest that validation accuracy is much higher for both of the networks. Overall accuracy has increased %7 for both networks and up to %20 of increase can be seen for individual classes such as “terrace” and “farmland”. One would anticipate that using test images with similar characteristics as the training images increase overall accuracy of the network. It is observed that the data supports this hypothesis. Despite increasing the size of the validation dataset results show that networks perform better overall. Analysis regarding to the accuracy and the error instances of each class are as follows.

		Reference Data																						
Classified Image		Airport	Chaparral	Dense Residential	Forest	Freeway	Golf Course	Ground Track Field	Industrial Area	Intersection	Meadow	Medium Residential	Overpass	Parking Lot	Rectangular Farmland	River	Runway	Sparse Residential	Storage Tank	Tennis Court	Terrace	Classified Totals	Users' Accuracy	
	Airport	94				2		1	1	2						1	4						105	%89
	Chaparral		97		3						1							1			3		105	%92
	Dense Residential			93	1				2	1		7									1		105	%88
	Forest				98	1	2				3				2								106	%92
	Freeway				1	90	1		2	3			3	2				3					105	%85
	Golf Course						97	3			2							2					104	%93
	Ground Track Field			2				95			2				2						3	1	105	%90
	Industrial Area	2		2		2		1	90	2		2							1	2			104	%86
	Intersection	1				1				95		2	3	1						1	1		105	%90
	Meadow				3		3	2			92			1	2				2				105	%87
	Medium Residential			6		2			2			90		1					2		2		105	%85
	Overpass	2				4				4			95										105	%90
	Parking Lot					2			2				2	99									105	%94
	R. Farmland	2			2							3			91	2						6	106	%85
	River				2		1				2				2	94						3	104	%90
	Runway	5				2									1		94		2		1		105	%89
	Sparse Residential		3	1							1	3							95		1	1	105	%90
	Storage Tank			1	1		2		3											98			105	%94
	Tennis Court						2	3		1	1	1									98		106	%92
Terrace	2	4												7							92	105	%87	
Reference Totals	108	104	105	111	106	108	105	102	108	107	105	103	104	107	97	101	103	103	106	107	2100			
Producers' Accuracy	%87	%93	%88	%88	%84	%89	%90	%88	%87	%85	%85	%92	%95	%85	%96	%93	%92	%95	%92	%85				
Overall Accuracy : 1887/2100 = %89.85																								

Figure 4.12 : Confusion matrix for the Inception-ResNet-v2 with %15 training ratio of NWPU-RESISC45 dataset.

		Reference Data																						
Classified Image		Airport	Chaparral	Dense Residential	Forest	Freeway	Golf Course	Ground Track Field	Industrial Area	Intersection	Meadow	Medium Residential	Overpass	Parking Lot	Rectangular Farmland	River	Runway	Sparse Residential	Storage Tank	Tennis Court	Terrace	Classified Totals	Users' Accuracy	
	Airport	90				3		2	3	2						2	4					1	107	%84
	Chaparral		91		4						3							2				5	105	%86
	Dense Residential			86	2				5	1		9									1		104	%82
	Forest				94	2	3				5				2								106	%86
	Freeway				2	84	2		3	4			3	3			3						104	%80
	Golf Course						95	3			3							4					105	%90
	Ground Track Field			3				91			3				4						3	2	106	%85
	Industrial Area	4		3		3		1	84	2		4						2	2				105	%80
	Intersection	2				2				90		3	5	1						1	1		105	%85
	Meadow				3		4	4			87			1	2				3				104	%83
	Medium Residential			7		3			4			86		1					3		2		106	%81
	Overpass	3				4				6			90										103	%87
	Parking Lot					2			4				4	95									105	%90
	R. Farmland	3			4										84	3						8	106	%79
	River				2		2				3				3	90						4	104	%86
	Runway	6				4									1		91		2		2	106	%85	
	Sparse Residential		4	2							2	5							90		1	2	106	%84
	Storage Tank			1	1		3		4											94			103	%90
	Tennis Court						3	4		1	2	1									95		106	%89
	Terrace	3	3												12							86	104	%83
	Reference Totals		111	98	102	112	7	112	105	107	106	112	108	102	101	108	95	98	104	99	103	110	2100	
	Producers' Accuracy		%81	%92	%84	%83	%78	%84	%86	%78	%84	%77	%79	%88	%94	%77	%94	%92	%86	%94	%92	%78		

Overall Accuracy : 1793/2100 = % 85.38

Figure 4.13 : Confusion matrix for the Inception-v4 with %15 training ratio of NWPU-RESISC45 dataset.

Least improved class over the previous experiment is the “airport” class. Only %2 increase in accuracy can be seen. High ground sampling distance for these images cause problems in terms of intra-class variability and the spaces between runways resemble farmlands which contributes to false negative instances. Also other error instances point out that classes “freeway” and “river” are labeled as “airport” for several occasions which is not present in the previous experiment. This is likely due to high ground sampling distances of the “airport” images.

“Chaparral” class shows significant improvement over the previous experiment for both networks. Increase up to %12 in accuracy can be seen. Intra-class variability for the test images of this class were lower and the error instances belong to classes “meadow”, “sparse residential” and “terrace” which is the same as the previous experiment.

One of the lowest class accuracy is obtained for the “dense residential” class. Yet there is still improvement over to the first training. It is observed that the roof material and the spaces between buildings have a significant impact on the performance of this class. Hence, the majority of the error instances belong to classes “medium residential” and “industrial area”. Same issue is persistent in “medium residential” class. Although there is %10 increase in accuracy in both of the networks compared to the previous experiment.

“Parking lot” class exhibit the highest accuracy reaching up to %94 for Inception-ResNet-v2 network. This result can be attributed to low ground sampling distance of the “parking lot” images in NWPU dataset, making it easier for the model to recognize car patterns. Error instances are for the scenes that have intra-class variability such as parking lots near overpass and freeways.

For classes “farmland” and “terrace” same problem as the previous experiment is recurrent. Although there is progress, due to high ground sampling distance as well as parallel spectral and spatial characteristics make it harder for the models to distinguish these classes. Same issue can be addressed for classes “tennis court” and “ground track field” as the error instances for these classes point each other.

Land cover classes such as “forest” and “meadow” does not show significant improvement. Spatial and spectral properties aren’t that complex to take advantage of the similar test images as the training dataset.

4.1.7 Discussion

The proposed approach in this thesis has shown that it can be applied to remote sensing applications for automated land cover and land use classification from VHR images. The approach demonstrated that networks trained on an unrelated image recognition task can actually be used to solve the land cover and land use classification problem. One would anticipate that a large amount of VHR spatial imagery that already exists and that continues to be collected at higher rates will have a significant impact on a variety of remote sensing applications. Both of the experiments carried out in this thesis show accuracies that are at par with the state-of-the-art accuracies on the land use land cover classification problem. Adapting a deep pre-trained network and fine-tuning the network on a new dataset that has a limited number of labeled images to train quickly, learn and adjust the weights and biases of the network on the new dataset in effect delivers promising results.

Main goal of the experiments in this thesis is to assess the performance of the given neural networks by providing validation data which has distinct characteristics compared to trained data. Second experiment is carried out to display the performance of the networks by feeding validation data that is similar to trained data yet completely unseen. Results indicate that even with moderate size training data, generalizability of the features extracted from the networks are reliable over the unseen data with different characteristics. However, important caveat of this analysis is that intra-class variability needs to be addressed when creating a validation dataset. Experiments show that networks are more sensitive to the intra-class mixtures with validation data that has different characteristics as the training data. Also ground sampling distances for each class need to be defined for both validation and training datasets to further improve the results.

Fine-tuning is a major aspect of training a successful deep learning model. There various parameters to be controlled to achieve the maximum results out of the networks. Although most of it are based on trial and error method, varying batch-size during training proven to be an effective method to increase training accuracy while the network converges to a fixed number of training accuracy. First experiment shows that this method improves training accuracy. However, causing loss function to oscillate to various local minimas.

5. CONCLUSIONS

5.1 Summary

Accessibility of the remote sensing imagery have become increasingly available in the recent years. For various environmental and urban planning problems analysis of these remote sensing imagery have crucial importance. Increase in the available remote sensing imagery, calls for an automatization for the process of land cover and land use classification.

For remote sensing image analysis, CNN's provide reliable results that can be generalized over the unseen data. Furthermore, feature engineering and expertise needed for traditional feature extraction methods are eliminated. Training a successful CNN model requires sufficient amount of training data and fitting selection of hyperparameters. Training deep network architectures from scratch requires significant amount of data and training time in order to learn low-level features. Therefore, using pretrained networks for any given classification task is a good idea. One of the main advantages of CNN's is final layers of the network can be fine-tuned for a specific goal including remote sensing image analysis.

In this study, two state-of-the-art pre-trained networks namely Inception-ResNet-v2 and Inception-v4 are trained for the purpose of land cover and land use classification. In both experiments, training dataset is created from NWPU-RESISC45 dataset which consist of 20 classes with 700 image. Images converted to the binary format of .tfrecords to minimize memory usage and loading time. In order to test the generalizability of features that are created by the networks, separate validation dataset is used for the first experiment. A validation dataset is created from Worldview-3 satellite images as to feed networks with images of different characteristics. Training accuracy of the networks Inception-ResNet-v2 and Inception-v4 for the first experiment are %97.7 and %94.2 respectively. Total training time for the networks are 13 and 11 hours respectively. Each network is trained for 100 epochs using decaying learning rate and various batch sizes to increase training accuracy. Validation dataset

consists of patches extracted from 30 cm 3-band RGB satellite imagery. These imagery are acquired from Worldview-3 satellite sensor with 3 distinct AOI's including Las Vegas, Shanghai and Paris. Validation accuracy for the networks are %82.8 and %77.8 respectively. Various metrics for evaluation and comparison of these networks are calculated such as precision recall and f1-scores. Results indicate that Inception-ResNet-v2 model outperforms Inception-v4 in terms of accuracy and generalization over the unseen data.

Second experiment is carried out to compare the performance of the networks by feeding validation data with similar characteristics as the training data. Both networks are trained with %15 training ratio of the NWPU dataset. Results indicate %89.95 and %85.38 validation accuracy for the Inception-ResNet-v2 and Inception-v4 networks respectively.

5.2 Conclusions

Results of the experiments in this thesis show that CNN's are powerful classification tools that can be effectively used in remote sensing problems such as land cover and land use classification. CNN models eliminate feature engineering expertise required by the traditional techniques and automatize the process. However, training deep networks can be a challenging task. Proper selection of hyperparameters and sufficient amount of training data are key factors of successfully training a CNN model.

Transfer learning approach demonstrated in this experiment shows that networks trained on an unrelated image recognition task can be used to solve the land cover and land use classification problem. Fine-tuning CNN's with pre-trained weights provide accurate results with unseen data.

Classes that are selected for the experiments have mutual properties and intra-class variability in order to further investigate the models ability to classify distinct scenes and assess the results with minimal bias. Results indicate that intra-class variability is the main reason for false classification, rather than the similar properties of the selected classes. However, "terrace" and "farmland" classes exhibit poor results compared to other classes. It is observed that the spatial attributes of these classes are identical and may require additional training data to further generalize on the unseen data.

5.3 Future Works

Deep learning algorithms such as CNNs show impressive results in both 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 for land cover and land use classification, experiments in this thesis indicate multi-label approach should be investigated. Due to intra-class variability of the scenes models can label features that belong to another class that are located in a small part of the image. This problem also brings up the question of whether to treat land cover land use classes as individual objects. Automatization of land cover and land use classification from a larger image that contains multiple classes remains as a challenge that needs to be re-visited in the future.



REFERENCES

- [1] **Mountrakis, G., Im, J. and Ogole, C.** (2011). “Support vector machines in remote sensing” A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(3), pp.247-259.
- [2] **Watanachaturaporn, P., Arora, M.K., Varshney, P.K.,** (2008). “Multisource classification using support vector machines: an empirical comparison with decision tree and neural network classifiers.” *Photogrammetric Engineering & Remote Sensing* 74 (2), 239–246.
- [3] **Albert, L., Rottensteiner, F., Heipke, C.,** (2017). “A higher order conditional random field model for simultaneous classification of land cover and land use.” *ISPRS Journal of Photogrammetry and Remote Sensing* 130: 63-80.
- [4] **Hermosilla, T., Ruiz, L. A., Recio, J. A., Cambra-López, M.,** (2012). “Assessing contextual descriptive features for plot-based classification of urban areas.” *Landscape and Urban Planning*, 106(1): 124-137.
- [5] **Camps-Valls, G., Gomez-Chova, L., Munoz-Mari, J., Vila-Frances, J., Calpe-Maravilla, J.,** (2006). “Composite kernels for hyperspectral image classification.” *IEEE Geoscience and Remote Sensing Letters* 3 (1), 93–97.
- [6] **LeCun Y., Bottou L., Bengio, Y. and Haffner, P.,** (1998) “Gradient-based learning applied to document recognition” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324.
- [7] **Krizhevsky, A., Sutskever I., Hinton, G. E.,** (2012). “ImageNet classification with deep convolutional neural networks”. In: *International Conference on Neural Information Processing Systems (NIPS'12)* 25 Vol. 1, pp. 1097-1105.
- [8] **Zeiler, M. D., Fergus, R.,** (2014). “Visualizing and understanding convolutional networks,” in *European Conf. on Computer Vision*, pp. 818–833, Springer.
- [9] **Szegedy, C. et al.,** (2015). “Going deeper with convolutions,” in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1–9.
- [10] **He, K., Zhang, X., Ren, S., Sun, J.,** (2016). “Deep residual learning for image recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778.
- [11] **Szegedy, C. et al.,** (2017). “Inception-v4, Inception-ResNet and the impact of residual connections on learning”. In *AAAI*, pp. 4278–4284.
- [12] **Marmanis, D., Datcu M., Esch T., et al.,** (2016). “Deep learning earth observation classification using ImageNet pretrained networks”, *IEEE Geoscience and Remote Sensing Letters* 13(1), 105–109.

- [13] **Donahue, J., Jia Y., Vinyals O. et al.**, (2014). “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition,” *Icml* 32, 647–655.
- [14] **Salberg, A.B.**, 2015. “Detection of seals in remote sensing images using features extracted from deep convolutional neural networks,” in *Proc. IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS)*, , pp. 1893–1986.
- [15] **Othman, E., Bazi, Y. Alajlan N. et al.**, (2016). “Using convolutional features and a sparse autoen- coder for land-use scene classification,” *International Journal of Remote Sensing* 37(10), 1977–1995.
- [16] **Url-1** <<http://vision.ucmerced.edu/datasets/landuse.html>>, date retrieved 16/04/2019
- [17] **Iftene, M., Liu, Q., Wang, Y.** (2016). “Very high resolution images classification by fine tuning deep convolutional neural networks,” in *Eighth International Conference on Digital Image Processing (ICDIP 2016)*
- [18] **Sheng, G., Yang, W., Xu, T., & Sun, H.** (2012). “High-resolution satellite scene classification using a sparse coding based multiple feature combination.” *International Journal of Remote Sensing*, 33(8), 2395–2412.
- [19] **Ghazi, M. M., Yanikoglu B., Aptoula E.** (2017). “Plant Identification Using Deep Neural Net- works via Optimization of Transfer Learning Parameters”, *Neurocomputing*.
- [20] **Lee, H., Kwon H.** (2016). “Contextual Deep CNN Based Hyperspectral Classification”, in *2016 IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, 1604.03519, 2–4.
- [21] **Fisher, P.** (1997). “The Pixel: A Snare and a Delusion”, *International Journal of Remote Sensing*, 18: 679-685.
- [22] **Rollet, R., Benie G.B., Li, W., Wang, S., Boucher, J.M.** (1998). “Image Classification Algorithm based on the RBF Neural Network and K-means”, *International Journal of Remote Sensing*, 19: 3003-3009.
- [23] **Shalaby, A., Tateishi R.** (2007). “Remote Sensing and GIS for Mapping and Monitoring Land Cover and Land-use Changes in the Northwestern Coastal Zone of Egypt”, *Applied Geography*, 27: 28-41.
- [24] **Atkinson, P.M.** (2004). “Spatially weighted supervised classification for remote sensing”, *International Journal of Applied Earth Observation and Geoinformation*, 5: 277–291.
- [25] **Zhu, H.W., Basir, O.** (2005). “An Adaptive Fuzzy Evidential Nearest Neighbor Formulation for Classifying Remote Sensing Images”, *IEEE Transactions on Geoscience and Remote Sensing*, 43: 1874-1889.
- [26] **Kulkarni, A.D., Kamlesh, L.** (1999). “Fuzzy Neural Network Models for Supervised Classification: Multispectral Image Analysis”, *Geocarto International*, 4: 42-51.
- [27] **Yuan, F., Sawaya, K.E., Loeffelholz, B.C., Bauer, M.E.** (2005). “Land Cover Classification and Change Analysis of the Twin Cities (Minnesota) Metropolitan Area by Multitemporal Landsat Remote Sensing”, *Remote Sensing of Environment*, 98: 317-328.

- [28] **Tang J., Wang L., Myint S.W.** (2007) “Improving Urban Classification through Fuzzy Supervised Classification and Spectral Mixture Analysis”, *International Journal of Remote Sensing*, 28; 4047-4063.
- [29] **Myint, S.W., Gober P., Brazel, A., Grossman-Clarke, S., Weng, Q.** (2011) “Per-pixel vs. Object-based Classification of Urban Land Cover Extraction using High Spatial Resolution Imagery”, *Remote Sensing of Environment*, 115: 1145-1161.
- [30] **Jackson, Q., Landgrebe D.A.** (2002). “Adaptive Bayesian Contextual Classification Based on Markov Random Fields”, *IEEE Transactions on Geoscience and Remote Sensing*, 40: 2454-2463.
- [31] **Lu D., Weng Q.** (2007). – “Survey of Image Classification Methods and Techniques for Improving Classification Performance”, *International Journal of Remote Sensing*, 28: 823-870.
- [32] **Zhang L., Zhang L., Kumar V.** (2016). “Deep learning for remote sensing data,” *IEEE Geosci. Remote Sens. Mag.* 4, 22–40.
- [33] **Elmannai, H., Loghmari, M.A., Naceur, M. S.** (2013) “Support Vector Machine for Remote Sensing image classification” International Conference on Control, Engineering & Information Technology (CEIT'13) Proceedings Engineering & Technology - Vol.2, pp.68-72.
- [34] **Huang, C., Song, K., Kim, S., Townshend, J.R.G., Davis, P., Masek, J.G., Goward, S.N.,** (2008). “Use of a dark object concept and support vector machines to automate forest cover change analysis”, *Remote Sensing of Environment* 112 (3), 970–985.
- [35] **Li, H., Gu, H., Han, Y., Yang, J.,** (2010). “Object-oriented classification of high-resolution remote sensing imagery based on an improved colour structure code and a support vector machine”, *International Journal of Remote Sensing* 31 (6), 1453–1470.
- [36] **Brenning, A.,** (2009). “Benchmarking classifiers to optimally integrate terrain analysis and multispectral remote sensing in automatic rock glacier detection”, *Remote Sensing of Environment* 113 (1), 239–247.
- [37] **Otukei, J R., Blaschke, T.,** (2010). “Land cover change assessment using decision trees, support vector machines and maximum likelihood classification algorithms”, *Int. J. Appl. Earth Obs. Geoinf.* 12 S27–S31.
- [38] **Punia, M, Joshi, P K and Porwal, M C.,** (2011) “Decision tree classification of land use land cover for Delhi, India using IRS-P6 AWiFS data”, *Expert Syst. Appl.* 38(5) 5577–5583.
- [39] **Pal, M., Mather P M.** (2003). “An assessment of the effectiveness of decision tree methods for land cover classification”, *Remote Sens. Environ.* 86 554–565.
- [40] **Chan, J.C.-W., Paelinckx, D.,** (2008). “Evaluation of Random Forest and Adaboost tree-based ensemble classification and spectral band selection for ecotope mapping using airborne hyperspectral imagery”, *Remote Sensing of Environment* 112 (6), 2999–3011.

- [41] **Ghimire, B., Rogan, J., Miller, J.,** (2010). “Contextual land-cover classification: incorporating spatial dependence in land-cover classification models using random forests and the Getis statistic”, *Remote Sensing Letters* 1, 45–54.
- [42] **Sesnie, S., Gessler, P., Finegan, B., Thessler, S.,** (2008). “Integrating Landsat TM and SRTM-DEM derived variables with decision trees for habitat classification and change detection in complex neotropical environments”, *Remote Sensing of Environment* 112 (5), 2145–2159.
- [43] **Waske, B., Braun, M.,** (2009). “Classifier ensembles for land cover mapping using multitemporal SAR imagery”, *ISPRS Journal of Photogrammetry and Remote Sensing* 64 (5), 450–457.
- [44] **Prasad, A.M., Iverson, L.R., Liaw, A.,** (2006). “Newer classification and regression tree techniques: bagging and random forests for ecological prediction”. *Ecosystems* 9 (2), 191–199.
- [45] **Lawrence, R., Wood, S., Sheley, R.,** (2006). “Mapping invasive plants using hyperspectral imagery and Breiman Cutler classifications (RandomForest)”, *Remote Sensing of Environment* 100 (3), 356–362.
- [46] **Verbeke, L.P.C.; Vancoillie, F.M.B.; De Wulf, R.R.,** (2004). “Reusing back-propagation artificial neural networks for land cover classification in tropical savannahs”. *Int. J. Remote Sens.* 2004, 25, 2747–2771.
- [47] **Yuan, H., Van Der Wiele, C.F., Khorram, S.** (2006). “An Automated Artificial Neural Network System for Land Use/Land Cover Classification from Landsat TM Imagery”, *Remote Sens.* 2009, 1, 243-265.
- [48] **Cao, J., Chen, Z., Wang, B.** (2016). “Deep convolutional networks with superpixel segmentation for hyperspectral image classification,” in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 3310–3313.
- [49] **Fang, Z., Li, W., Du, Q.** (2016). “Using CNN-based high-level features for remote sensing scene classification,” in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 2610–2613
- [50] **Gong, M., Zhou Z., and Ma J.** (2016). “Change detection in synthetic aperture radar images based on deep neural networks,” *IEEE Trans. Neural Networks Learn. Syst.* 27(1), 125–138.
- [51] **Yang, J. et al.,** (2016). “Hyperspectral image classification using two-channel deep convolutional neural network,” in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 5079–5082.
- [52] **Kussul, N. et al.,** (2016). “Deep learning approach for large scale land cover mapping based on remote sensing data fusion,” in *IEEE Geoscience and Remote Sensing Symp. (IGARSS '16)*, pp. 198–201, IEEE.
- [53] **Guanetal, H.** (2015). “Deep learning based tree classification using mobile LiDAR data, ”*Remote Sens. Lett.* 6(11), 864–873.

- [54] **Li, P. et al.**, (2016). “Road network extraction via deep learning and line integral convolution,” in IEEE Int. Geoscience and Remote Sensing Symp. (IGARSS '16), pp. 1599–1602, IEEE.
- [55] **Tang, J. et al.**, (2015). “Compressed-domain ship detection on spaceborne optical image using deep neural network and extreme learning machine,” IEEE Trans. Geosci. Remote Sens. 53(3), 1174–1185.
- [56] **Li, W. et al.**, (2016). “Stacked autoencoder-based deep learning for remote-sensing image classification: a case study of African land-cover mapping,” Int. J. Remote Sens. 37(23), 5632–5646.
- [57] **Li, J.**, (2016). “Active learning for hyperspectral image classification with a stacked autoencoders based neural network,” in IEEE Int. Conf. on Image Processing (ICIP '16), pp. 1062–1065.
- [58] **Li, T., Zhang J., Zhang Y.** (2014). “Classification of hyperspectral image based on deep belief networks,” in IEEE Int. Conf. on Image Processing (ICIP '14), pp. 5132–5136.
- [59] **Diao, W. et al.**, (2015). “Object recognition in remote sensing images using sparse deep belief networks,” Remote Sens. Lett. 6(10), 745–754.
- [60] **Zhao, Q. et al.**, (2015). “Three-class change detection in synthetic aperture radar images based on deep belief network,” in Bio-Inspired Computing-Theories and Applications, pp. 696–705, Springer
- [61] **Hochreiter, S., Schmidhuber, J.** (1997). “Long Short-Term Memory” Neural Computation 9:8, 1735-1780.
- [62] **Sharma A., Liu X., Yang X.** (2018)., “Land cover classification from multi-temporal, multi-spectral remotely sensed imagery using patch-based recurrent neural networks”, Neural Networks, Volume 105, 346-355.
- [63] **Simonyan,K.Zisserman.**, (2014). “A Very deep convolutional networks for large-scale image recognition”. In Proc. International Conference on Learning Representations.
- [64] **Duchi, J., Hazan, E., and Singer, Y.** (2011). “Adaptive subgradient methods for online learning and stochastic optimization,” J. Mach. Learn. Res. 12, 2121–2159.
- [65] **Tielemanand, T., Hinton, G.** (2012). “Lecture6.5 rmsprop:divide the gradient by a running average of its recent magnitude,” COURSERA: Neural Networks Mach. Learn. 4(2), 26–31.
- [66] **Kingma, D., Ba J.** (2014). “Adam: a method for stochastic optimization,” arXiv:1412.6980.
- [67] **Ioffe, S., Szegedy C.** (2015). “Batch normalization: accelerating deep network training by reducing internal covariate shift,” arXiv:1502.03167.
- [68] **Cheng, G., Han, J., Lu, X.** (2017). “Remote Sensing Image Scene Classification: Benchmark and State of the Art”. *Proceedings of the IEEE*, 105(10): 1865-1883.

- [69] **Abadi, M., Agarwal A., Barham P.,... Yu Y, Zheng X.** (2015). “TensorFlow: Large-scale machine learning on heterogeneous systems,” Software available from tensorflow.org.
- [70] **Smith, S. L., Kindermans, P.J., Ying, C., Le, Q. V.,** (2018). “Don’t decay the learning rate, increase the batch size”, ICLR.
- [71] **Zhang, W., Tang, P., & Zhao, L.** (2019). Remote Sensing Image Scene Classification Using CNN-CapsNet. *Remote Sensing*, 11(5), 494.



CURRICULUM VITAE



Name Surname : **Berk GÜNEY**
Place and Date of Birth : **Fatih 23/04/1994**
E-Mail : **berk__guney@hotmail.com**

EDUCATION :

- **B.Sc.** : 2016, Kadir Has University, Faculty of Engineering and Natural Sciences, Computer Engineering.
- **M.Sc.** : 2019, Istanbul Technical University, Informatics Institute, Satellite Communication and Remote Sensing.