

NILAY TUĞÇE SÜBERK

DEEP LEARNING TECHNIQUES FOR BUILDING DENSITY
ESTIMATION FROM REMOTELY SENSED IMAGERY

M.S. Thesis

NILAY TUĞÇE SÜBERK

2019

IŞIK UNIVERSITY
2019

DEEP LEARNING TECHNIQUES FOR BUILDING DENSITY
ESTIMATION FROM REMOTELY SENSED IMAGERY

NİLAY TUĞÇE SÜBERK

B.S., Electrical & Electronic Engineering, IŞIK UNIVERSITY, 2015

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electronics Engineering

IŞIK UNIVERSITY

2019

IŞIK UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

DEEP LEARNING TECHNIQUES FOR BUILDING DENSITY
ESTIMATION FROM REMOTELY SENSED IMAGERY

NİLAY TUĞÇE SÜBERK

APPROVED BY:

Prof. Dr. Hasan Fehmi ATEŞ Medipol University
(Thesis Supervisor)



Prof. Dr. Bahadır K. GÜNTÜRK Medipol University



Assist. Prof. Boray TEK Işık University



APPROVAL DATE:

05/04/2019

DEEP LEARNING TECHNIQUES FOR BUILDING DENSITY ESTIMATION FROM REMOTELY SENSED IMAGERY

Abstract

This thesis is about point-wise estimation of building density on the remote sensing optical imagery by applying deep learning methods. The goal of the project is to reduce mean square error of the estimated density by applying architectural modifications on the deep learning network and using augmented training data.

Recently, deep learning is one of popular field of science and convolutional neural networks (CNNs) are well-known deep neural network. Recent studies indicate that some of the convolutional neural networks are highly effective in large scale image works such as recognition, semantic segmentation. There has been limited research in using deep networks to learn urbanization characteristics from remote sensing images. Remote sensing images could be used for regression problems and building density estimation is one of them. Building density information provides knowledge for real estate agents and urban planners, estimating disaster risk areas, environment protection and resource allocation. Our method provides a cheap and fast solution to these needs when there is no cadastral information.

The main objective of this thesis is to achieve fast and accurate local building density estimation using high resolution remote sensing images. Deep learning methods based on CNN are applied in this project. Pre-trained visual geometry group (VGG-16) and fully convolutional network (FCN) are tested as convolutional neural network. We tested three different modified networks and then applied data augmentation in the train data to reduce mean square error value. The networks that we have performed simplified original VGG-16 network for regression, VGG-16 network with sigmoid layer added and simplified VGG-16 network with sigmoid layer. The best result (lowest mean square error) is obtained from sigmoid layer added VGG-16 network with data augmentation. Sigmoid layer added VGG-16 network gives us ($\sim 0,084$) RMSE on building density estimation with the augmented train dataset. Original VGG-16 network gives ($\sim 0,105$) RMSE, sigmoid layer added VGG-16 network gives ($\sim 0,095$) RMSE and sigmoid

layer added simplified VGG-16 network gives ($\sim 0,090$) RMSE on building density estimation with the small train dataset. FCN is one of the ideal network for classification tasks so we have also applied fully convolutional network result to compare our results with its result. We have modified the network to perform building density estimation in addition to semantic segmentation. The root mean square error of FCN is ($\sim 0,084$) and our best result (lowest mean square error) is also ($\sim 0,084$) RMSE at the same iteration number.

Our results show that fast and accurate building density estimation is possible by using vanilla CNNs. Sigmoid layer addition, simplification of the network for small dataset and data augmentation improves accuracy in the regression. Data augmentation is the most effective method to reduce RMSE in this thesis.

Keywords: mean square error, data augmentation, deep learning, image regression, cnn models

UZAKTAN ALGILANAN GÖRÜNTÜLERDE BİNA YOĞUNLUĞU TAHMİNİ İÇİN DERİN ÖĞRENME TEKNİKLERİ

Özet

Bu tez, derin öğrenme yöntemlerini uygulayarak uzaktan algılamalı optik görüntülerde bina yoğunluğunun noktasal olarak tahmin edilmesi ile ilgilidir. Projenin amacı, derin öğrenme ağına mimari değişiklikler uygulayarak ve artırılmış eğitim verilerini kullanarak tahmini yoğunluğun ortalama kare hatasını azaltmaktır.

Son zamanlarda, derin öğrenme popüler bilim alanlarından biridir ve evrişimli sinir ağları (CNN) iyi bilinen derin sinir ağlarından biridir. Son çalışmalar, evrişimli sinir ağlarının bazılarının tanıma, anlamsal bölümlendirme gibi büyük ölçekli görüntü çalışmalarında oldukça etkili olduğunu göstermektedir. Uzaktan algılama görüntülerinden kentleşme özelliklerini öğrenmek için derin ağları kullanma konusunda sınırlı araştırma yapılmıştır. Regresyon problemleri için uzaktan algılama görüntüleri kullanılabilir ve bina yoğunluğu tahmini bunlardan biridir. Bina yoğunluğu bilgisi, emlakçılar ve şehir plancıları için bilgi sağlar, afet risk alanlarını, çevre koruma alanlarını ve kaynak tahsisini tahmin eder. Metodumuz kadastro bilgisi olmadığında bu ihtiyaçlara ucuz ve hızlı bir çözüm sunar.

Bu tezin temel amacı, yüksek çözünürlüklü uzaktan algılama görüntüleri kullanılarak hızlı ve doğru yerel bina yoğunluğu tahmini elde etmektir. Evrişimsel sinir ağına (CNN) dayalı derin öğrenme yöntemleri bu projede uygulanmaktadır. Önceden eğitilmiş görsel geometri grubu (VGG-16) ve tamamen evrişimsel sinir ağı (FCN) ile testler uygulanmıştır.

Biz üç farklı değiştirilmiş ağı test ettik ve ardından ortalama kare hata değerini azaltmak için veri artırma yöntemine başvurduk. Uyguladığımız ağlar, regresyon için basitleştirilmiş orjinal VGG-16 ağı, sigmoid katmanı ekli VGG-16 ağı ve sigmoid katmanı ekli basitleştirilmiş VGG-16 ağıdır. En iyi sonuç ise (en düşük ortalama kare hatası) eğitim veri kümesi artırılmış ve sigmoid katmanı eklenmiş VGG-16 ağından elde edilir. Sigmoid katmanı eklenmiş VGG-16 ağı, artırılmış eğitim veri setiyle bina yoğunluğu tahmininde ($\sim 0,084$) kök ortalama kare hatası

verir. Orjinal VGG-16 ađı bize ($\sim 0,105$) kk ortalama kare hatası, sigmoid katmanı eklenmiř VGG-16 ađı bize ($\sim 0,095$) kk ortalama kare hatası ve sigmoid katmanı eklenmiř - basitleřtirilmiř VGG-16 ađı, kçük eđitim veri kmesi ile bina yođunluđu tahmininde bize ($\sim 0,090$) kk ortalama kare hatası verir. Tamamen evriřimsel sinir ađı (FCN), sınıflandırma grevleri iin ideal ađlardan biridir. Bu yzden tamamen evriřimsel sinir ađı (FCN) sonucunu, sonularımız ile karřılařtırma amacıyla projemize ekledik. Anlamsal blmlendirmeye ek olarak, ađı bina yođunluđu tahmini yapması iin deđiřtirdik. Aynı yineleme sayısında, tamamen evriřimsel sinir ađın (FCN) kk ortalama kare hatası ($\sim 0,084$) tr ve bizim en iyi sonucumuz da ($\sim 0,084$) kk ortalama kare hatasına eřittir.

Sonularımız, hızlı ve dođru bina yođunluđu tahmininin vanilya evriřimsel sinir ađlarını (CNN) kullanarak mmkn olduđunu gstermektedir. Sigmoid katmanının eklenmesi, ađın kçük veri kmesi ve veri bytme iin basitleřtirilmesi, regresyondaki dođruluđu arttırır. Veri bytme, bu tez alıřmasında kk ortalama kare hatasını azaltmada en etkili yntemdir.

Anahtar kelimeler: ortalama karesel hata, veri bytme, derin đrenme, grnt regresyonu, cnn modelleri

Acknowledgements

The research work was supported by Coordination Office for Scientific Research Project, Işık University, Project Number: 10244269, Project Title: Deep Learning Techniques for Building Density Estimation from Remotely Sensed Imagery.

First of all, Special thanks to Prof .Dr. Hasan Fehmi ATEŞ who is my project supervisor for always supporting me with his extensive knowledge in the all parts of the project.

Also, I want to thank to Alper Acabey who is my best friend and always makes me feel never alone with his real support, help and patience during the academic years.

Finally, I thank my lovely parent Aynur Süberk who always supports me with her love and patience.



To My Family...

Table of Contents

Abstract	ii
Özet	iv
Acknowledgements	vi
List of Tables	x
List of Figures	xi
List of Abbreviations	xiii
1 Introduction	1
1.1 Remote Sensing	3
1.2 Deep Learning	4
1.3 Related Work	6
1.4 Organization of Thesis	9
2 Convolutional Neural Network	10
2.1 Batch Size and Learning Rate	15
2.1.1 Stochastic Gradient Descent (SGD)	17
2.2 Convolutional Neural Network Layer Patterns Structure of CNN	17
3 Network Architecture and Implementation Details	21
3.1 Dataset	22
3.2 The Architecture: VGG-16 Convolutional Neural Network	24
3.3 Fully Convolutional Network (FCN)	28
3.4 Data Augmentation	29
3.5 Simplifying The Architecture	30
4 Experimental Work	31
4.1 Experiment with Original VGG-16 Network	32
4.2 Experiment with Sigmoid Layer	35
4.2.1 Experiment with Sigmoid Layer and Data Augmentation	37
4.3 Experiment with Simplified Architecture and Sigmoid Layer	38

4.4 Comparing the Results	41
5 Conclusion	54
Reference	56



List of Tables

3.1	Layers of Our VGG-16 Network	25
4.1	Classes of the Data	32
4.2	Layers of VGG-16 Network	33
4.3	Result of Original Network	33
4.4	Last Layers of the Network	35
4.5	Result of Sigmoid Layer + Original Network	35
4.6	Result of Sigmoid Layer + Data Augmentation	37
4.7	Layers of Simplified VGG-16 Network	39
4.8	Result of Simplified Network + Sigmoid Layer	40
4.9	All Experiment Results	41
4.10	Predicted and Real Values of Experiment 1(E1)	46
4.11	Predicted and Real Values of Experiment 2(E2)	46
4.12	Predicted and Real Values of Experiment 3(E3)	47
4.13	Predicted and Real Values of Experiment 4(E4)	47
4.14	Differences between Predicted and Real Values	47
4.15	Image Patches That Have Either Small or Large Estimation Errors	47
4.16	Predicted and Real Values of Experiment 1(E1)	51
4.17	Predicted and Real Values of Experiment 2(E2)	52
4.18	Predicted and Real Values of Experiment 3(E3)	52
4.19	Predicted and Real Values of Experiment 4(E4)	52
4.20	Differences between Predicted and Real Values	52
4.21	Image Patches That Have Either Small or Large Estimation Errors	53

List of Figures

1.1	General Overview of the Experiments	3
1.2	Neural Networks: They are organized in layers with set of nodes [12]	5
1.3	Mathematical Model of Neuron [11]	5
2.1	Architecture of LeNet [19]	11
2.2	Architecture of AlexNet [19]	11
2.3	Architecture of VGG-Network [19]	12
2.4	Process of Convolutional Neural Network [20]	13
2.5	Fully Connected Network [20]	14
2.6	Batch and Mini Batch: According to batch and mini batch imple- mentations, the variation of error value [22]	16
2.7	Learning Rate Effect On the Gradient Descent [23]	16
2.8	Stride of 2 Pixels and Kernel Size (filter) 3x3 Pixels [27]	18
2.9	Virtual Effect of ReLu [29]	19
2.10	Rectified Linear Unit Function [30]	19
2.11	Dropout [31]	20
2.12	Sigmoid Activation Function [23]	20
3.1	Steps of Experimental Work	22
3.2	Example of Original Training Data	23
3.3	Example of Original Test Data	23
3.4	Example of Training Data Which Is Cropped and Its Building La- bels (226x226)	23
3.5	Example of Test Data Which Is Cropped and Its Building Labels (226x226)	24
3.6	Original VGG-16 Network [33]	27
3.7	Modified VGG-16 Network For Regression	27
3.8	Sigmoid Layer Added VGG-16 Network	27
3.9	Simplified VGG-16 Network	27
3.10	Fully Convolutional Neural Network Flow [34]	28
3.11	The Modification On the FCN	29
4.1	Iteration Number vs RMSE	34
4.2	Predicted and Real Values vs Test Data Numbers with Original Network	34
4.3	Iteration Number vs RMSE	36

4.4	Predicted and Real Values vs Test Data Numbers with Sigmoid Layer	36
4.5	Iteration Number vs RMSE	37
4.6	Predicted and Real Values vs Test Data Numbers with Sigmoid Layer and Augmented Dataset	38
4.7	Iteration Number vs RMSE	40
4.8	Predicted and Real Values vs Test Data Numbers of Simplified Network with Sigmoid Layer	41
4.9	All Experiments Iteration Number vs RMSE	42
4.10	All Experiment Heat-Map Results For Test Data (0005) : E1, E2, E3, E4, respectively	43
4.11	Original Test Image and Its Semantic Segmentation Map	44
4.12	Real Heat Map (0005)	44
4.13	Difference Heat-Maps of Test Data (0005) : E1, E2, E3, E4, respectively	45
4.14	All Experiment Heat-Map Results For Test Data (0030): E1, E2, E3, E4, respectively	48
4.15	Original Test Image and Its Semantic Segmentation Map	49
4.16	Real Heat Map (0030)	50
4.17	Difference Heat-Maps of Test Data (0030): E1, E2, E3, E4, respectively	50

List of Abbreviations

BCR	B uilding C overage R atio
CAFFE	C onvolutional A rchitecture for F ast F eature E MBEDDING
CNN	C onvolutional N eural N etwork
FCN	F ully C onvolutional N etwork
MSE	M ean S quare E rror
NN	N eural N etwork
RMSE	R oot M ean S quare E rror
SGD	S tochastic G radient D escent
VGG	V isual G eometry G roup
VHR	O ptical V ery H igh R esolution

Chapter 1

Introduction

World population and economy have been growing day by day and it causes high rate of urbanization. Fast urbanization brings with it great urban expansion that causes problems such as urban congestion, air pollution and urban heat island [1]. Therefore, urban extent, human settlements and urban building density have become more crucial for urban planning and environmental management. In addition, buildings are the places that all humans live and conduct their activities. It means that buildings are also significant indicator of population distribution [2].

The ratio of the coverage of the buildings which is called building density is a fundamental urban environmental parameter and it is more effectual at estimating disaster risk than building/non-building dichotomy [1]. Several studies show that building density effects the wind conditions, the access of sunlight and solar radiation, the interior temperatures of buildings, surface of thermal conditions etc. Thus, building density is essential information for empirical and scientific exploration and social problems [3]. In addition, building density has great interest from many stakeholders such as real estate agents and urban planners. It plays a guiding role in many aspects such as city planning, land management, environment protection and resource allocation [1], [4].

In developing countries, the information of building density is often unavailable or incomplete; also field investigation and manual depiction is time consuming,

and not suitable for data updating. For these necessities, building density information is needed for the related research community and high resolution remote sensing data provide an efficient way to collect the information of buildings [1], [5]. Fortunately, high resolution remote sensing data is the information source which is economically and commercially accessible.

This thesis is about pointwise estimation of building density on the remote sensing optical imagery by applying deep learning methods. A well-known Convolutional Neural Network architecture is used to train and then test the dataset. Every test image is assigned a building density value that is estimated from regression result of the deep network. We have divided our original data into train and test sets. After that, network learned building density information from train data then computed pointwise estimation for every test data. In the end of the test, we have combined pointwise estimates of a region together to see the regional density in the form of a heat-map representation. General overview of the experiments is shown in the Figure 1.1. We have used pre-trained network and applied fine-tuning methods on the network to reach lowest mean square error.

The main objective of this thesis is to show improvements in mean square error by applying deep learning methods. This thesis presents data augmentation method and sigmoid activation function layer to reduce root mean square error. The proposed method achieves satisfactory results for building density estimation in terms of the low root mean square error ($\sim 0,084$). We have tested how different modifications to network architecture and augmenting train data affects the regression performance. We have also compared our results to the results obtained by building segmentation using a well-known deep semantic segmentation network.

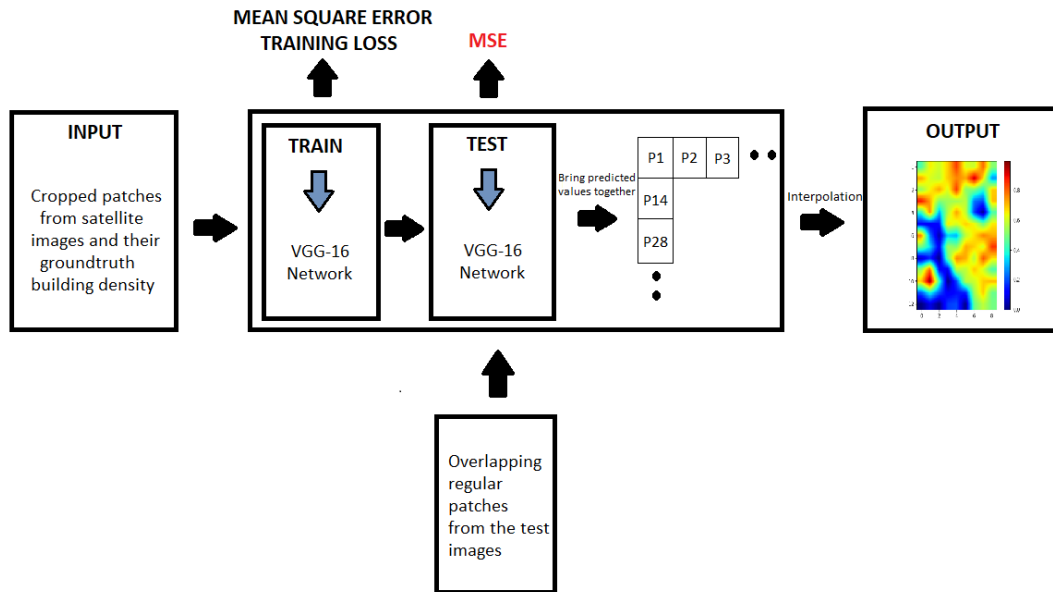


Figure 1.1: General Overview of the Experiments

1.1 Remote Sensing

Remote sensing refers to the technology of getting information about the surface of earth (land and ocean) and remote sensing images have form of the digital images [6]. Surface altitude samples that are taken from airborne scanning are much more reliable and accurate in the estimation problems when it is compared the traditional photogrammetric techniques [3]. In data collection tasks, remote sensing techniques are becoming more important and these data are used by governments, researchers etc [7]. Some of the techniques of remote sensing are Synthetic Aperture Radar (SAR), The Light Detection and Ranging (LiDAR) and Optical Very High Resolution (VHR).

Synthetic Aperture Radar (SAR) can be defined as a powerful earth observation tool for large scale application and the images that are taken from it can be used to estimate building density [1] [4]. Their polarization information or textural features help to estimate building density but complex scattering mechanism in urban areas could be reason of reducing the estimation accuracy [1]. The Light Detection and Ranging (LiDAR) remote sensing technology is an invention for

surveying and mapping buildings in urban environments. This technique also allows for an accurate delineation of the footprints of buildings [3]. However, with the improvement of remote sensors, Optical Very High Resolution (VHR) remotely sensed imagery contributes rich geospatial details.

In this respect VHR aerial and satellite images are useful in detecting buildings [8] and rich geospatial details provide new possibility for building density estimation [3]. One of the basic method for building density estimation is based on building detection. The features that are procured from the VHR imagery can supply a sophisticated description of buildings and building clusters, which provides more accurate building density information [1].

1.2 Deep Learning

In recent years, many domains of science, business and government use deep learning technology in the world. Deep learning is the field of artificial intelligence and computers can learn many things by using large amounts of data without human involvement. The algorithms of deep learning inspired by the human brain, human brain learns from experience, deep learning algorithm would perform a task repeatedly and each time evolves the outcome by using given data [9].

Deep learning is characterized by neural networks (NNs) and it usually has more than two hidden layers, so they are called 'deep' [10]. Basic neural network is shown in the Figure 1.2 and each connection between neurons is associated with a weight such as w_{ij} , w_{jk} , w_{kl} . Each neuron performs dot product with its input matrices that contain all pixels of the images and its weight matrices. Then neuron sums all dot product results and adds bias. After that, each neuron applies the non-linearity which is also called activation function [11]. For our case activation function is ReLu which will be explained. Figure 1.3 shows the mathematical model of neuron.

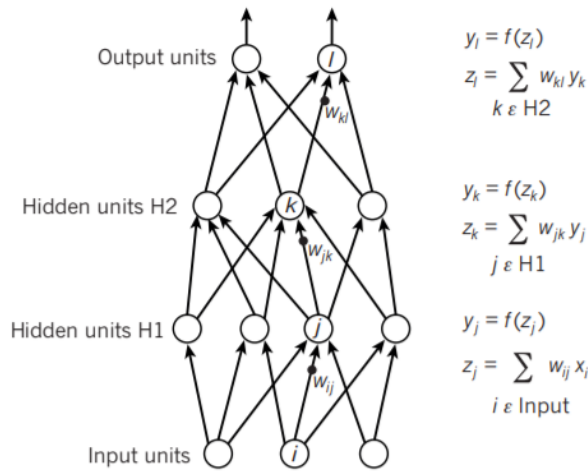


Figure 1.2: Neural Networks: They are organized in layers with set of nodes [12]

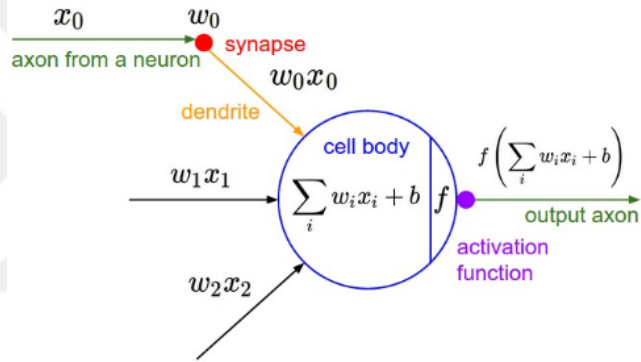


Figure 1.3: Mathematical Model of Neuron [11]

Deep Neural Networks make use of feature representations which are learned from data instead of handcrafting features which are designed domain-specific knowledge [10].

Deep learning methods are used in target recognition and it can be divided into two main categories: supervised and unsupervised methods. Unsupervised method learns features from the input data but it does not know any correlated label or information [7]. Supervised method has prior knowledge, uses given sample data to make best approximates that between input and output data. Supervised method or learning is also divided into two categories that are classification and

regression. Classification predicts the class the data belongs to and regression predicts the numeric value based on the observed data which is trained data [13].

Deep learning has won remarkable success and its neural networks are powerful learning machines. It generalizes to different domains, for example, semantic segmentation model performs classifying pixels, the same model that is trained on different data can be used to predict oil fault lines and detect cars etc [14]. High resolution satellite images are one of the resources and they have been an active research topic in the last decades. Increasing availability of data and computational resources provide the use of deep learning in remote sensing [10]. Deep learning is being used to solve problems in the field of urban planning and computational sustainability. For instance, popular deep neural network which is convolutional neural network (CNN) has been used to estimate spatial distribution of poverty. CNN, which will be explained in next chapters, shows that it is effective at the problem of remote sensing image scenes classification [15].

In this thesis, The Visual Geometry Group (VGG)- convolutional neural network which is well-known deep learning neural network architecture is used in regression for building density estimation in the test data with our modifications on the network.

1.3 Related Work

Remote sensing is the technology that has large bulk of images of the earth is accessible nowadays and can be applied for building detection, segmentation and density estimation. Learned features from deep convolutional networks are successfully applied for building segmentation [16]. However, there has been limited research in using deep networks to learn urbanization characteristics such as building density, from remote sensing images [16].

Yu [3] proposed a new object based method that extract automatically building objects and it computes building density with the dataset that is taken from airborne LiDAR. Their object based method and software tool applies three different scales to extract building objects and calculates building density:

- First, some geometric and volumetric features are calculated to describe shape and size of the buildings
- Second, Building Coverage Ratio (BCR) and Floor Area Ratio (FAR) are computed and mapped at land lot scale by associating with land lots through topological operation
- Third, to describe 3-dimension quantitative definition of urban district, density attributes are calculated at urban district scale.

This approach-software tool can be used for urban planning and management [3] because BCR map shows planimetric and horizontal density of buildings, FAR map shows 3D density of building.

In work of Qiong Wu [4], urban building density is estimated by using high resolution Synthetic Aperture Radar (SAR) images. The applied building detection algorithm methods are empirical threshold method and fmax-filter algorithm [4]. Empirical threshold estimates threshold values to the experimental data and it based on the concept of a permutation test [17]. Fmax-filter algorithm characterizes urban and non-urban areas using its local histogram [4]. They compare the differences between two methods to detect effective method for urban building density information.

In work of Yi Zhou [2], building density is estimated by applying CART algorithm and integration of SAR and optical data. CART is the classification and regression tree based approach and it can handle both classification and regression tasks. If the land has spectral signatures with the building areas, separation of them is tough task when only using optical data. Thus, they have applied integration

of two types of data. In the result of this study proposed that the data which are taken from different sensors and satellites have potential to improve building density estimation performance.

Tao Zhang [1] proposed building density estimation by using multiple features and support vector regression that is the algorithm for regression tasks. Optical very high-resolution (VHR) remotely sensed images are used as dataset. Three categories of features (hand-crafted) are taken into to compute the estimate building density and best regression performance (lowest root mean square (RMSE) is obtained when three categories of features are applied at the same time:

- *Spectral feature* belongs to remote sensing imagery as a fundamental property. It exploits from spectral differences between building and vegetation (non-building).
- If the buildings have suitable conditions, *morphological features* have the ability of describe the spectral-structural characteristics of buildings such as size, contrast.
- In high resolution images, spectral features have been found to be insufficient for the distinction of spectrally similar classes. Therefore, *textural features* used as a third category of feature and it is effective in describing building clusters to estimate building density.

We have applied Optical very high-resolution (VHR) remotely sensed images to compute best regression performance (lowest root mean square error (RMSE)) as in the previous related work. While Zhang [4] applied support vector regression algorithm, we have applied convolutional neural network to obtain the density estimation. In addition, we have computed pointwise density estimates in local regional windows while the last related work used the whole image as input to the classifier in order to obtain a density heat-map at the output.

1.4 Organization of Thesis

This thesis is organized as following steps; In Chapter 2 we will explain Convolutional Neural Network, its history and layer patterns, gradient descent and hyperparameters of the network. Then implementation details, dataset, VGG-16 network architecture and fully convolutional network will be explained with details in Chapter 3. Our experiment results and their details will be in Chapter 4. We conclude our work and discuss the future works in the Chapter 5.



Chapter 2

Convolutional Neural Network

There are some models of deep neural networks and one of the commonly used model is Convolutional Neural Networks also known as CNN or ConvNet. CNN made up of neurons as neural network and these neurons have learnable biases and weights that are used to get less error in the result [18].

In addition, there are several popular state-of-the-art CNN architectures and they are made of a key set of basic layers. These layers include convolution layers, soft-max or Euclidean loss layer, fully connected layers, etc. Every different architecture has its different combination and some of the popular architectures are LeNet, AlexNet, VGGNet. Their state-of-the-art performances are more efficient on different benchmarks. [19]

- LeNet (1998) algorithm is proposed in the 1990s but in those days there were limited computation and memory. Therefore, it is implemented around 2010 with proposed CNN and back-propagation algorithm [19]. Also known as LeNet-5 is experimented on handwritten digits dataset and its basic configuration is shown Figure 2.1

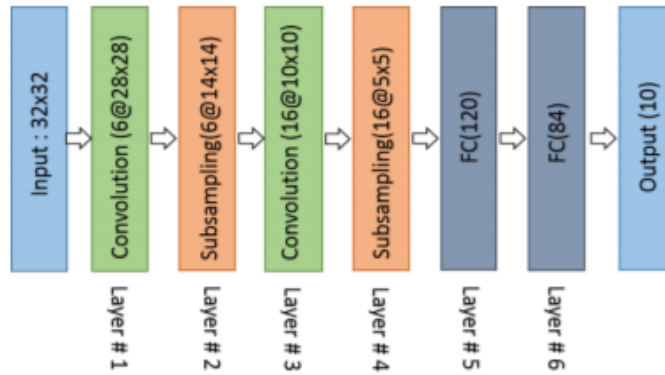


Figure 2.1: Architecture of LeNet [19]

- AlexNet is the deeper and wider CNN model compared to LeNet. It is proposed by Alex Krizhevsky in 2012 and AlexNet won the most difficult ImageNet challenge for visual object recognition which is called the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. AlexNet was a significant breakthrough for visual recognition and classification tasks and it achieved state-of-the-art recognition. There are two main new concepts which is different from LeNet,,: Local Response Normalization (LRN) and dropout layers [19]. Its basic configuration is shown Figure 2.2.

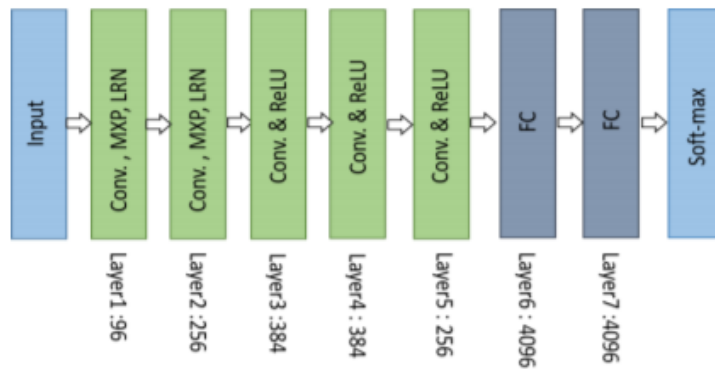


Figure 2.2: Architecture of AlexNet [19]

- The Visual Geometry Group (VGG) Network: This group is the runner up of 2014 ILSVRC and they showed that to achieve better recognition or classification, accuracy depends on the depth of a network. VGG-Net includes convolutional layers, activation function layers that are called ReLu,

max pooling layers, several fully-connected layers and the final layer which is softmax or euclidean loss layer. VGG-Net has, VGG-11, VGG-16 which we have used and VGG-19 models. They have same three fully connected layers but their convolution layer numbers are different. VGG-16 has 13, VGG-19 has 16 convolutional layers and VGG-19 is most computational expensive model [19]. Its basic building blocks configuration is shown Figure 2.3.

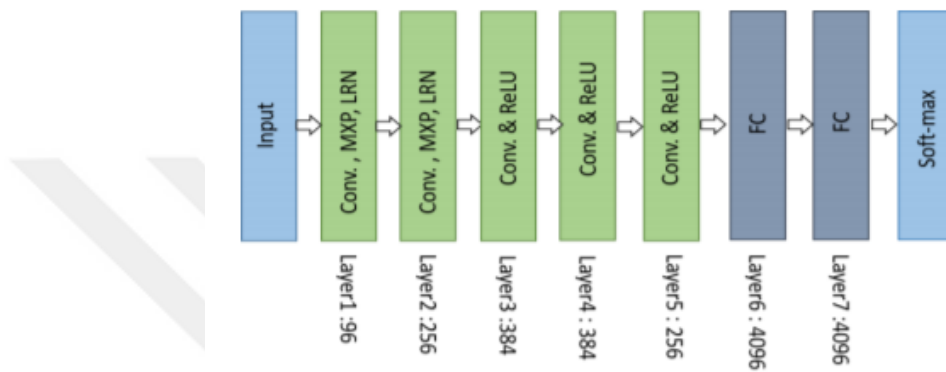


Figure 2.3: Architecture of VGG-Network [19]

ConvNet architectures are more effective when images are used as their inputs because it allows to encode certain properties into the architecture. Ordinary neural network receives a single vector as an input then series of hidden layers are transformed. All neurons of the previous layer are fully connected to next each neuron. The last layer that is called output layer is fully connected layer and class scores is represented with it. Full image is not scaled by using regular neural networks.

CIFAR-10 is the computer vision dataset used for object recognition. According to previous experiences in this dataset, the images that have the size $32 \times 32 \times 3$ (wide, high, 3 color channels) has $32 * 32 * 3 = 3072$ weights in its single fully connected neuron in a first hidden layer. Networks have several such neurons and this size seems manageable. However, if the image has the size $200 \times 200 \times 3$, it means that $200 \times 200 \times 3 = 120,000$ weights. These weights with several neurons make these full connections wasteful in the fully connected structure. As a result

of this, overfitting occurs with huge number of parameters [18]. This is why CNN is chosen as a neural network when image is chosen as an input.

The process of CNN starts off with an input image then applies filters (also known as weights or feature detector) to the image. This is called convolutional layer process and then feature maps are created as shown in the Figure 2.4. After every convolutional layer, to break up the linearity, activation function is used for every feature maps [20]. We have applied ReLu as an activation function which will be explained in the next section. The image becomes ready for pooling step and its aim is providing spatial invariance to the convolutional neural network. As a pooling, we have used max-pooling that will be also explained in the next section. Then, pooled feature map goes for flattening step that is input layer for the artificial neural network [20].

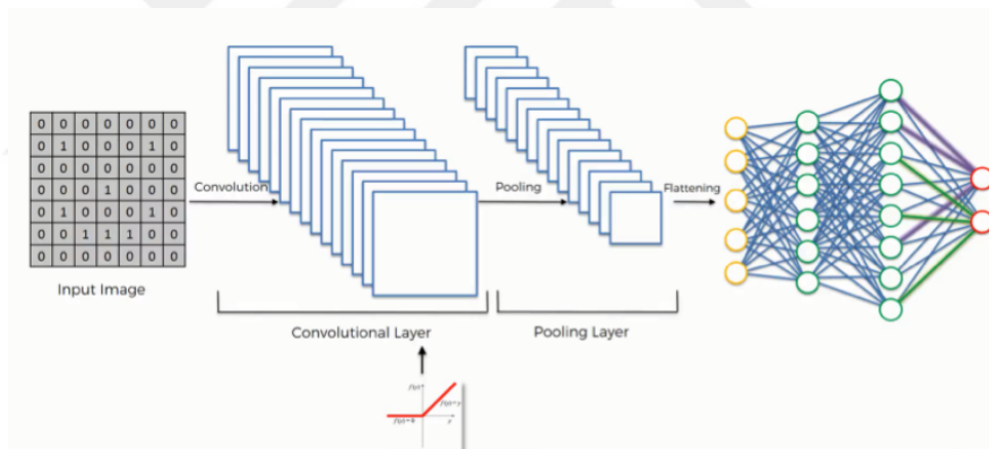


Figure 2.4: Process of Convolutional Neural Network [20]

After the flattening step, it ends up with a long vector of data then pass through the artificial neural network to process further. As shown in the Figure 2.5, the full connection step has three layers;

1. Input Layer
2. Fully Connected Layer

3. Output Layer

The role of this network is that taking data and combining the features into a wider diversity of attributes. By applying this step, convolutional neural network will have the ability to classify images, that is the aim of convolutional neural network [20].

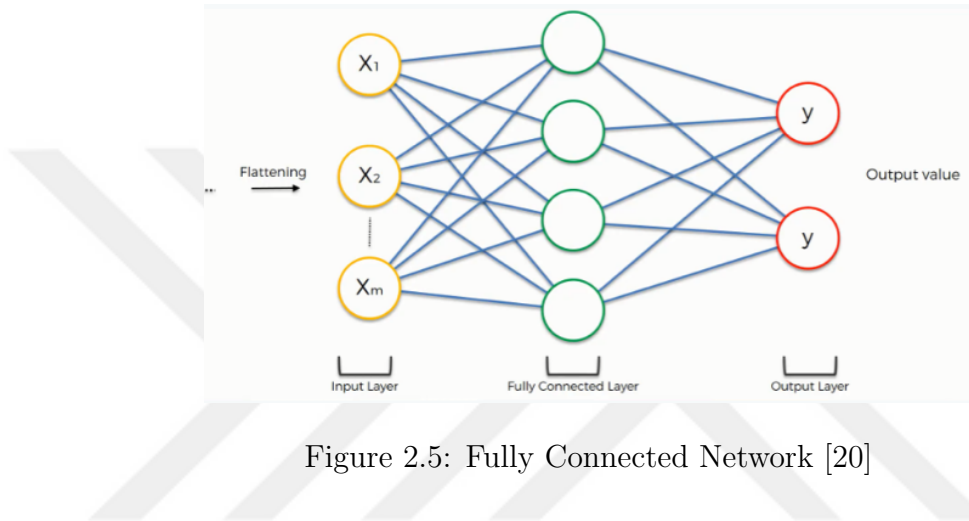


Figure 2.5: Fully Connected Network [20]

There are two main process in fully connected networks : forward pass and backward pass. Forward pass is a set of operations that transform network input into output area. Every neuron uses ReLu as an activation function that brings non linearity into the process [21]. The output depends on the last layer of network, for example, softmax layer is for classification and euclidean loss layer is for regression. Softmax assigns the probabilities for each class in the multi class tasks and euclidean loss assigns the square root of euclidean distance.

After all of the calculations, it end up with an output. However, the result can be improved by updating weights and biases of the neuron. This is called backpropagation and it is an algorithm that computes error gradients with respect to the neuron weights and biases. Before calculating error gradients, error should be computed and it is called loss function.

The loss function could be any differentiable mathematical expression such as cross entropy, root mean square error. Updating weights and biases in the opposite direction of the gradient reduces the error, so gradient descent algorithm is used in the network. To sum up, running gradient descent algorithm multiple times improves the result/s [21].

There are some gradient descent hyper-parameters that affect the loss function such as batch size and learning rate. They do not have specific values and their values are determined from earlier experiences. Their values are selected by considering system properties and time.

2.1 Batch Size and Learning Rate

Network has the parameters that can be changed or modified such as batch size and learning rate. They determine some behaviors of the convolutional neural network and they have not exact value to reduce error.

In the deep learning applications, it is hard to learn all the data which are in the training dataset at the same time. The reason of it is that learning all data at the same time needs more time and memory. Batch size is one of the hyper-parameter and determines that how many data are being process at the same time in the training. There are three ways to choose batch size of the network:

- Gradient Descent: batch size equals to number of train dataset (n)
- Stochastic Gradient Descent: batch size equals to 1
- Mini Batch Gradient Descent: Batch size has a value that between 1 and n

Mini batch gradient descent causes the oscillations on the error values as in the shown in the Figure 2.6. The reason of it is that different data processes in every iteration. It causes while the given parameters appropriate for chosen data, same given parameters could not be appropriate for different data [22].

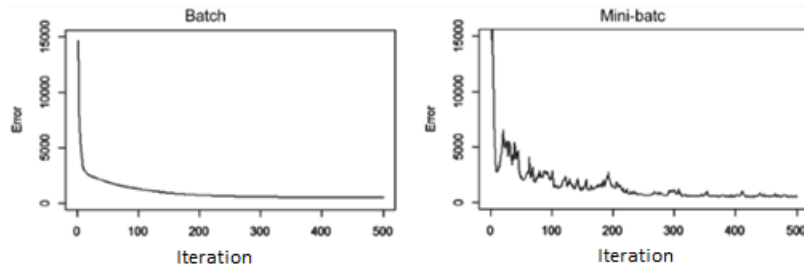


Figure 2.6: Batch and Mini Batch: According to batch and mini batch implementations, the variation of error value [22]

Learning rate is a hyper-parameter and controls the weight of our network according to the lost gradient. It effects gradient descent as shown in the Formula (2.1). It determines how quickly a network updates its parameters. If learning rate has a too small value, the gradient descent can be slow but if has a too large value, the gradient descent can overshoot the minimum. This means that it may not convergence [23] as shown in the Figure 2.7. In addition, small learning rate value causes longer training process.

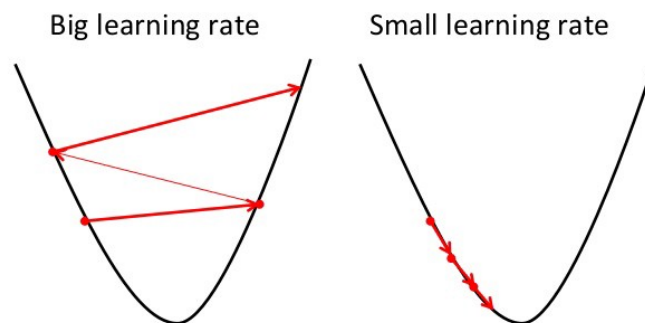


Figure 2.7: Learning Rate Effect On the Gradient Descent [23]

The solver orchestrates model optimization by using running gradient descent algorithm. CAFFE which is the deep learning framework has caffe solvers such as Stochastic Gradient Descent (type: SGD), Adam (type Adam), AdaDelta (type: AdaDelta). We have applied SGD in our project .

2.1.1 Stochastic Gradient Descent (SGD)

Gradient is the slope of the error curve at the computed point and stochastic gradient descent is an algorithm that repeatedly run through the training set. The parameters are updated according to the gradient of the error with respect to single training example [24]. SGD updates the weights W and it uses a linear combination of the gradient ∇ with loss function $L(W_{t-1})$. The gradient has a weight that is called learning rate, α [25] and b refers to bias. The overall cost function depends on the mean cost function that calculated on all of the N training samples.

- The update weight W_t for SGD is computed by using the Formula 2.1 and Formula 2.2;

$$W_t = W_{t-1} - \alpha \nabla L(W_{t-1}, b) \quad (2.1)$$

$$L(W_{t-1}) = \frac{1}{N} \sum_{z=1}^N L(W_{t-1}, b, X^{(z)}, Y^{(z)}) \quad (2.2)$$

Where $X^{(z)}, Y^{(z)}$ are each training sample pairs, $\nabla L(W_{t-1})$ is slope of the cost function (mse) [26].

- For mini batch, loss function is computed by using the Formula 2.3;

$$L(W_{t-1}) = \frac{1}{bs} \sum_{z=1}^{bs} L(W_{t-1}, b, X^{(z)}, Y^{(z)}) \quad (2.3)$$

“ bs ” represents batch size of the network.

2.2 Convolutional Neural Network Layer Patterns Structure of CNN

As mention in previous chapter, convolutional neural network architecture generally stacks a few CONVOLUTION, RELU, DROP OUT, MAX-POOLING layers. CNN layers also have layer sizing patterns which are STRIDE, KERNEL SIZE

and ZERO-PADDING. The last layer is fully-connected layer and it holds the output score of classification or regression [18].

- STRIDE is a value for sliding the filter. The filter moves the one pixel at a time when it is given 1 or two pixels at a time when it is given 2 which is shown in Figure 2.8. It is commonly chosen 1 or 2 and rarely 3 because at a time jumping more pixels produce smaller output that is not recommended [18].
- KERNEL SIZE is the filter size of the convolution. Generally, it is chosen 3x3 with the padding is 1 which retain the original size of input. The Formula 2.4 shows the relationship between kernel size and padding [18];

$$P = \frac{(F - 1)}{2} \quad (2.4)$$

Where P is padding size and F is Kernel size

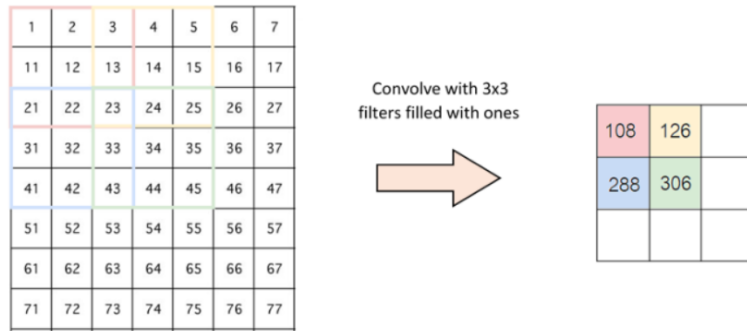


Figure 2.8: Stride of 2 Pixels and Kernel Size (filter) 3x3 Pixels [27]

- MAX-POOLING is one of the function that is used in our network because network needs to obtain a property that is known as spatial variance. The network has capable of detecting the object in the image by using this property [20] and POOLING layer down samples the volume spatially. Generally, its size is chosen 2x2 with a stride and kernel size of 2. It covers 2x2 matrix of the input then takes maximum value of the pixels that are

covered with it. Max pooling helps to extract to sharpest features of the image [18].

- *ZERO-PADDING* keeps the spatial sizes constant after every convolutional layer and padding size depends on the convolutional filter size [18]. Its aim is that protects information after applying filter on the image (convolutional layer). According to given value of the pad, it covers the input volume with zeros around the border.
- *ReLU* (rectified linear unit) is the nonlinear function or activation function which helps to become zero for negative values. It is used to provide the output is not simply a linear transformation of the input [28]. The virtual effect of ReLu is shown in the Figure 2.9 and function of it is shown in the Figure 2.10.

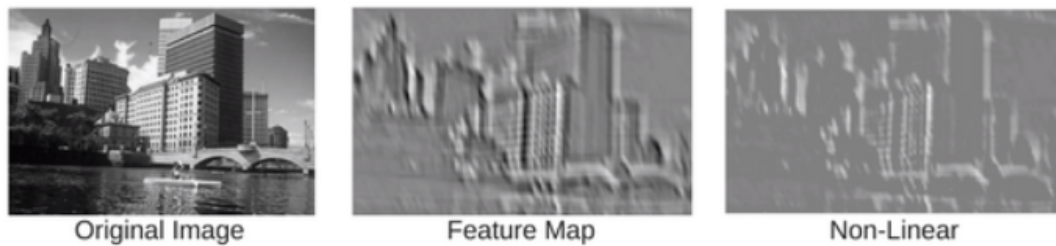


Figure 2.9: Virtual Effect of ReLu [29]

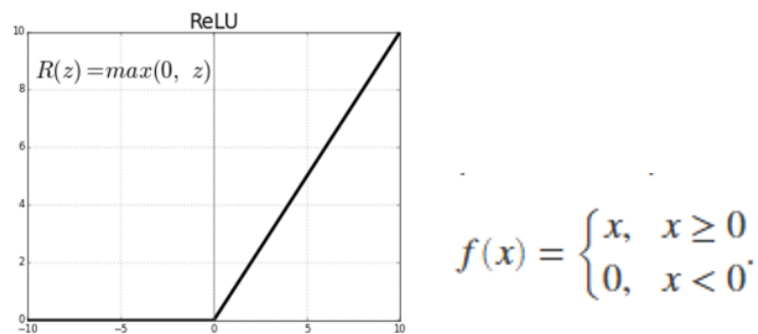


Figure 2.10: Rectified Linear Unit Function [30]

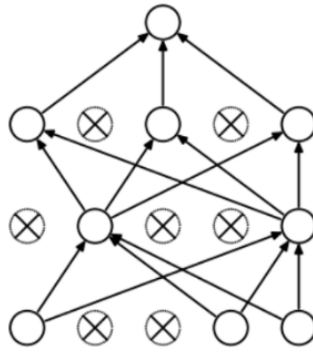


Figure 2.11: Dropout [31]

- *DROPOUT* is one of the technique to reduce overfitting. Dropout sets the output to zero according to dropout rate. For example, the rate of dropout 0.5 means that half of the neurons are randomly chosen and their outputs set to zero. A neuron cannot rely on the presence of other neurons, so it reduces co-adaptions between neurons and this is the how to reduce overfitting [28]. Those dropped neurons are not contributing forward pass and also backpropagation is not updated because of the same reason. By using dropped out, network is forced to learn more representative features [28].
- *SIGMOID* is one of the activation function that has the range (0,1). It makes binary predictions [23] and it can represent the probability of binary class. In addition, it provides stability in the network.

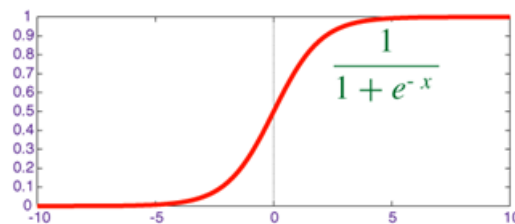


Figure 2.12: Sigmoid Activation Function [23]

Chapter 3

Network Architecture and Implementation Details

In this thesis, the aim is pointwise estimation of building density on the remote sensing optical imagery by applying deep learning methods. There are two stages to reach estimation result which are train and test. VGG-16 is a well-known convolutional neural network and we have used it in these two stages. However, we have fine-tuned on this network to improve our estimation result. Improving estimation result means that reducing mean square error in this work. Firstly, we have got original network result then added sigmoid layer on the network. Then we have simplified our network by reducing neuron number in the fully connected layers. Lastly, we have applied data augmentation on the data and applied our modified networks. Totally 16 different size high resolution images are used as dataset. They are from Vaihingen, Germany and 12 of them is used for training dataset, 4 of them is used for test dataset. Each satellite image is divided into sub-regions, building density is estimated locally in each sub-region, density estimates are combined and interpolated to form a density heat map for the whole region.

As in the mentioned above, the implementations that we have applied for experimental work are data augmentation and modifying / simplifying architecture. Experimental work steps are shown in the Figure 3.1.

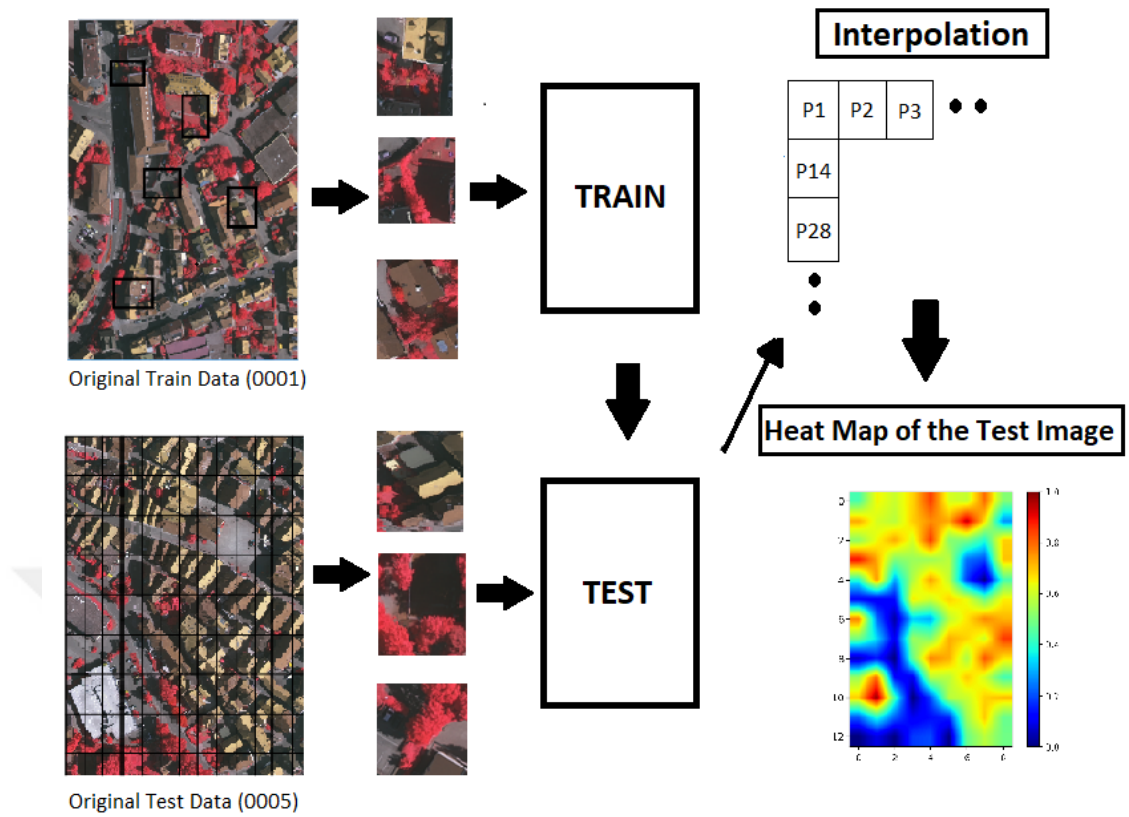


Figure 3.1: Steps of Experimental Work

3.1 Dataset

In this project, we have used remote sensing data which is very high resolution Vaihingen in Germany dataset and they are provided by German Association of Photogrammetry and Remote Sensing (DGPF) for International Society for Photogrammetry and Remote Sensing Organization (ISPRS). Area covers urban scenes and has relatively small village with many detached buildings and small multi story buildings [32]. This data is provided to us from ISPRS with their semantic segmentation files. Generally, this dataset has been used for classification in ISPRS, so their semantic segmentation has five classes that are surface, building, vegetation, tree and car. However, we are interested in only building class so we have adjusted the segmentation map that is able to show only buildings in the images.

We have sixteen original images and they are divided into two main parts; four of them as test data, twelve of them as train data. According to the pointwise estimation, we have cropped the original train and test images like in the Figure 3.4 and Figure 3.5.

In addition, we see the images with different from RGB color bands because RGB bands of the TIFF files correspond to near infrared, red and green bands delivered by the camera [32].



Figure 3.2: Example of Original Training Data



Figure 3.3: Example of Original Test Data

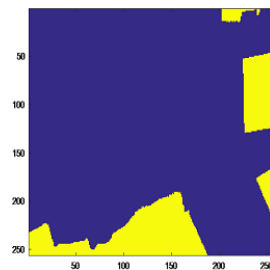


Figure 3.4: Example of Training Data Which Is Cropped and Its Building Labels (226x226)

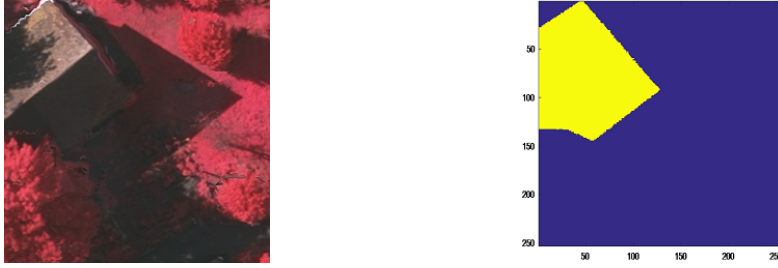


Figure 3.5: Example of Test Data Which Is Cropped and Its Building Labels (226x226)

Building density is also called building coverage ratio (BCR) refers to the ratio of the total standing area of buildings to the total area of the interest area. The way of the computing building density is calculating in a pixel unit with the idea of local processing [5]. It is actualized for every test data and calculated by using the Formula 3.1;

Real building density is calculated from semantic segmented images that are obtained from ISPRS with the same way;

$$Realbuildingdensity = \frac{Real\ building\ area(pixel)}{Real\ region\ Area\ (pixel)} \quad (3.1)$$

3.2 The Architecture: VGG-16 Convolutional Neural Network

The Visual Geometry Group (VGG) Network, this group runner up of the 2014 ILSVRC and we have applied VGG-16 CNN which is one of the network that has small error rate in classification tasks. In this thesis, our approach is that using this network for regression and test how successful in the regression task on the remote sensing images.

There are two network in our project and one of them is train network, other one is test network. They have all same network layers and our VGG-16 convolutional neural network structure is;

3.1;

data	(8L, 3L, 224L, 224L)
sem	(8L,)
conv1_1	(8L, 64L, 224L, 224L)
conv1_2	(8L, 64L, 224L, 224L)
pool1	(8L, 64L, 112L, 112L)
conv2_1	(8L, 128L, 112L, 112L)
conv2_2	(8L, 128L, 112L, 112L)
pool2	(8L, 128L, 56L, 56L)
conv3_1	(8L, 256L, 56L, 56L)
conv3_2	(8L, 256L, 56L, 56L)
conv3_3	(8L, 256L, 56L, 56L)
pool3	(8L, 256L, 28L, 28L)
conv4_1	(8L, 512L, 28L, 28L)
conv4_2	(8L, 512L, 28L, 28L)
conv4_3	(8L, 512L, 28L, 28L)
pool4	(8L, 512L, 14L, 14L)
conv5_1	(8L, 512L, 14L, 14L)
conv5_2	(8L, 512L, 14L, 14L)
conv5_3	(8L, 512L, 14L, 14L)
pool5	(8L, 512L, 7L, 7L)
fc6	(8L, 4096L)
fc7	(8L, 4096L)
fcB	(8L, 1L)
loss	()

Table 3.1: Layers of Our VGG-16 Network

- The first row represents the data which has 3 dimensions (NearInfraRed-Red-Green), 224x224 image size and 8 batch size
- The first convolution layer has 64 filters, 8 batch size and 224x224 image size

- After every convolutional layer, ReLu layer follows it to turn the negative values to zero to adding non-linearity in the network.
- In the pooling (max-pooling) layer, image size reduces by the ratio 1/2 that depends on the kernel size and stride. Therefore, pooling layers always have image data that has half size of the image in the convolution layer which comes before it.
- The layers that are represented by “ fc6 ” and “ fc7 ” are called fully connected layers. The input layer of the fully connected layer has neurons and all of them is connected to every neuron that belong next fully connected layer. First fully connected layer which is shown as “ fc6 ” takes a stack of feature maps then they are flattened to a vector and this vector transformed to the output space. These two fully connected layer has 4096 neurons in this network.
- After the last layer which represented by “ fcB ”, we have added sigmoid layer to have binary predictions.
- Euclidean loss is represented as “ fcB ” in our network and it is computed by using given Formula 3.2. Instead of softmax layer that is used for classification tasks, we have used euclidean loss because we wanted to compute regression result.

$$Euclidean\ Loss = \frac{1}{2N} \sum_{n=1}^N (m(predicted) - m(real))^2 \quad (3.2)$$

- $m(predicted)$: Predicted building density estimation rate
- $m(real)$: Real building density rate
- N : number of samples

Original VGG-16 neural network is shown in the Figure 3.6. The modifications that we have implemented on the network are shown in the Figure 3.7, Figure 3.8 and Figure 3.9.

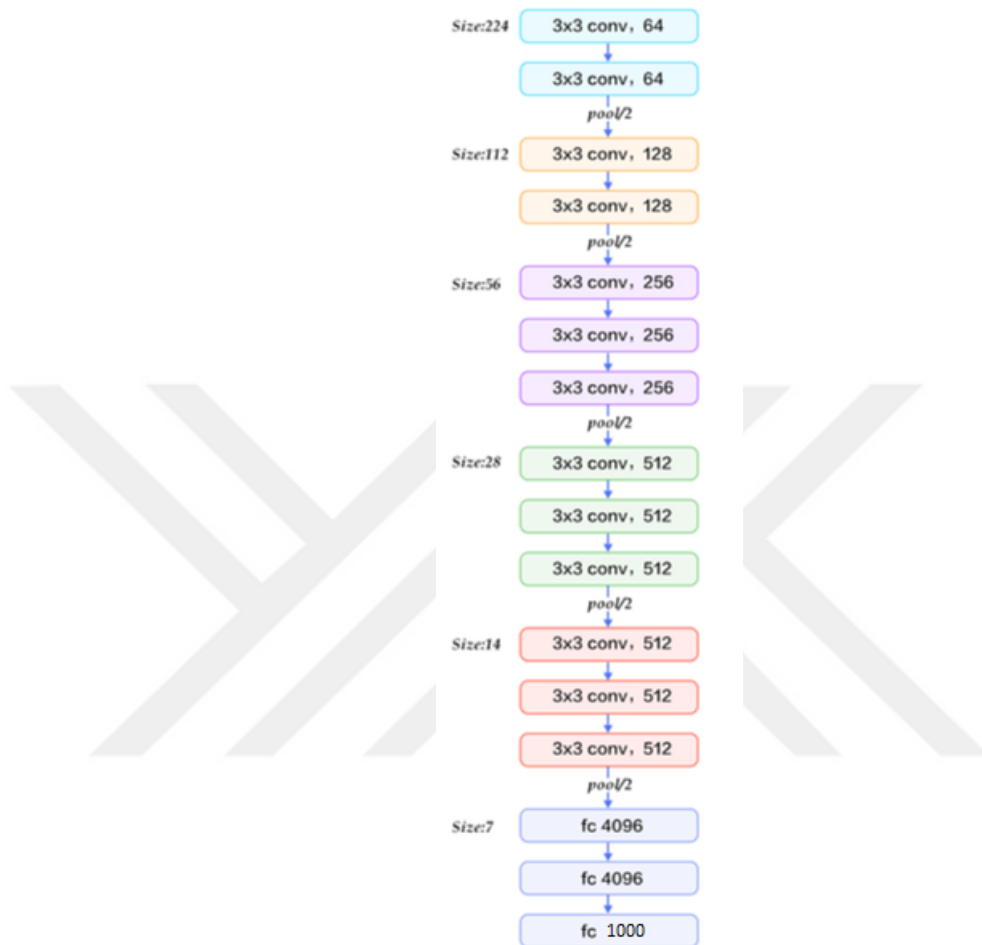


Figure 3.6: Original VGG-16 Network [33]

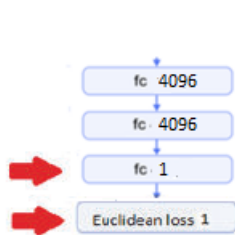


Figure 3.7: Modified VGG-16 Network For Regression

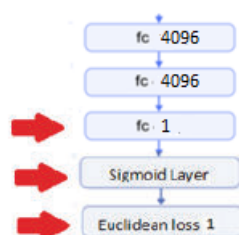


Figure 3.8: Sigmoid Layer Added VGG-16 Network

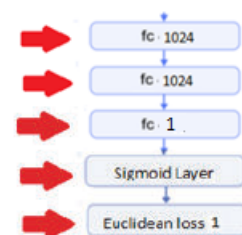


Figure 3.9: Simplified VGG-16 Network

3.3 Fully Convolutional Network (FCN)

Fully convolutional network classifies each pixel on the image and then creates a map of all noticed object areas on the image [34]. It can efficiently learn to make density estimation for per-pixel tasks such as semantic segmentation [35]. Basically, it detects class and location of every pixel in the image. It uses convolutional neural networks so fully connected layers of the network are converted to 1×1 convolutional layers. After that, deconvolutional or transposed convolutional layer is used to recover the activation positions. This is important for gaining some form of localization. Deconvolutional layer is also provide to scale up to the same image size with the input. Figure 3.10 shows fully convolutional neural network flow [34].

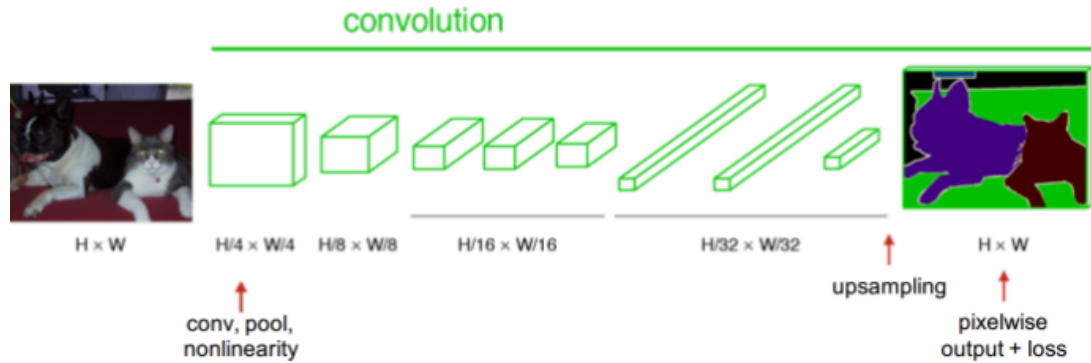


Figure 3.10: Fully Convolutional Neural Network Flow [34]

As in the mention above, FCN classifies each pixel on the image. Therefore, it can be said that it is one of the efficient-ideal network for per-pixel tasks. This is why we wanted to compare our results with its result and modified neural network as in the Table (4.9).

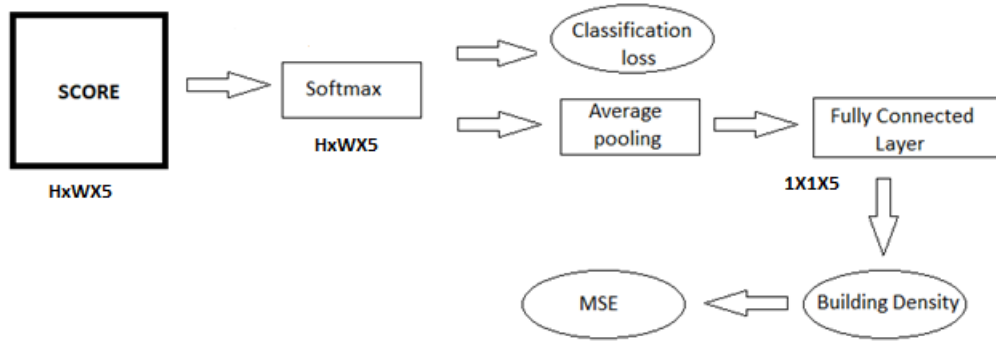


Figure 3.11: The Modification On the FCN

Classification loss is computed in FCN then we have added average pooling layer and fully connected layer to have building density calculation. Result of the fully connected layer gave us MSE and then we have computed total loss:

$$Total\ Loss = Classification\ loss + 10^7 mse \quad (3.3)$$

3.4 Data Augmentation

Data augmentation means that creating new ‘train data’ by applying some methods such as rotating, flipping, translating etc, In the result of previous researches show that data augmentation can act as a regularizer in preventing overfitting in neural networks and improve performance [36]. Therefore, we have provided more data to our train data by using random cropping algorithm. The algorithm crops original train data randomly and it obtains 2375 images from every original train data. We have added 28500 new images which have different sizes. Totally augmented dataset has 34280 small size images at the end of the augmentation. Before they enter the VGG-16 neural network, their sizes are reduced to 224x224 because network fits to 224x224 image size.

3.5 Simplifying The Architecture

Our pre-trained network is trained with large databases containing millions of images. Therefore, we thought that the network architecture could be complex for our data. In this way, we have tried to adapt this complex architecture to our smaller and more specialized domains remote sensing images. The original VGG-16 convolutional neural network has 1000 neurons in the last layer. However, we have reduced it to “ 1 ” because our network makes regression. Therefore, we have also reduced neuron sizes in the fully connected layers which are named as “ fc6 ” and “ fc7 ” in our network. They have 4096 neurons then we modified/reduced them to 1024 neurons.

Chapter 4

Experimental Work

In our work, we applied VGG-16 convolutional neural network and Caffe (Convolutional Architecture for Fast Feature Embedding) as a deep learning framework. VGG-16 is the pre-trained network and it means that network features have been learned from other dataset. This dataset has 14 million images and it is called ImageNet dataset. Pre-trained network is used to train and then test our test data, but we have applied fine-tuning/modification on the network. Fine-tuning is one of the transfer learning method to reduce the error and we have applied it by changing the network layers. As a weight file, we used the file that is obtained from ImageNet dataset.

We have computed the estimations pointwise-locally by dividing our original train and test data into specific numbers. 12 original train data and 4 original test data are divided into small images.

We have divided train data into 5780 small size images then to perform data augmentation, we divided train data into 34280 small size images. They are obtained by cropping randomly in specific sizes. In addition, 4 original test data is also divided into 481 small images. We have focused on regression for building density estimation and originally our data has 5 classes in the image. Table 4.1 shows the classes:

1. Surface	2. Building	3. Vegetation	4. Tree	5. Car
------------	-------------	---------------	---------	--------

Table 4.1: Classes of the Data

Standard deviation shows that how data spreads out according to arithmetic mean of the total data. Therefore, it can be said that the worst result is obtained from standard deviation calculation in these types of experiments. In the end of the experiments, we have computed our root of the mean square error (RMSE) in terms of standard deviation (σ). According to our test dataset, we have evaluated standard deviation (σ) as 0.224. Results are interpreted according to this value.

In addition, we have semantic segmented images of original train and test dataset. They are provided to us from ImageNet Large Scale Visual Recognition Competition (ILSVRC) and we have used them to compute our error rate. Real values represent real building densities of the original test images that are calculated by using semantic segmented images. We computed them pointwise-locally as we have computed predicted values in the test.

4.1 Experiment with Original VGG-16 Network

Original VGG-16 network has 4096 neurons in the fully connected layers. The layers of the original network that used in this experiment is shown in Table 4.2. We initialized to train with the original network that has 13 convolutional layer. Our network is trained by using stochastic gradient descent and small dataset is used to train network. Test result which is obtained by applying test dataset is shown in Table 4.3.

- Small train dataset = 5780 images
- Test dataset = 481 images

data (8L, 3L, 224L, 224L)
sem (8L,)
conv1_1 (8L, 64L, 224L, 224L)
conv1_2 (8L, 64L, 224L, 224L)
pool1 (8L, 64L, 112L, 112L)
conv2_1 (8L, 128L, 112L, 112L)
conv2_2 (8L, 128L, 112L, 112L)
pool2 (8L, 128L, 56L, 56L)
conv3_1 (8L, 256L, 56L, 56L)
conv3_2 (8L, 256L, 56L, 56L)
conv3_3 (8L, 256L, 56L, 56L)
pool3 (8L, 256L, 28L, 28L)
conv4_1 (8L, 512L, 28L, 28L)
conv4_2 (8L, 512L, 28L, 28L)
conv4_3 (8L, 512L, 28L, 28L)
pool4 (8L, 512L, 14L, 14L)
conv5_1 (8L, 512L, 14L, 14L)
conv5_2 (8L, 512L, 14L, 14L)
conv5_3 (8L, 512L, 14L, 14L)
pool5 (8L, 512L, 7L, 7L)
fc6 (8L, 4096L)
fc7 (8L, 4096L)
fcB (8L, 1L)
loss ()

Table 4.2: Layers of VGG-16 Network

Number of Test	Number of Trained Image	Iteration Number	Mean Square Error (MSE) (%)	RMSE /Standard Deviation
E1	5780	20000	0.011	0.49 σ

Table 4.3: Result of Original Network

According to experiment results, iteration number versus root mean square error is shown in the Figure 4.1.

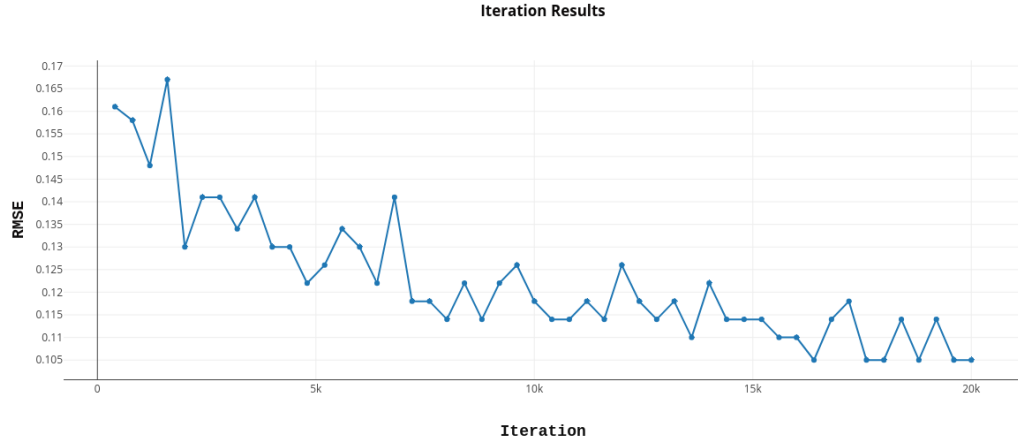


Figure 4.1: Iteration Number vs RMSE

As a result, we obtained ($\sim 0,105$) RMSE from experiment 1 in 20000th iteration. Although its general trend has oscillations but at the end of the experiment, oscillation is very small. Therefore, we stopped at the 20000th iteration because we can say that RMSE values almost converged in that iteration number.

According to every test images, pointwise real values and predicted probabilities are shown in Figure 4.2.

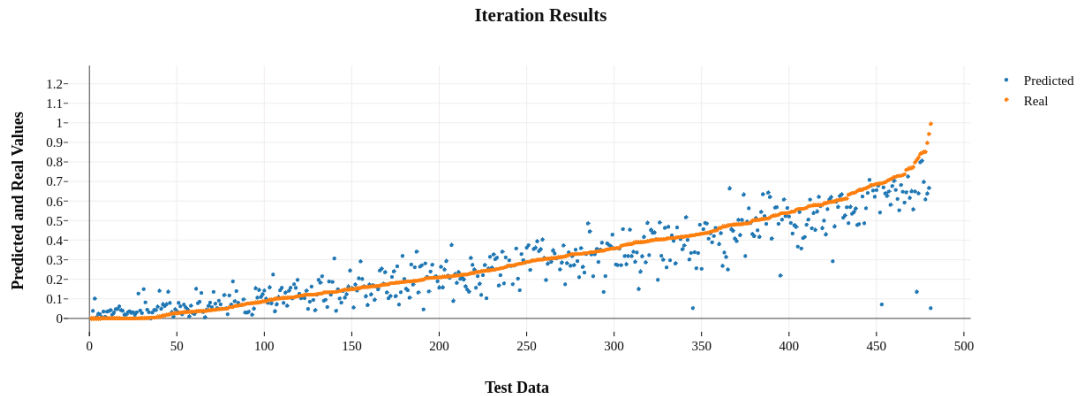


Figure 4.2: Predicted and Real Values vs Test Data Numbers with Original Network

4.2 Experiment with Sigmoid Layer

The last five layers of the network that used in this experiment is shown in Table 4.4. The only difference between previous network is that sigmoid layer is added before the loss layer. We initialized train with the original network that has 13 convolutional layer. Our network is trained by using stochastic gradient descent and train is performed with small train dataset. Test result which is obtained by applying test dataset is shown in Table 4.5.

- Small train dataset = 5780 images
- Test dataset = 481 images

fc6 (8L, 4096L)
fc7 (8L, 4096L)
fcB (8L, 1L)
Sigmoid (8L, 1L)
loss ()

Table 4.4: Last Layers of the Network

Number of Test	Number of Trained Image	Iteration Number	Mean Square Error (MSE) (%)	RMSE /Standard Deviation
E1	5780	20000	0.009	0.42 σ

Table 4.5: Result of Sigmoid Layer + Original Network

According to experiment results, iteration number versus root mean square error is shown in the Figure 4.3.

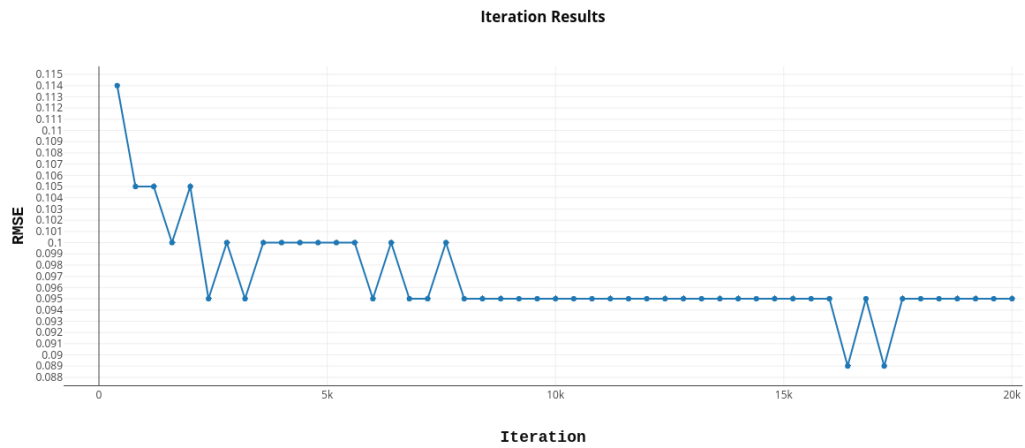


Figure 4.3: Iteration Number vs RMSE

As a result, we obtained ($\sim 0,095$) RMSE from experiment 2 in 20000th iteration. Its general trend has small oscillations in the beginning of the experiment but after the 8000th iteration, it converged. We stopped at the 20000th iteration because we can say that RMSE values almost converged.

According to every test images, pointwise real values and predicted probabilities are shown in Figure 4.4.

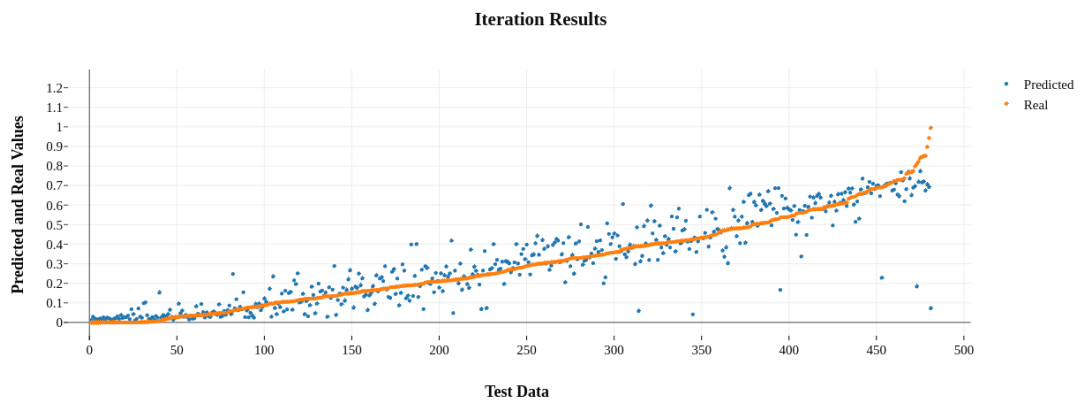


Figure 4.4: Predicted and Real Values vs Test Data Numbers with Sigmoid Layer

4.2.1 Experiment with Sigmoid Layer and Data Augmentation

Sigmoid layer implemented network is used as shown in the Table 4.4. We initialized train with the original network that has 13 convolutional layer. Our network is trained by using stochastic gradient descent and train is performed with augmented dataset. Test result which is obtained by applying test dataset is shown in Table 4.6.

- Augmented train dataset = 34280 images
- Test dataset = 481 images

Number of Test	Number of Trained Image	Iteration Number	Mean Square Error (MSE) (%)	RMSE /Standard Deviation
E1	34280	20000	0.007	0.35 σ

Table 4.6: Result of Sigmoid Layer + Data Augmentation

According to experiment results, iteration number versus root mean square error is shown in the Figure 4.5.

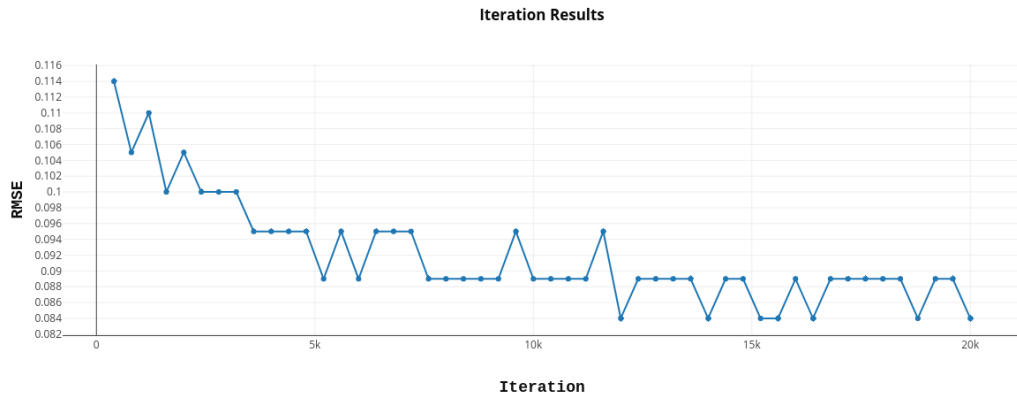


Figure 4.5: Iteration Number vs RMSE

As a result, we obtained ($\sim 0,084$) RMSE from experiment 3 in 20000th iteration. Its general trend has small oscillations in the beginning of the experiment but

after the 12000th iteration, its convergence has very small differences. Therefore, we stopped at the 20000th iteration because we can say that RMSE values almost converged. The difference between experiment 2 and experiment 3 is that data augmentation. Network needs to adapt to the larger dataset, this is why in this experiment oscillations are more than experiment 2.

According to every test images, pointwise real values and predicted probabilities are shown in Figure 4.6.

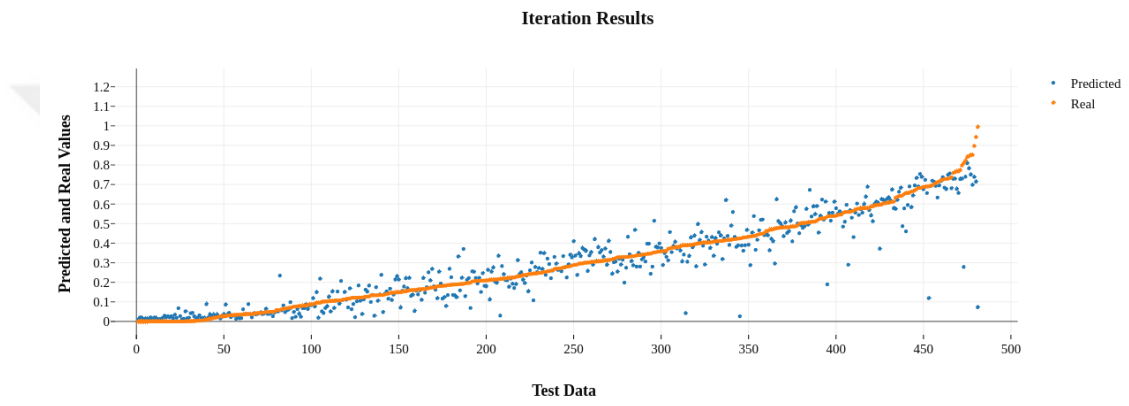


Figure 4.6: Predicted and Real Values vs Test Data Numbers with Sigmoid Layer and Augmented Dataset

4.3 Experiment with Simplified Architecture and Sigmoid Layer

Original VGG-16 network has 4096 neurons in the fully connected layers. However, we have fine-tuned on the network by reducing the neuron number of the fully connected layers. We have reduced them to 1024 neurons. The layers of the simplified network that used in this experiment is shown in Table 4.8. We initialized to train with the simplified network that has 13 convolutional layer. Our network is trained by using stochastic gradient descent and small train dataset is used to train network. Test result which is obtained by applying test dataset is shown in Table 4.8.

- Small train dataset = 5780 images

- Test dataset = 481 images

data (8L, 3L, 224L, 224L)
sem (8L,)
conv1_1 (8L, 64L, 224L, 224L)
conv1_2 (8L, 64L, 224L, 224L)
pool1 (8L, 64L, 112L, 112L)
conv2_1 (8L, 128L, 112L, 112L)
conv2_2 (8L, 128L, 112L, 112L)
pool2 (8L, 128L, 56L, 56L)
conv3_1 (8L, 256L, 56L, 56L)
conv3_2 (8L, 256L, 56L, 56L)
conv3_3 (8L, 256L, 56L, 56L)
pool3 (8L, 256L, 28L, 28L)
conv4_1 (8L, 512L, 28L, 28L)
conv4_2 (8L, 512L, 28L, 28L)
conv4_3 (8L, 512L, 28L, 28L)
pool4 (8L, 512L, 14L, 14L)
conv5_1 (8L, 512L, 14L, 14L)
conv5_2 (8L, 512L, 14L, 14L)
conv5_3 (8L, 512L, 14L, 14L)
pool5 (8L, 512L, 7L, 7L)
fc6 (8L, 1024L)
fc7 (8L, 1024L)
fcBa (8L, 1L)
loss ()

Table 4.7: Layers of Simplified VGG-16 Network

Number of Test	Number of Trained Image	Iteration Number	Mean Square Error (MSE) (%)	RMSE /Standard Deviation
E1	5780	20000	0.008	0.40 σ

Table 4.8: Result of Simplified Network + Sigmoid Layer

According to experiment results, iteration number versus root mean square error is shown in the Figure 4.7.

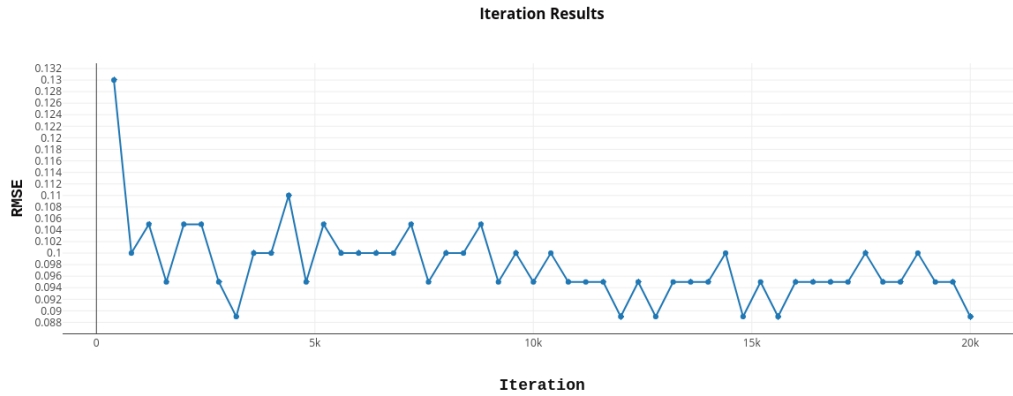


Figure 4.7: Iteration Number vs RMSE

As a result, we obtained ($\sim 0,090$) RMSE from experiment 4 in 20000th iteration. Its general trend has small oscillations in the beginning of the experiment but after the 15000th iteration, its convergence has very small differences. Therefore, we stopped at the 20000th iteration because we can say that RMSE values almost converged.

According to every test images, pointwise real values and predicted probabilities are shown in Figure 4.8.

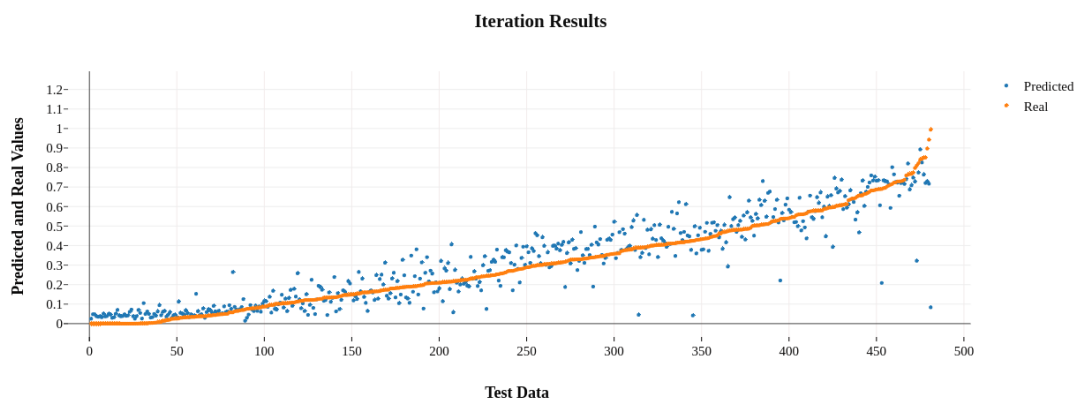


Figure 4.8: Predicted and Real Values vs Test Data Numbers of Simplified Network with Sigmoid Layer

4.4 Comparing the Results

As we mentioned in the section 3.3, the result of FCN is added in the Table 4.9 just for comparing between its result and our results. According to all results of experiments, Table 4.9 is listed and Figure 4.9 is shown as in the below:

Number of Test	RMSE /Standard Deviation
Fully Convolutional Network (E0)	0.35 σ
Orjinal Network (E1)	0.47 σ
Sigmoid Layer (E2)	0.42 σ
Sigmoid Layer + Data Augmentation (E3)	0.35 σ
Simplified Network + Sigmoid Layer (E4)	0.40 σ

Table 4.9: All Experiment Results

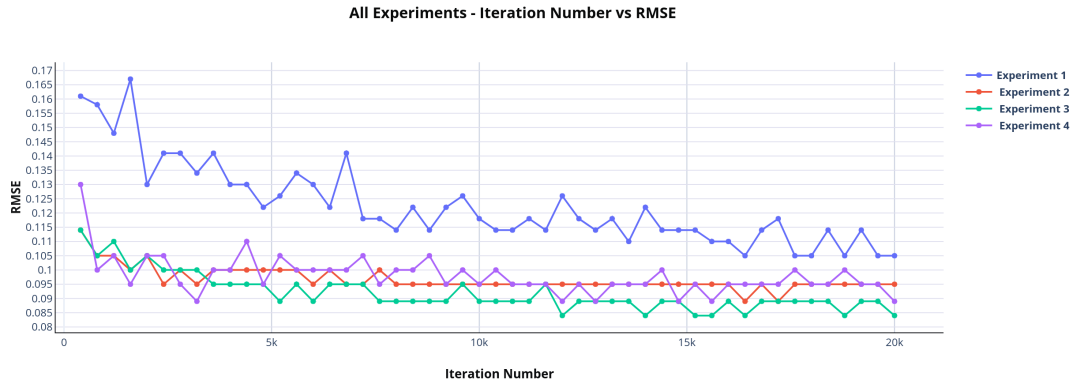


Figure 4.9: All Experiments Iteration Number vs RMSE

As in the Figure 4.9, we obtained best trend at the iteration number vs RMSE by using sigmoid layer added network and augmented data. This is obtained from experiment 3. Sigmoid layer provides true probability estimation and data augmentation for train data provides better learning for network. Therefore, experiment 3 gave us lowest RMSE in the 20000th iteration. In addition, we obtained the same MSE result (0.007) from the FCN network which has a more complex structure than the VGG-16 network.

As we mentioned in the previous section, fully convolutional network (FCN) is one of the ideal network. This network gave the same RMSE value with our lowest RMSE value in the same iteration. However, while the time spent by FCN during the testing phase is 1.66 seconds/image, the time spent by VGG-16 network during the test phase is 0.48 seconds/image. This shows that our network produces estimation results faster than FCN. In addition, VGG-16 network gives the numeric value as output and does not need any other information to estimate density. However, FCN needs ground truth segmentation maps for training. Therefore, it can be said that FCN is harder to train and deploy in real-life applications of building density estimation.

After that, we brought all pointwise predicted results of test data (0005) together to visualize predicted images of the experiments. Then heat-maps are obtained from them to compare results of the experiments. Illustrations of the test data

(0005) are shown in Figure 4.10. The original image and its semantic segmentation map is also shown in the Figure 4.11 and its real heat map is shown in the Figure 4.12.

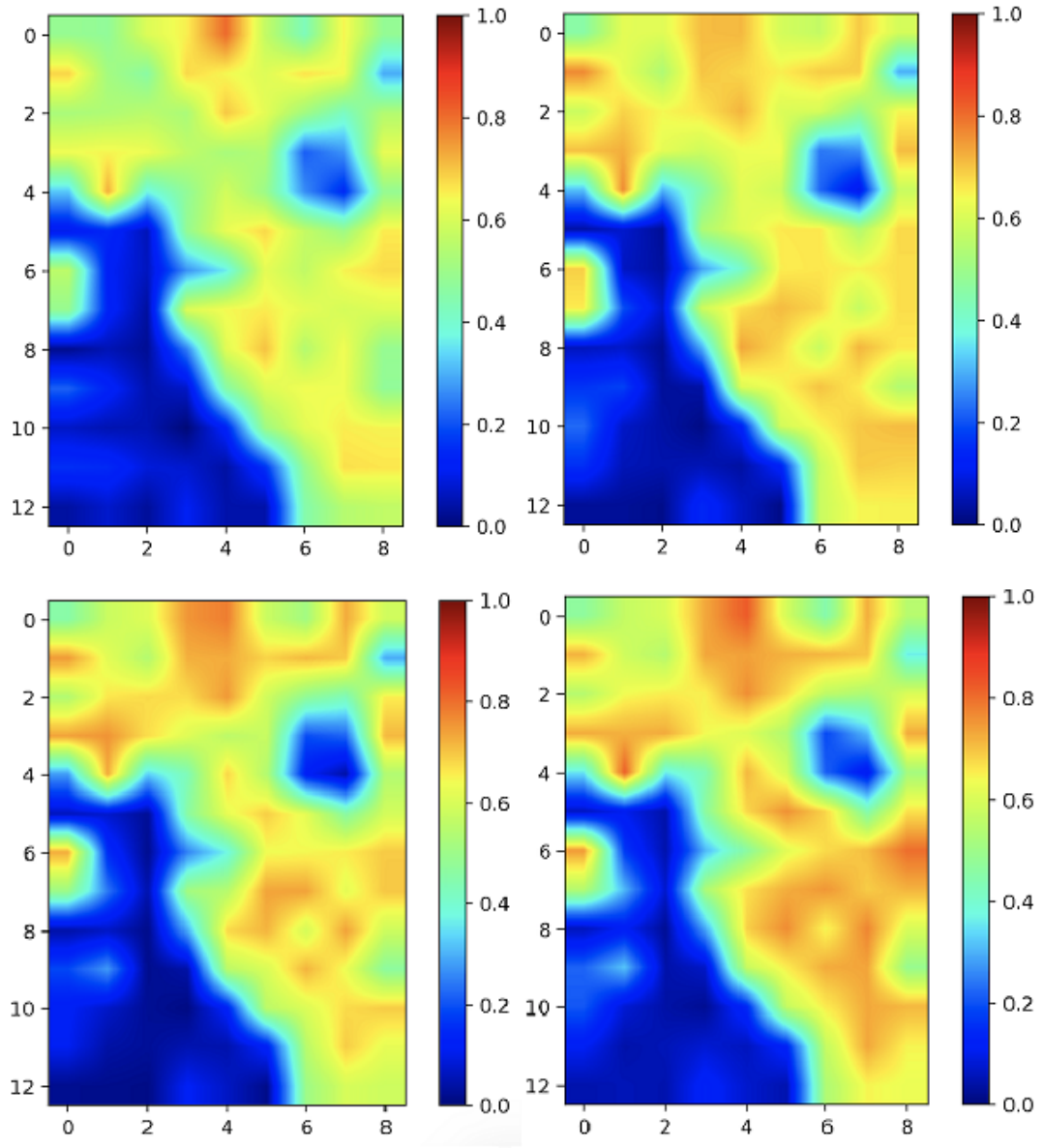


Figure 4.10: All Experiment Heat-Map Results For Test Data (0005) : E1, E2, E3, E4, respectively

Firstly, there is something that we need to consider when comparing the heat maps of experiments and real heat map. Color differences between the heat maps highlight the differences in estimated and true building densities. Therefore, when we are comparing the experiment results, we considered these color differences.

Some differences between heat maps of experiments and real heat map are designated in the Figure 4.12.

As shown in the Figure 4.10, all of the heat maps look like similar but they have small differences that affect their results. In addition, the difference heat maps are obtained by using absolute differences between predicted and real values for test data (0005). They are shown in Figure 4.13. Dark blue color represents small differences, red color represents big differences between predicted and real values in difference heat maps.

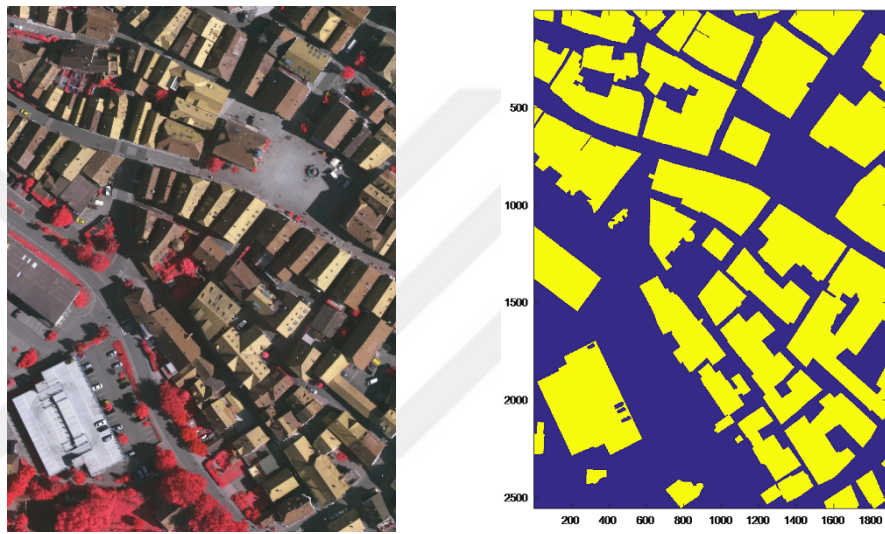


Figure 4.11: Original Test Image and Its Semantic Segmentation Map

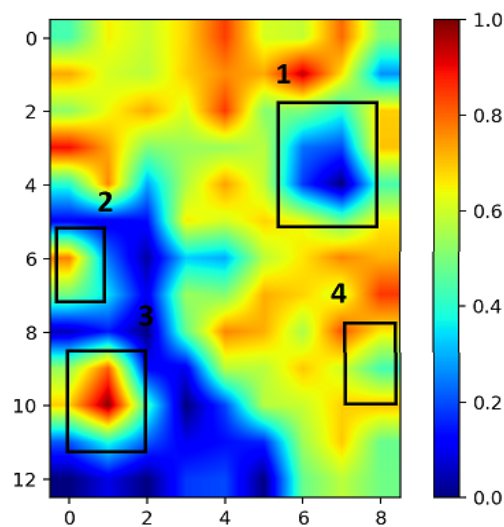


Figure 4.12: Real Heat Map (0005)

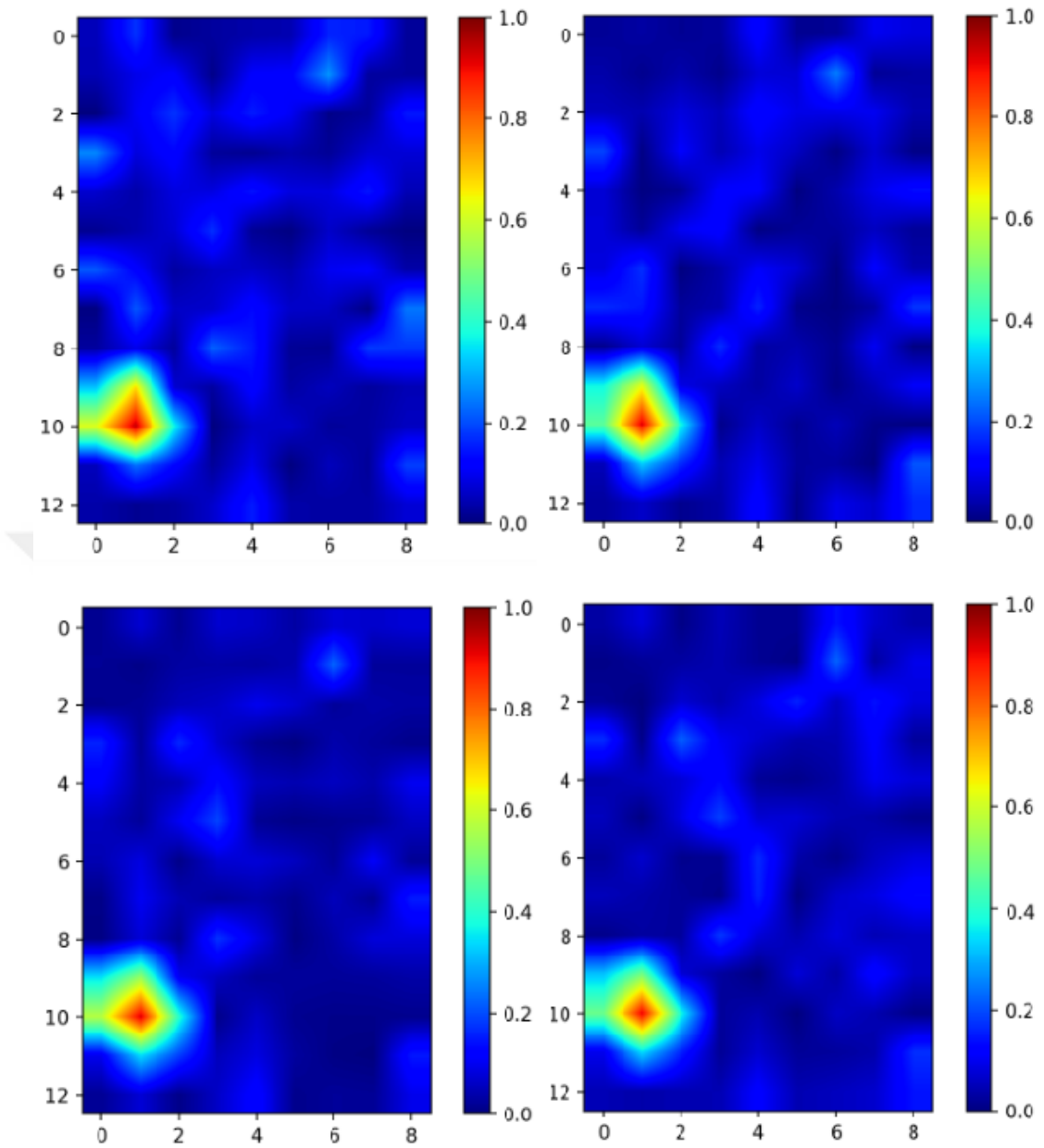


Figure 4.13: Difference Heat-Maps of Test Data (0005) : E1, E2, E3, E4, respectively

In addition, RMSE values of heat maps of E1, E2, E3, E4 (test data 0005) are 0.163, 0.150, 0.144, 0.143, respectively.

The experiment 3 and experiment 4 have closest RMSE values but experiment 4 gives the lowest RMSE value. It means that experiment 4 has predicted values that are nearest to the real values. In general, experiment 3 has darkest blue but

experiment 4 has smallest red area. This can be the reason that experiment 4 has lowest individual RMSE value.

According to difference heat maps, experiment 3 gives the best density estimate for the area indicated by number 1. Experiment 3 and 4 gives the best density estimate for the area indicated by number 2. All experiments missed the building that is located in the left corner. This area is indicated by the number 3 and shown in the Figure 4.12. However, experiment 3 and 4 gives better estimate results for that building when we compare to them with experiment 1 and experiment 2. Lastly, experiment 2,3 and 4 give better estimate for the area indicated by number 4, compared to experiment 1. According to these illustrations and comments, we can say that experiment 3 and 4 give better estimation results for test data (0005) compared to experiment 1 and 2.

Predicted and real values are shown in the Table 4.10, Table 4.11, Table 4.12 and Table 4.13. We chose image patches from the test data (0005) that have either small or large estimation errors as shown in the Table 4.15. The difference values between predicted and real values for these images are shown in the Table 4.14.

Result (E1)	Image Number and Name	Predicted Value	Real Value
Small difference	0005_4_33	0.626	0.702
	0005_12_29	0.670	0.694
Big difference	0005_11_5	0.052	0.994
	0005_12_33	0.664	0.477

Table 4.10: Predicted and Real Values of Experiment 1(E1)

Result (E2)	Image Number and Name	Predicted Value	Real Value
Small difference	0005_4_33	0.710	0.702
	0005_12_29	0.695	0.694
Big difference	0005_11_5	0.072	0.994
	0005_12_33	0.685	0.477

Table 4.11: Predicted and Real Values of Experiment 2(E2)

Result (E3)	Image Number and Name	Predicted Value	Real Value
Small difference	0005_4_33	0.714	0.702
	0005_12_29	0.692	0.694
Big difference	0005_11_5	0.072	0.994
	0005_12_33	0.624	0.477

Table 4.12: Predicted and Real Values of Experiment 3(E3)

Result (E4)	Image Number and Name	Predicted Value	Real Value
Small difference	0005_4_33	0.729	0.702
	0005_12_29	0.734	0.694
Big difference	0005_11_5	0.083	0.994
	0005_12_33	0.648	0.477

Table 4.13: Predicted and Real Values of Experiment 4(E4)

Result	Image Number and Name	E1 Difference (Real-Predicted)	E2 Difference (Real-Predicted)	E3 Difference (Real-Predicted)	E4 Difference (Real-Predicted)
Small difference	0005_4_33	0.076	-0.008	-0.012	-0.027
	0005_12_29	0.024	-0.001	0.002	-0.040
Big difference	0005_11_5	0.942	0.922	0.922	0.911
	0005_12_33	-0.187	-0.208	-0.147	-0.171

Table 4.14: Differences between Predicted and Real Values

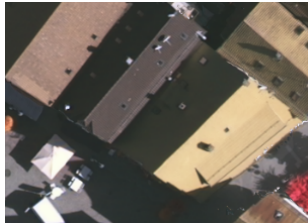

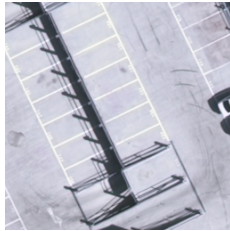

Test Images with Small Estimation Errors		Test Images with Big Estimation Errors	
			
0005_4_33	0005_12_29	0005_11_5	0005_12_33

Table 4.15: Image Patches That Have Either Small or Large Estimation Errors

Images with small estimation errors have noticeable building colors and shapes as shown in the Table 4.15. However, first image (0005_11_5) with big estimation error has not noticeable building colors or shapes even if it has building density of 99%. The second image (0005_12_33) has building color and shape but building color is similar to shadow color of the building. These reasons cause the best and worst estimated results. According to them, experiment 4 and experiment 3 gave us the smallest difference for the images with big estimation errors. Experiment 2 gave us the smallest difference for the images with small estimation errors.

After that, we brought all pointwise predicted results of test data (0030) together to visualize building density heat maps. Illustrations of the test data (0030) are shown in Figure 4.14. The original image and its semantic segmentation map are also shown in the Figure 4.15 and its real heat map is shown in the Figure 4.16.

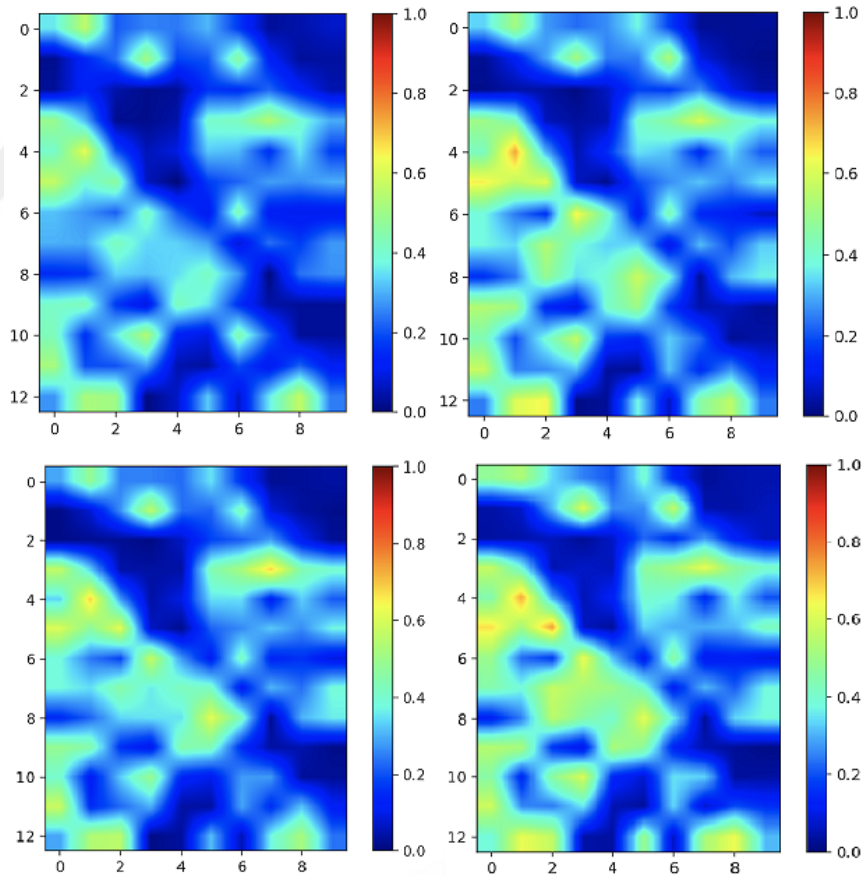


Figure 4.14: All Experiment Heat-Map Results For Test Data (0030): E1, E2, E3, E4, respectively

As we mentioned for the test data (0005) heat maps, we also need to consider color differences between the heat maps highlight the differences in estimated and true building densities for test data (0030). When we are comparing the experiment results, we considered these color differences again. Some differences between heat maps of experiments and real heat map are designated in the Figure 4.16.

As shown in the Figure 4.14, all of the heat maps have small differences that affect their results. In addition, the difference heat maps are also obtained by using absolute differences between predicted and real values for test data (0030). They are shown in Figure 4.17. Dark blue color represents small differences, red color represents big differences between predicted and real values in difference heat maps.

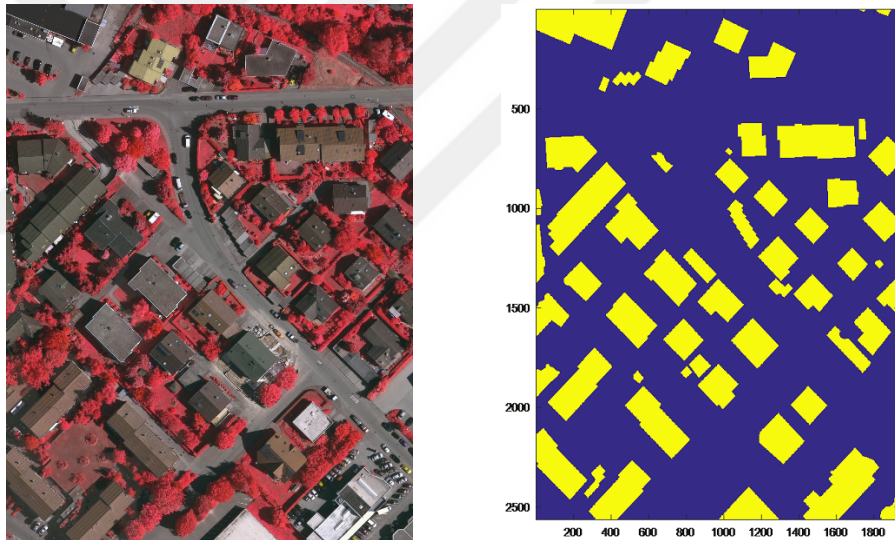


Figure 4.15: Original Test Image and Its Semantic Segmentation Map

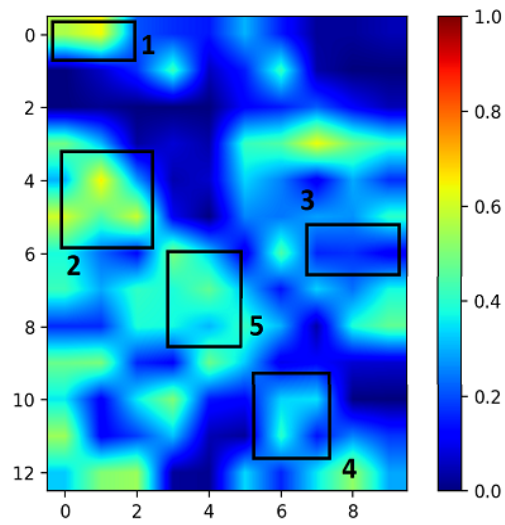


Figure 4.16: Real Heat Map (0030)

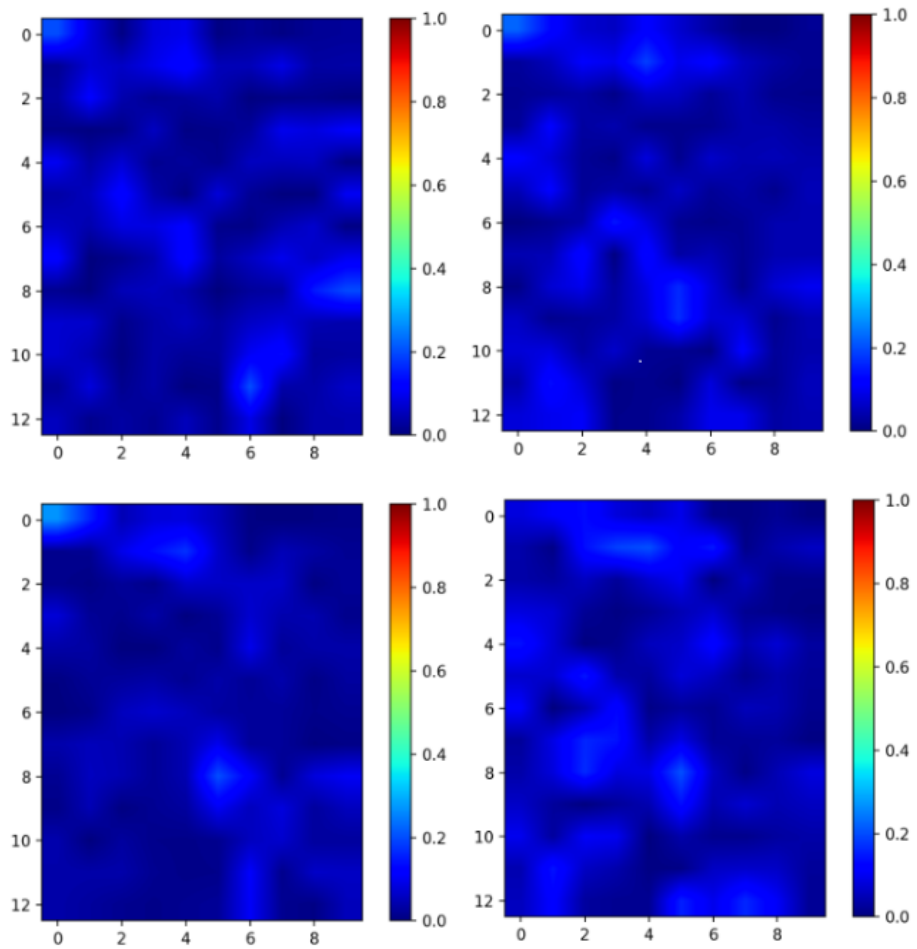


Figure 4.17: Difference Heat-Maps of Test Data (0030): E1, E2, E3, E4, respectively

In addition, RMSE values of heat maps of E1, E2, E3, E4 (test data 0030) are 0.066, 0.069, 0.059, 0.077, respectively.

The experiment 3 gives the lowest RMSE value and we can say that it has more dark blue color in its difference heat map. It means that experiment 3 has predicted values that are nearest to the real values.

According to difference heat maps, experiment 4 gives the nearest probability for the density of the area indicated by number 1. Experiment 1 and 3 give better estimation compared to experiment 2 and 4 for the area indicated by number 2. For the area indicated by number 3, experiment 3 has the best estimation value and experiment 1 has the worst estimation value. Experiment 4 gives the darkest blue that means nearest probability for the area indicated by number 4 compared to other experiments. Lastly, for the area indicated by number 5, experiment 3 has better estimation values than other experiments. According to these illustrations and comments , we can say that experiment 3 and experiment 4 give better estimation results for test data (0030).

Predicted and real values are shown in the Table 4.16, Table 4.17, Table 4.18 and Table 4.19. We chose images from the test data (0030) that have small and big differences between predicted and real values as shown in the Table 4.21. The difference values between predicted and real values for these images are shown in the Table 4.20.

Result (E1)	Image Number and Name	Predicted Value	Real Value
Small difference	0030_6_29	0.285	0.286
	0030_12_17	0.049	0.048
Big difference	0030_1_1	0.357	0.560
	0030_12_25	0.197	0.403

Table 4.16: Predicted and Real Values of Experiment 1(E1)

Result (E2)	Image Number and Name	Predicted Value	Real Value
Small difference	0030_6_29	0.323	0.286
	0030_12.17	0.033	0.048
Big difference	0030_1.1	0.337	0.560
	0030_12.25	0.319	0.403

Table 4.17: Predicted and Real Values of Experiment 2(E2)

Result (E3)	Image Number and Name	Predicted Value	Real Value
Small difference	0030_6_29	0.327	0.286
	0030_12.17	0.038	0.048
Big difference	0030_1.1	0.290	0.560
	0030_12.25	0.290	0.403

Table 4.18: Predicted and Real Values of Experiment 3(E3)

Result (E4)	Image Number and Name	Predicted Value	Real Value
Small difference	0030_6_29	0.301	0.286
	0030_12.17	0.053	0.048
Big difference	0030_1.1	0.478	0.560
	0030_12.25	0.341	0.403

Table 4.19: Predicted and Real Values of Experiment 4(E4)

Result	Image Number and Name	E1 Difference (Real-Predicted)	E2 Difference (Real-Predicted)	E3 Difference (Real-Predicted)	E4 Difference (Real-Predicted)
Small difference	0030_6_29	0.001	-0.037	-0.041	-0.015
	0030_12.17	-0.001	0.015	0.010	-0.005
Big difference	0030_1.1	0.203	0.223	0.270	0.082
	0030_12.25	0.206	0.084	0.113	0.062

Table 4.20: Differences between Predicted and Real Values


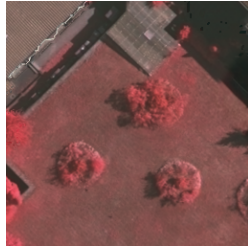
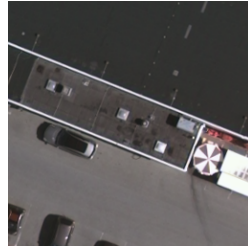

Test Images with Small Estimation Errors		Test Images with Big Estimation Errors	
			
0030_6_29	0030_12_17	0030_1_1	0030_12_25

Table 4.21: Image Patches That Have Either Small or Large Estimation Errors

Images with small estimation errors have noticeable building colors and shapes as shown in the Table 4.21. However, first image (0030_1_1) with big estimation error has not noticeable building colors or shapes even if it has building density of 40%. The second image (0030_12_25) has building color and shape but building color is similar to shadow color of the building and shadow color of the roof. These reasons cause the best and worst estimated results.

According to them, experiment 4 gave us the smallest difference for the images with big estimation errors. Experiment 1 gave us the smallest difference for the images with small estimation errors.

As a result, our RMSE results have small differences between them. Therefore, considering all differences in heat maps or numerical values, it can be said that the best experiment does not always have the best value. Generally, experiment 3 has best trend and result so we can say that data augmentation with sigmoid layer in the network provides us best result.

Chapter 5

Conclusion

In this project, pointwise estimation of building density on the remote sensing optical imageries by applying deep learning method is performed. The main objective of the thesis is to reduce mean square error of the estimated density by applying architectural modifications on the deep learning network and using augmented training data. We improved accuracy of the estimation by applying sigmoid layer addition in the network, simplifying the network for small sized training dataset and using data augmentation.

Data augmentation provided considerable improvement in density estimation performance. In addition, we showed that sigmoid layer is effective network layer for regression task and simplifying network has marginal contribution to regression results.

The main contribution of this thesis is to show that some network modifications and data augmentation provide better learning in deep neural networks. Better learning brings low RMSE and data augmentation provides it. More data for train dataset provides more information for neural network and more information increased accuracy of the estimation. Regression is performed by using VGG-16 network and adding sigmoid layer has also improved accuracy. Sigmoid layer as a final activation function avoids the estimated probability being too large or too small and this provides true probability estimation and stability of network. Simplifying network is the other modification method to reduce RMSE.

Simplified network has also sigmoid layer in the network and provides better result than original network and only sigmoid layer added network in small sized train dataset. Although only sigmoid layer added network and simplified network reduced RMSE, the lowest RMSE is obtained from sigmoid layer added network by using data augmentation.

As a conclusion, we managed to reduce RMSE in the pointwise density estimation by modifying the network and augmenting the training dataset. We compared the performance of our modified network with that of FCN which is adapted to density estimation. Applying FCN is computationally expensive compared with VGG-16 network and it needs segmentation maps to train. We have proven the network (VGG-16) that we have used can have same RMSE by applying network modification with less information and less time compared to FCN.

As a future work, the research in this thesis can be improved, as follows:

- Other gradient descent methods such as Adam, AdaDelta could be tested.
- Bigger patch sizes could be tried to provide more contextual information from surrounding regions in the image
- Digital elevation maps could be used in addition to the multispectral information.

References

- [1] J. Zhang, T.; Huang, X.; Wen, D.; Li,, “Urban building density estimation from high-resolution imagery using multiple features and support vector regression,” *Ieee Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 7, pp. 3265–3280, 2017.
- [2] Y. Zhou, C. Lin, S. Wang, W. Liu, and Y. Tian, “Estimation of building density with the integrated use of gf-1 pms and radarsat-2 data,” *Remote Sensing*, vol. 8, no. 11, p. 969, 2016.
- [3] B. Yu, H. Liu, J. Wu, Y. Hu, and L. Zhang, “Automated derivation of urban building density information using airborne LiDAR data and object-based method,” *Landscape and Urban Planning*, vol. 98, no. 3-4, pp. 210–219, 2010.
- [4] Q. Wu, R. Chen, H. Sun, and Y. Cao, “Urban building density detection using high resolution SAR imagery,” in *2011 Joint Urban Remote Sensing Event, JURSE 2011 - Proceedings*, 2011, pp. 45–48.
- [5] X. Z. Pan, Q. G. Zhao, J. Chen, Y. Liang, and B. Sun, “Analyzing the variation of building density using high spatial resolution satellite images: The example of Shanghai City,” *Sensors*, vol. 8, no. 4, pp. 2541–2550, 2008.
- [6] S. C. Liew, “Principles of Remote Sensing - Centre for Remote Imaging, Sensing and Processing, CRISP,” *Interpreting Optical Remote Sensing Images*, pp. 2–6, 2001.
- [7] L. Zhang, L. Zhang, and B. Du, “Deep learning for remote sensing data,” *IEEE Geoscience and Remote Sensing Magazine*, 2016.

- [8] Q. A. Action, *Issues in Analysis, Measurement, Monitoring, Imaging, and Remote Sensing Technology*. Scholarly Editions, 2012.
- [9] “What Is Deep Learning AI? A Simple Guide With 8 Practical Examples,” Forbes. [Online]. Available: <https://www.forbes.com/sites/bernardmarr/2018/10/01/what-is-deep-learning-ai-a-simple-guide-with-8-practical-examples/#198bc2e68d4b>
- [10] X. X. Zhu, D. Tuia, L. Mou, G.-S. Xia, L. Zhang, F. Xu, and F. Fraundorfer, “Deep learning in remote sensing: A comprehensive review and list of resources,” *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 8–36, 2017.
- [11] “CS231n Convolutional Neural Networks for Visual Recognition,” GitHub. [Online]. Available: <http://cs231n.github.io/neural-networks-1/>
- [12] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [13] “Supervised machine learning: Classification,” Towardsdatascience. [Online]. Available: <https://towardsdatascience.com/supervised-machine-learning-classification-5e685fe18a6d>
- [14] “Deep learning for remote sensing,” Intel. [Online]. Available: <https://www.intel.ai/deep-learning-for-remote-sensing/#gs.cwc2cbcQ>
- [15] C. Robinson, F. Hohman, and B. Dilkina, “A deep learning approach for population estimation from satellite imagery,” in *Proceedings of the 1st ACM SIGSPATIAL Workshop on Geospatial Humanities*. ACM, 2017, pp. 47–54.
- [16] X. Yang, C. Zhang, S. Gao, F. Yu, and D. Song, “A Model of extracting building from high resolution remote sensing image based on Bayesian Convolutional Neural Networks,” , 2018.
- [17] G. A. Churchill and R. W. Doerge, “Empirical threshold values for quantitative trait mapping.” *Genetics*, vol. 138, no. 3, pp. 963–971, 1994.

- [18] “CS231n Convolutional Neural Networks for Visual Recognition,” GitHub. [Online]. Available: <http://cs231n.github.io/convolutional-networks/>
- [19] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, M. Hasan, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, “The history began from alexnet: A comprehensive survey on deep learning approaches,” 2018.
- [20] “The ultimate guide to convolutional neural network (cnn),” Superdatascience. [Online]. Available: <https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn>
- [21] “Under the hood of neural networks. part 1: Fully connected.” Towardatascience. [Online]. Available: <https://towardsdatascience.com/under-the-hood-of-neural-networks-part-1-fully-connected-5223b7f78528>
- [22] “Derin öğrenme uygulamalarında en sık kullanılan hiper-parametreler,” Medium. [Online]. Available: <https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4>
- [23] “What are hyperparameters ? and how to tune the hyperparameters in a deep neural network?” Towardsdatascience. [Online]. Available: <https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a>
- [24] “CS229 Lecture notes,” Stanford. [Online]. Available: <http://cs229.stanford.edu/notes/cs229-notes1.pdf>
- [25] “Caffe tutorial,” Berkeley. [Online]. Available: <http://caffe.berkeleyvision.org/tutorial/>
- [26] “Stochastic gradient descent - mini-batch and more,” Adventuresinmachinelearning. [Online]. Available: <https://adventuresinmachinelearning.com/stochastic-gradient-descent/>

- [27] “Understanding of convolutional neural network (cnn) deep learning,” Medium. [Online]. Available: <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>
- [28] L. Wang, K. Scott, D. C. R. Sensing, and U. 2017, “Sea Ice Concentration Estimation during Freeze-Up from SAR Imagery Using a Convolutional Neural Network,” *Remote Sensing*, vol. 9, no. 5, p. 408, 2017.
- [29] “Deep learning a-z: Convolutional neural networks (cnn) - step 1(b):,” Slideshare. [Online]. Available: <https://www.superdatascience.com/convolutional-neural-networks-cnn-step-1b-relu-layer-presentation/>
- [30] “Relu : Not a differentiable function: Why used in gradient based optimization?” Medium. [Online]. Available: <https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec>
- [31] “Learning less to learn better,” Medium. [Online]. Available: <https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5>
- [32] “2d semantic labeling - vaihingen data,” ISPRS. [Online]. Available: <http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-vaihingen.html>
- [33] “What is the VGG neural network?” Quora. [Online]. Available: <https://www.quora.com/What-is-the-VGG-neural-network>
- [34] “Review: FCN Fully Convolutional Network (Semantic Segmentation),” Towardsdatascience. [Online]. Available: <https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1>
- [35] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

- [36] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, “Understanding Data Augmentation for Classification: When to Warp?” in *2016 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2016*.

