KERNEL CLASSIFICATION OF MOTION BLURRED IMAGES USING ARTIFICIAL NEURAL NETWORKS

Master of Science Thesis

Muhammet Ali ŞİRVAN

Eskişehir 2019

KERNEL CLASSIFICATION OF MOTION BLURRED IMAGES USING ARTIFICIAL NEURAL NETWORKS

Muhammet Ali ŞİRVAN

MASTER OF SCIENCE THESIS

Electrical and Electronics Engineering Program
Supervisor: Prof. Dr. Ömer Nezih GEREK

Eskişehir
Anadolu University
Graduate School of Sciences
June 2019

FINAL APPROVAL FOR THESIS

This thesis titled "Kernel Classification of Motion Blurred Images Using Artificial Neural Networks" has been prepared and submitted by Muhammet Ali ŞİRVAN, in partial fulfillment of the requirements in "Anadolu University Directive on Graduate Education and Examination" for the Degree of Master of Sciences in Electrical and Electronics Engineering Department has been examined and approved on 24/06/2019.

Committee Members	**Title, Name Surname**	**Signature**
Member (Supervisor)	: Prof. Dr. Ömer Nezih GEREK	.....
Member	: Assoc. Prof. Dr. Kemal ÖZKAN	.....
Member	: Assoc. Prof. Dr. Tansu FİLİK	.....

Director of Graduate School of Science

ABSTRACT

KERNEL CLASSIFICATION OF MOTION BLURRED IMAGES USING ARTIFICIAL NEURAL NETWORKS

Muhammet Ali ŞİRVAN

Department of Electrical And Electronics Engineering

Anadolu University, Graduate School of Sciences, June 2019

Supervisor: Prof. Dr. Ömer Nezih GEREK

When capturing a moving object with the camera, motion blur occurs in the picture depending on the speed and direction of movement. The motion blur can be classified in terms of filter matrices and the picture can be restored using it. The purpose of this project is finding the filter matrices that causes to the motion blur using artificial neural networks. Once the filter matrix or point spread function (PSF) found, the motion-blurred images can be sharpened using well-known algorithms such as Wiener, Lucy Richardson Deconvolution. A few datasets that have lots of training images are blurred with the parameters (blur angle and length) of specific motion blur kernels and they are trained using VGG, ResNet and a network model that is shared in an article. Then, the prediction of the filter matrices is evaluated using validation datasets for every trained networks. The prediction accuracies for the trained networks are compared to show which network type is best for the purpose of this thesis. Some types of artificial neural networks and their structures are investigated for their suitability in terms of finding the accurate PSF that causes motion blur. In addition, some modifications are applied to the network to increase the prediction accuracy. This improved the estimation of the correct filter matrices.

Keywords: Artificial Neural Network, Blur Kernel Detection, Motion Blur.

ÖZET

RESİMLERDEKİ HAREKET BULANIKLIĞINA NEDEN OLAN FİLTRE MATRİSLERİNİN YAPAY SİNİR AĞLARI KULLANILARAK SINIFLANDIRILMASI

Muhammet Ali ŞİRVAN

Elektronik Anabilim Dalı

Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Haziran 2019

Danışman: Prof. Dr. Ömer Nezih GEREK

Hareketli bir objenin kamera ile çekilmesi sırasında, hıza ve hareket yönüne bağlı olarak resimde hareket bulanıklığı oluşmaktadır. Hareket bulanıklığı, filtre matrisleriyle ifade edilebilinir ve bu matrislerle görüntü restorasyonu sağlanabilir. Bu projenin amacı, yapay sinir ağlarının eğitilip, resimdeki bulanıklığa neden olan filtre matrislerinin tahmin edilmesidir. Filtre matrisi bulunduktan sonra, Wiener ve Lucy-Richardson Dekonvolüsyonu gibi bazı algoritmalar kullanılarak, resimler kolayca daha keskin ve detaylı hale getirilebilinir. Bu projede, VGG, ResNet ve bir makalede paylaşılmış ağ yapısı kullanılarak, belirli açı ve uzunluk parametreleriye bulanıklaştırılan resimler eğitilmiştir. Eğitilen yapay sinir ağlarıyla, hareket bulanıklığı içeren resimlerin filtre matrislerinin tahmini gerçekleştirilip, tezin amacı doğrultusunda ağ tipleri kıyaslanmıştır. Eğitilen bazı yapay sinir ağ tiplerinin ve yapılarının, hareket bulanıklığına sebep olan filtre matrisini bulma bakımından uygunluğu araştırılmıştır. Ağ katmanlarındaki bazı düzenlemeler ile, filtre matrislerini doğru tahmin etme oranının nasıl arttırılacağı gösterilmiştir.

Anahtar Sözcükler: Yapay Sinir Ağları, Filtre Matrislerinin Bulunması, Hareket Bulanıklığı,

ACKNOWLEDGEMENT

I would like to express my deepest appreciation and special thanks to my advisor Prof. Dr. Ömer Nezih GEREK for his great constructive guidance, supports at motivation and helps to increase my excitement to this work. His immense knowledge, experience, high skills and especially the excellence in Signal Processing area is helped me to learn very well and complete this thesis. I would also like to express my sincere thanks to my committee members Assoc. Prof. Dr. Kemal ÖZKAN and Assoc. Prof. Dr. Tansu FİLİK who manifested their distinguished skills in their own fields as seen in their way of correction, suggestions and ideas shared.

A thesis cannot be successfully prepared without high effort and work. In other words, to achieve a goal and do a successful work, material and moral support is needed. Therefore, I would like to thank to my friends, my sister Fatma ŞİRVAN, my family who helped me with their support, patience and confidence.

Without technology and knowledge that is obtained from the science, it was not possible to complete this project. Therefore, I would like to gratitude to people who contribute to science with sharing their studies. Also special thanks goes to Google Team for supporting the scholars, researchers and students for offering free and high performance GPU hardware&software platform Colaboratory. Without using this platform, lots of time would be required to complete this work.

..../..../20....

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with "scientific plagiarism detection program" used by Anadolu University, and that "it does not have any plagiarism" whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

.....

(Signature)

Muhammet Ali ŞİRVAN

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

Page

LIST OF ABBREVIATIONS AND SYMBOLS

Ɵ	: Teta
CPU	: Central Processor Unit
dB	: Decibel
GAN	: Generative Adversarial Network
GPU	: Graphics Processor Unit
MRF	: Markov Random Field
PSF	: Point Spread Function
PSNR	: Peak Signal-to-Noise Ratio
RELU	: Rectified Linear Units
ResNet50	: 50 Layered Residual Network Model
ROI	: Region of Interest
SGD	: Stochastic Gradient Descent
TPU	: Tensor Processing Unit
VGG	: Visual Geometry Group
VGG16	: 16 Layered Neural Network Model
VGG19	: 19 Layered Neural Network Model

1. INTRODUCTION

1.1. Overview

Increasing usage of the mobile phones and tablets are provided people to make the life easier. Using Internet connection, wireless almost the control and accessibility of everything is in our hands. Nowadays, social media is used by people to have a good time; capturing memories using camera or recording videos and sharing them is a trend. On the other hand, somebody may want to scan a barcode, capture a picture to zoom in and see the image in detailed. These types of usages require having a high quality camera. Besides having a good camera, the need of image processing maybe required. Images may have distortions, noise or defocus problem during capturing. The current technology supports the hardware and software solutions for image processing in phones or cameras. The software solutions can be handled with algorithmic operations or using artificial neural networks.

While the camera and the interested object have a relatively motion difference during the capturing, motion blur problem shows itself. Deblurring operation is used to recover the original images to solve this problem. There are lots of ways to recover the motion blurred images. For example, in frequency domain, Wiener filter and inverse filter can be used. In spatial domain some iterative algorithms such as Richardson-Lucy [1] and Landweber Van Cittert [2] can be used. There are two shapes of motion blur; uniform motion blur and non-uniform motion blur. The non-uniform blurred images can be divided into patches and deblurred as uniform motion deblurring method. Deblurring is also possible using Artificial Intelligence [3]. For decades, many works are done based on artificial intelligence and most of them can solve the image processing problems such as [4] and [5]. Using artificial neural networks, it is possible to find the blur kernel parameters from motion blurred images. After finding parameters, deblurring will be easier with restoration algorithms.

1.2. Outline of Thesis

In this project, the main purpose is obtaining kernel parameters of uniform motion blur using artificial neural networks. The outline of the thesis, the objective and literature review is discussed in Chapter 1. There is various type of articles related to this thesis. These are researched and mentioned.

In the beginning of Chapter 2, motion blur problem and its causes are introduced. Also, the mathematical equation, definition of the blur kernel and its parameters are given. Furthermore, in Chapter 2.1 and Chapter 2.2, uniform motion blur and non-uniform motion blur is outlined and demonstration is made with the examples of images.

In Chapter 3, artificial neural networks are outlined. In Chapter 3.1, the definition of the layers and their tasks are described. In order to minimize the error during training of neural networks, loss functions are outlined in Chapter 3.1. In Chapter 3.3, various types of optimization algorithms mentioned and the mostly used one is explained. Transfer learning which speeds up the training process and reduces the time required to train is explained in Chapter 3.4. A few popular image datasets and their properties, usages and types of classes are introduced. In Chapter 3.6, a few of the most popular network models VGG16, VGG19 [6] and ResNet50 [7] are introduced in detail. Chapter 3.7 outlines the training frameworks and importance of the training speed that is directly related with GPU performance.

In Chapter 4, the networks which will be trained, the classes of the selected blur kernels and their parameters are outlined. Also, producing blur kernel using MATLAB [8] and some examples are illustrated. In Chapter 4.1, the datasets which will be used for training and testing purpose are shown as diagram. Furthermore, methods to produce the motion blur datasets and labels are described. Chapter 4.2 explains the importance of dropout layers to prevent overfit. In order to obtain the high performance from the trained model, the fine tuning process which is used for this task is described. In Chapter 4.4, a few training tricks to overcome overfit problem, a few trained network and their different behavior are presented. Also, for performance issues, applying transfer learning and fine tuning is proposed.

In Chapter 5, the results of training accuracies and optimizer parameters are presented for each network models that are VGG16, ResNet50, Jian Sun's model [9] and VGG19. The problems and bugs in the training codes are outlined and the found solutions for them are proposed in Chapter 5.5. In order to show the performance of the best network model, the classification kernel count of motion blur is increased and results are shared in Chapter 5.6. In Chapter 5.7, simple restoration task is applied to a blurred image. This is done by using the prediction values that is obtained from the previously trained neural network models.

In Chapter 6, future works are discussed and some advanced methods are proposed for the restoration of the blurred images.

In Chapter 7, the obtained results are discussed to demonstrate which type of neural network model is better. In addition, some modifications for forward type network models are suggested to increase the performance of neural networks for the purpose of detecting motion blur kernels.

1.3. Related Works

There are many works from the literature about training the noisy images and with their ground truth kernels. Authors of [10] are developed an approach to find the angle with using Gabor filter. The length is found by a neural network that depends on radial basis function and sum of Fourier coefficients as learnable feature. In [11], Radon transform used for cepstral analysis to find parameters of motion blur. Using Discrete Cosine Transform, [12] has showed it is possible to obtain the parameters of uniformly motion blurred images. Another work presents blur kernel estimation blindly by analyzing the image statistics [13]. Using Hough Transform, a literature work [14] which obtains blur kernel by the periodical lines from the frequency spectrum of motion blurred image. After lines are detected, they found the blur angle using orientation of Hough lines. The blur length is found by decreasing two dimensional spectrum into one dimensional within the direction of Hough lines.

In an article [9], the authors who did similar work compared with the topic of this thesis proposed a method to obtain blur kernel using artificial neural networks. Uniformly motion blurred images are trained with their custom network. The designed network that has six layers is shown in Figure 1.1. There are two convolutions, two max pooling, a fully connected and softmax layer. The layers have Relu as activation function type. The input images are 30x30 RGB images. The work mentions that they have chosen 1000 images from PASCAL VOC 2010 database (http-1), patched to 30x30 and applied specific kernels to them. All the motion blurred data labelled with their processed kernel type and the final dataset is constructed. The final dataset consists of 1.4 million images. In training process, stochastic gradient descent has used as optimization algorithm and the batch size has selected 64. Caffe (http-2) deep learning framework which is developed by Berkeley AI Research community,

has been used. The author did not mention about the accuracy of network but in this thesis this network is trained to compare the results with another-well known networks and proved that the network type is not accurate enough. Also, the number of blur kernel classes are less. Their proposed model is shown in Figure 1.1 [9].

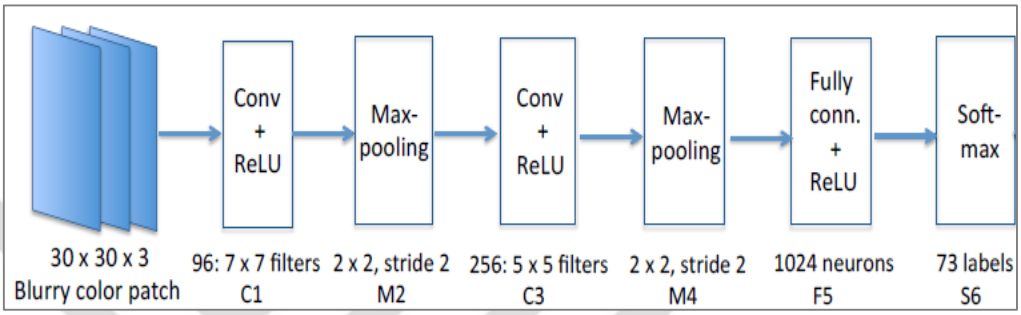

Figure 1.1. *Jian Sun's network model*

The authors in [15] is constructed a network to solve the blind image deblurring problem. The recovery of the image is inverse problem, so blur kernel and desired images are not known. The blurring method assumes some sparsity inducing image prior for clean input image, and follows iteratively, multi-scale estimation scheme, alternating between blur and latent image estimation. Their idea is, unrolling this reconstruction procedure and solving it as a nonlinear regression problem. The parameters are expected to be learned from the training process with the known dataset and its correspondences. Their network performs these in three steps;

- Feature extraction, which finds useful feature and patches in order to estimate kernel.
- Kernel estimation takes the previously extracted features and estimates the kernel.
- Image estimation tries to extract sharp image using estimated kernel.

Their results showed the success is competitive with the methods from another literature works, but it fails when the kernel size is 25x25 and below. Therefore, the non-uniform images cannot be divided into mini patches smaller than 25x25. This prevents to obtain PSF's in a well manner. In the next sections, the reason of this failure will be investigated and will be resulted as the requirement of deep network for the forward network types.

2. BLUR PROBLEM

The speed of moving objects in films or videos can be high and the relation between capturing device and object may not be good enough to obtain clean and sharp outputs. Also accidently shaking the camera leads to this problem. Because of these, the captured images or image frames seems blurry. The edges, color transitions and gradients becomes smoothed. There are three known type of blur effects; average blur, Gaussian blur and motion blur [16]. In this thesis, the uniform motion blur problem is focused. The noise term is neglected. The blur in the captured image is related with the motion parameters of objects and camera. The motion blur parameters can be expressed as an angle and a motion blur length. The angle value estimated from the direction of the moving object and camera relatively. Motion blur length depends to the relative speed of object and camera. Also capturing capability of camera may add some noise to the image, but it is neglected in this thesis. The motion blur may not be uniform. If the PSF is globally same for the whole image, it can be uniform. Otherwise, it is called as non-uniform motion blur. The uniform motion blur can be expressed mathematically as follows;

$$B(x,y) = I(x,y) * h(x,y) + n(x,y) \tag{2.1}$$

The B(x,y) is motion blurred 2D image, I(x,y) is the original 2D sharp image, h(x,y) is the point spread function(PSF) or can be said blur kernel and n(x,y) is the noise. There is another demonstration of this equation and the only difference is noise added to the I(x,y) before convolution operation. In this thesis, noise is neglected.

2.1. Uniform Motion Blur

Uniformly motion blurred images have same PSF in whole image. The motion blur kernel or PSF can be expressed with a length and an angle value because it is possible to construct the PSF with these parameters. An example for uniformly motion blurred image is shown in Figure 2.1.

Figure 2.1. *a) Original image, b) Uniformly motion blurred image with parameters Θ(Angle)=11º, L(Length)=31, no additive noise*

The motion blurred image that is shown in Figure 2.1 is created with MATLAB.

2.2. Non-Uniform Motion Blur

The relative speed between the object and camera may not be same always; it can vary in small amount of time. On the other hand, if the focused scene is not only the object but there is background behind the moving object, non-uniform motion blur problem shows itself. An example is shown in Figure 2.2 (http-3).

In Figure 2.2, the bus causes motion blur because of its speed, but the background or environment is static so it is clear. Also, the motion blur length shows itself in front of the bus. The whole image does not have same PSF. In Figure 2.3 (http-4) whole image is non-uniformly motion blurred. If the picture divided into small patches, the PSF will not be same in every patch.

Figure 2.2. *Non-uniform motion blurred image while background is clear*

Figure 2.3. *Non-uniform motion blur*

2.3. Deblurring

Motion blurred images are needs to be restored to obtain clean and sharp images. This can be done with some linear or non-linear algorithms. In linear algorithms, the image is equally restored. Non-linear algorithms analyze the noise and it use statistically calculations. Moreover, some algorithms use the clean image or initial PSF for restoration operation. The algorithms can provide iterative or non-iterative solutions. An iterative algorithm such as Richardson-Lucy, Landweber and Iterative Wiener Filter applies restoration process to the blurred image and repeats this process iteratively to increase deblurring performance. Even if the number of iteration increases the performance, the processing time required for this job costs much more. A non-iterative algorithm does the calculation operation single time and outputs the result. Its calculation speed is higher than iterative methods, but the performance may be different related to how strong the algorithms are. Artificial Neural Networks is another option for deblurring operation. It can both restore the image and enhance the pixels to obtain super resolution. Neural networks can also be used to detect the PSF of the motion blurred images blindly. The found PSF can be used with well-known algorithms such as Wiener deconvolution function to restore the image.

3. ARTIFICIAL NEURAL NETWORKS

Neural network is one of the machine learning type that has many algorithms inside. Its structure is inspired from human brain. It is mostly used to classify or cluster and extract the features from input data. The type of input data can be various such as video, image, text, audio and so on. The input set of data feed as input of the neural network and their labelled outputs are required at the output. The neural network is expected to find similarities between data, extracting features and classifying. This is done with training the network. Training the network requires huge dataset and there are lots of important parameters to be set.

Neural network design is very important and training a specific data may not be appropriate with every neural network architecture. Moreover, training conditions, training parameters, type of the layers and their designs are other important conditions. These things may increase the stability of the network accuracy and decrease the time required to train the system well. The training speed can be increased by using transfer learning which will be described in next sections of this thesis. Training provides to solve the specific problem with finding weights in layers. The problem usually has unknown input to output relationship. Neural network is expected to find a solution for non-linear input output relation. Usually linear problems are solved mathematically while non-linear problems are solved with iteratively converging to approximate solution. When neural networks are used, it is not important to find how network works besides how successful results are obtained.

There are 3 different layers in a neural network:

- Input Layer (All the inputs are fed in the model from this layer)
- Hidden Layers (hidden layers are used for processing the inputs received from the previous layer)
- Output Layer (In the hidden layers, input data is processed and processed data is now available at the output layer as label or in required format)

The most of the image related neural network aims to classify the input and labels the output. Generally, outputs are labels; the expected result is just a number. The input and output and its unknown transfer function can be defined as a system. The transfer function of the system can have linear or non-linear transfer function.

Giving an unknown input(x) to the trained network (h), our aim is obtaining desired output(y) correctly. There are 2 types of problem which can be solved using neural network [17]:

- Forward problem
- Inverse problem

In forward problem, a transformation is applied to the input image. In inverse problem, the aim is obtaining the input image. Inverse problem can be solved by using blind or semi-blind methods. In blind inverse problems, the input image is tried to obtain without any knowledge about the system or input image. In semi-blind inverse problems, the knowledge about some parameters of the system or input image may be known.

3.1. Neural Network Layers

In neural network, there are various types of layers which construct the network. Every layer has specific ability and cannot be used for all kind of network. The operations in each layers can be expressed mathematically and implement with an algorithm.

3.1.1. Convolution

Considering we have 6x6 image. If we convolve the input image with 3x3 filter (stride=2, no padding), the output is obtained as 2x2 image. Stride means the count of pixel shifts during convolution process. An example is shown in Figure 3.1 (http-5).

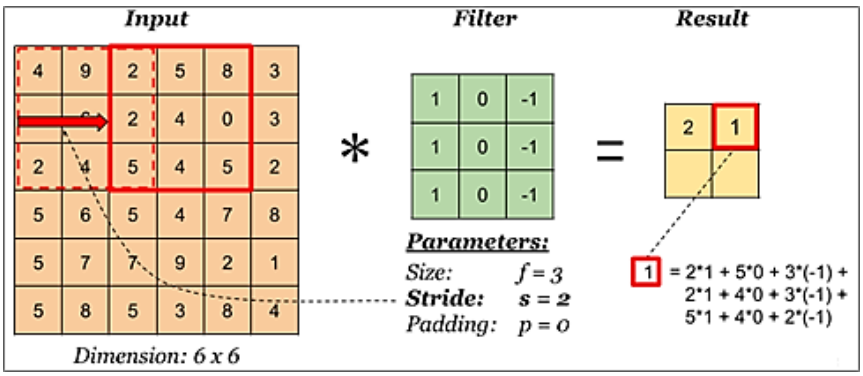

Figure 3.1. *Convolution operation in neural network*

2D convolution layers are used to detect features. It extracts high level features from objects. In the beginning of the network, convolution layers are responsible for learning low level features such as edges, color transition, gradients, etc. In the next layers of network, convolution layers have ability to detect high level features.

3.1.2. Pooling Operation

It is useful to decrease the size of data, that provides less calculation and the found features can be detected precisely better. There are a few types such as max pooling and average pooling. Also, authors of [18] suggest a new method which is called alpha-pooling.

This method basically a combination of both average and max pooling with a trainable parameter of α. The advantage of alpha pooling is increasing the image recognition accuracy. Every layer has its own α parameter and has unique alpha pooling layer. The paper also explain why max pooling is not the optimal. Mathematically, in max pooling operation, the min or max value of pixel is selected in a specific window size. In this example, poolmax with size of [2, 2] window is applied. Pooling windows of size 2x2 are used, 75% of the information provided by the input feature maps is lost. An example for max pooling is shown in Figure 3.2 (http-5).

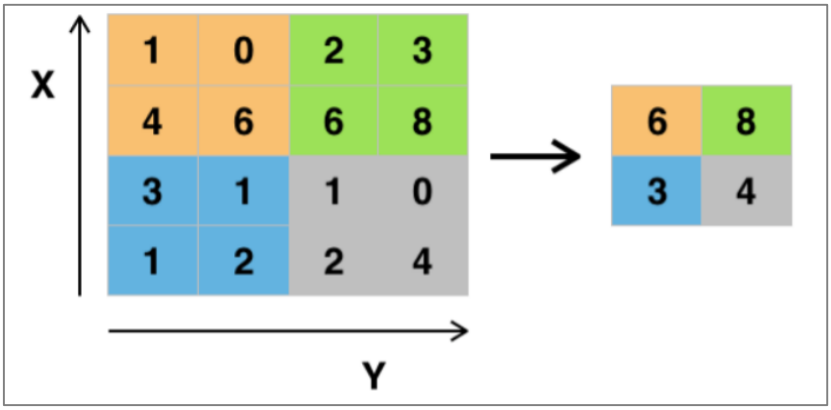

Figure 3.2. *Demonstration of the max pooling algorithm*

3.1.3. Transposed Convolution

Can be said deconvolution process which is not the mathematically a proper deconvolution operation. This can be called as transposed (gradient) of the convolution in two dimension.

3.1.4. Dropout

Dropout layer [19] is a neural network layer that increases the training loss a bit. It sounds like it makes the training worse, besides it helps to overcome overfitting problem. As the number of convolution layers increase, the depth of the neural network increase. The more neurons are more prone to overfit in the system. Basically, dropout layer is cancelling some neuron connections in the network and it prevents interdependencies from emerging between nodes. This provides to learn more features and robust training process.

3.1.5. Normalization

In this process, the inputs of each layer is auto standardized during training. This technique allows each layer to learn independently. May speed up learning. Layer normalization [20], Instance normalization [21], group normalization [22] and batch normalization [23] are some of the normalization types. The mostly used one is batch normalization.

3.2. Loss Function

Loss function is optimization score function that provides to find difference between actual label and predicted value. The robustness can be measured from this function. While loss value decreases, the performance of the model is getting better. Mean squared error and mean absolute error methods are some of the widely used loss functions to construct neural network model.

3.3. Optimization Algorithms

To find the transfer function, the weights must be found. Therefore, the training process usually requires a huge dataset and good computation power.

Training is just a process. After the process starts, it continues with given number of epoch. There may be problems such as overfitting or not to converge to labels. There are lots of optimization algorithms which may overcome these problems. Stochastic gradient descent (SGD), Momentum, Nesterov Momentum, Adam, Adagrad and RMSProp are the popular optimization algorithms (http-6). SGD is the most used optimizer algorithm which has a proof about the stability of this system and always finds a solution [24].

In SGD optimizer, learning rate is an important parameter. It simply guides the optimizer to how much movement it may step over in the direction of gradient.

While converging to solution, small steps or bigger steps may fail due to the shape of the error surface as shown in Figure 3.3 (http-7).

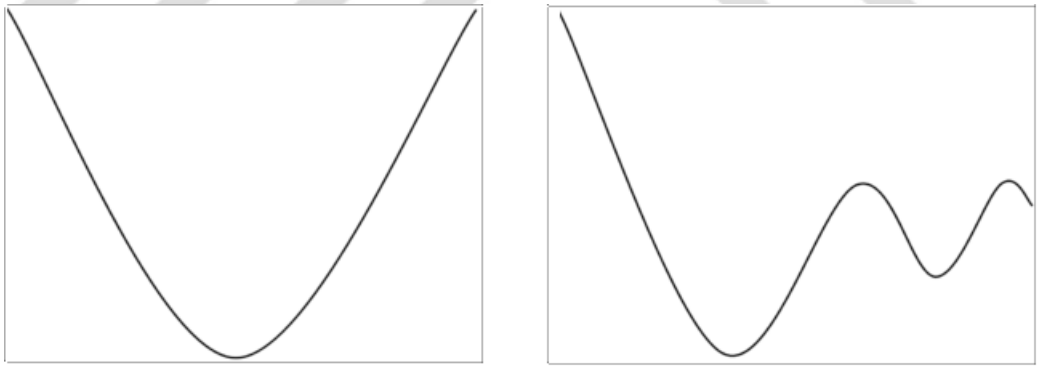

Figure 3.3 *Convex error surface and non-convex error surface*

The convergence failure leads to decrease the performance of training. It may be very problematic to choose learning-rate value randomly for initial training. The accuracy in the system may not converge and the loss may go worse. If the network is initially starts with a high learning rate, the system whose weights are not optimal will fit into the optimum weights faster. Unsupervised method [25] which suggest decreasing the learning rate 3 times until divergence not happen. This is called as coarse-tuning. After coarse-tuning, a decreased learning rate will be better for fine-tuning. The training process should be split into parts. In

each training part, the learning rate can be updated to get minimum loss, maximum accuracy. If the loss is being constant, the changes in the learning rate can be increased even if the losses increase. If the loss continuously increases, the weight updates diverge from the optimum. On the other hand, the losses may increase and decrease quickly, then converge to the optimum weights [26].

Classic SGD is enhanced with momentum parameter and this type of SGD is the most popular optimization algorithm. Lots of state-of-art models have it. It is basically works as moving average on gradients. The momentum increases and accelerates the weight updates in the true direction. The derivative of the loss function is calculated with mini batches but not with exact loss function.

Even if training looks better and obtain high accuracy, the overfitting problem can show itself. The overfitting is unrecognizable with a small number of dataset. The dataset must be huge enough. The most important thing is separate test dataset should not be used during training process. During training process batch size and number of epoch is another important thing. In every epoch, the training accuracy should be better compared with previous one. If accuracy is stable and not changing, the learning rate in SGD might be decreased for fine tuning. Momentum parameter can be changed to monitor if the result is getting better or worse. The momentum is another important parameter which helps to speed up the gradient in the right way [27]. The batch size can be increased to use computer power and parallelism but it is dangerous after the number of 32. The ideal number of batch sizes are suggested in [28] is 2 to 32 for efficient convergence and reliable performance.

3.4. Transfer Learning

Transfer learning is a task to improve the performance of new model using the weights of previously trained network. This task usually used to train the model with related purpose such as object detection, language processing which use text data. In transfer learning, previously trained features according to learned filters are directly used. The weights of layers can be used directly with pre-trained weights or they can be developed by setting the layers as trainable. In both approaches, the fully connected layers and prediction layers are modified for specific types of classification or purposes. The prominent advantage of the

transfer learning is; it reduces the time required to train a network. A well trained network may need days or weeks to be accurate. Someone can use transfer learning and save the time.

3.5. Training Datasets

There are lots of training datasets publicly available on the Internet. This is very useful to save the time and compare our works based on a well-known reference. Commonly used image classification datasets such as ImageNet [29], Cifar10, Cifar100 (http-8), Mnist (http-9) and some others have train, validation or/and test datasets. In this thesis, Cifar10, Cifar100 and Places365 training datasets [31] are used for training. Cifar10, Cifar100 and Tiny-ImageNet-200 test samples are used for both validation process and test process. The training dataset is used to create the mode with biased score and makes model to it, where the test and validation data is to score the performance of system. The resulted validation score shows the error rate of the trained model [30].

3.5.1. Tiny-ImageNet-200 Dataset

Tiny-ImageNet Challenge is the competitive project for Stanford University CS231N lesson. This challenge aims to classify images. The challenge needs to use Tiny-ImageNet-200 dataset (http-10) and classify the images with minimum loss as possible. The submissions of the trained network are transferred to a main computer and scored. There is a leaderboard and successful results are graded. This useful and various classed images are downloaded and used in this thesis. A few of the images are shown in Figure 3.4.

Figure 3.4. *Noisy test samples from Tiny-ImageNet-200*

Lots of very noisy and low resolution images exist in Tiny-ImageNet-200 dataset. Actually, these bad images are useful to measure the performance of our trained model. In order to show the resistance of the model to the noise and low resolution, it is an opportunity to have originally untouched images. This will provide to compare any future projects with this thesis. A few of them are shown in Figure 3.4. The dataset has 32x32 images. Therefore, in order to display the small dataset images in Figure 3.4, an upscale is applied. Because of the upscale algorithm, the pictures are shown less noisy and more detailed, but original ones are quite bad quality.

3.5.2. Places365 Dataset

In 2016 Places Challenge, Places365 standard dataset [31], which has 1.8 million 256x256 training images have been shared as public. In 2017 it is updated with newly 10 million image [31]. The dataset has called Places365 because it has 365 different classified place of images and labels. It is used for scene recognition, generic deep scene feature extraction. The original dataset is 476 GB sized and called Places2 database that has 6.2 million more images compared to standard Places365 dataset (108 GB). Also there are lightweight versions of Places365 dataset available. This small dataset, which has more than 36000 images, is downloaded for training purpose. A few of images are shown in Figure 3.5.

Figure 3.5. *Some blurry and unfocused images of Places365 Dataset*

There are lots of advantages of this dataset. It has unfocused, blurry, sheared and shrink images. The main purpose of selecting this dataset is, it has various kinds of objects and places. The aim is training the system to recognize motion blur, not to classify objects,

so lots of objects makes the training powerful and resistant to overfitting problem. The objects are not being memorized, different features of the edge types are expected to learn.

3.5.3. Cifar-10 and Cifar-100 Datasets

Cifar-10 and Cifar-100dataset is the most preferred image dataset (http-8), for training neural network. It is firstly used in a master thesis [32] by one of the collector of this dataset. Its name 'Cifar' which means 'Canadian Institute for Advanced Research' and the numerical part of the dataset names ('10' or '100') represent the number of different classes that exist in the dataset. It has 32x32 pixel RGB images. Randomly selected one image from the classes of Cifar10 is shown in Figure 3.6. This dataset is very useful to pre-train the network to learn small features in large images. Also, there are many examples about how to use this dataset and train it. The examples can be used with C programs, MATLAB and Python platform.

Figure 3.6. *Cifar10 dataset sample images for 10 classes*

3.6. Common Networks for Object Detection

There are lots of neural network models and some of them have advantages due to its architecture. Also, type of dataset and type of application are important for selection of specific network model.

3.6.1. VGG16 & VGG19

VGG is abbreviation of Vision Geometry Group of Oxford University (http-11). This group focused on lots of computer vision projects. They submitted a paper [6] based on object detection and called "Very deep convolutional networks for large-scale image recognition".

It became very popular in ILSVRC2014 (ImageNet Large Scale Visual Recognition Challenge 2014) (http-12). They presented various types of networks related to depth of model. The layers of VGG models are given in Table 3.1 [6].

Table 3.1. Proposed deep VGG models

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 (224 × 224 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					
soft-max					

The proposed VGG models shown in Table 3.1 are A) Vgg11, A-LRN) VGG11 with local normalization layer, B) VGG13, C) VGG16 (with 1x1 convolution layers), D) VGG16, E) VGG19. The deepest model is VGG19. It has only 3 more convolution layer compared to VGG16. VGG19 has 144 million number of parameters while VGG16 has 138 million.

They trained 1000 classes and 1.2 million images and obtained %0.073 classification error. This means that their network has approximately %92.7 classification accuracy. With this success rate, VGG became first and second place in this challenge. The founders of this network model proposed that increasing the depth of the network with 3x3 convolution filters are increasing the accuracy. They also mentioned, why keeping the stride and convolution filter sizes small increases the performance. In the paper they proposed a few neural network models with variable size of layers and input size of image is 224x224. Time required to train VGG19 may be a little slow compared with the VGG16 but the performance is satisfying. As a conclusion they proved the importance of the depth.

3.6.2. Resnet50

ResNet is new network architecture which introduced itself as Residual Network. Many other network has limited capability during training. After some training stages, the accuracy saturates and may be decreases if classification types added to the system. ResNet is an appropriate architecture to cope with these problems [7]. The founder of the ResNet has proved that ResNet is more successful than VGG16. The proof is made with using PASCAL VOC 2007/2012 test sets (http-1).

Another advantage of the Residual network is, the trained layers can be removed and the network can continue to work well. This shows that the layers are not completely strong dependent to the previous or next layers. However, it is impossible with feed-forward network types such as VGG and AlexNet [33].

The idea behind the residual network is simple. The input x is fed into system denoted by F(x) and output is obtained as H(x). The residual part F(x) is difference between input and output, which is shown as;

$$H(x) = F(x) + x$$
$$F(x) = H(x) - x \tag{3.6}$$

The residual layer is simply demonstrated in Figure 3.7.

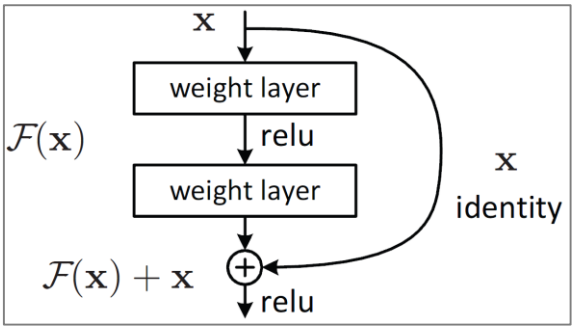

Figure 3.7. *Residual Layer building block*

Learning residuals enhances the system and increases the performance. Because of the equation, the gradients from last layer to first layer can be skipped and only forwarded. Therefore, it decreases the computation and helps to forward real gradients to other layers without changing it. A sample piece of code is shown in Figure 3.8.

```
def Unit(x,filters):
    res = x
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    x = Conv2D(filters=filters,kernel_size=[3, 3], strides=[1, 1], padding="same")(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    x = Conv2D(filters=filters, kernel_size=[3, 3], strides=[1, 1], padding="same")(x)
    out = keras.layers.add([res,x])
    return out
```

Figure 3.8. *Python code for one residual block of ResNet50 architecture*

3.7. Training Frameworks

There are lots of programming languages used for deep learning. MATLAB, Python, C++ are commonly used languages. In this thesis, Python is used with the most popular deep learning library Keras (http-13). Alternatively, Caffe library that is developed by Berkeley AI Research community or MatConvNet library [34] of MATLAB could be used. Keras is preferred because the documentation, examples, support and bug fixes for this library seems better. Using Python and Keras, sample Cifar10 object detection examples are tried. The current training hardware is notebook and its graphical processing unit (GPU) is Nvidia

GT650m. Because of slow GPU card, training the example took too much. Therefore, in order to train millions of images, the need of new GPU card noticed. The fastest GPU cards for training the neural purpose are expensive. An Nvidia Tesla K80 is 1389$ in online shopping website Amazon (http-14). In addition, this GPU requires cooling system and special computer hardware such as main board or RAM memory. A little search in Google showed that, Google is supplying GPU training machine which is controlled by web browser for scientist, researchers and students. The machines have Ubuntu 16.04 operating system and can be integrated with Google Drive. This provides sharing the results, running the python codes and save the files more flexible. This system is Google Colaboratory (http-15). This platform supplies Nvidia Tesla K80 or Tesla T4 (beta) GPUs. In addition, a tensor-processing unit (TPU) can be used but it is slow, because central processing unit (CPU) is running for the training task. A training task in the Google Colaboratory is approximately 10 times faster than training in notebook with Nvidia GT650m. In this thesis, the training task is done and evaluated using Google Colaboratory platform.

4. TRAINING THE NEURAL NETWORKS

In the training process, VGG16, ResNet50, Jian Sun's Network and VGG19 is used as neural network model. To obtain better results, these architectures are trained. Some overfit and non-convergence problems are happened during training but, using some technics and training with more dataset is helped to improve accuracy. In this thesis, the motion blur kernels are sampled with 45º (degree) interval and with a length of 2-4-6-8-10 pixels that is totally 20 different blur kernels (point spread functions). These PSFs are produced using MATLAB. An example of code shown is in Figure 4.1.

```
%%  Blur kernel
motionblurpix=10; %Motion blur length as pixels.
motionblurDegree=90; %Motion blur angle as degree.

%Find Motion Blur Kernel or Point Spread Function (PSF)
psf = fspecial('motion',motionblurpix,motionblurDegree);
```

Figure 4.1. *Producing motion blur kernels using MATLAB*

All the blur kernels are produced with different angles (Θ) and lengths (L) and saved as '. mat' file format. Two produced blur kernels are shown in Figure 4.2.

a)

	1	2	3	4	5
1	0	0	0	0.0133	0
2	0	0	0.0665	0.2269	0.0133
3	0	0.0665	0.2269	0.0665	0
4	0.0133	0.2269	0.0665	0	0
5	0	0.0133	0	0	0

b)

	1	2	3	4	5	6	7	8	9
1	0.0625	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.1250	0.0625

Figure 4.2. *Blur kernel as 2D float array a) Θ =45º, L=4 and b) Θ =0, L=8*

Input images are selected as 32x32x3 colorful images because reducing the size of images decreases the number of learnable parameters in the network and thus reduces the

training time. If we assume any shape has twenty pixels at the center of 32x32 image, the blur length interval should not be bigger than ten. Therefore, the motion blur lengths are selected with a limited range. Otherwise, a part of motion blur artifacts of objects would be outside of the 32x32 image.

4.1. Data Augmentation

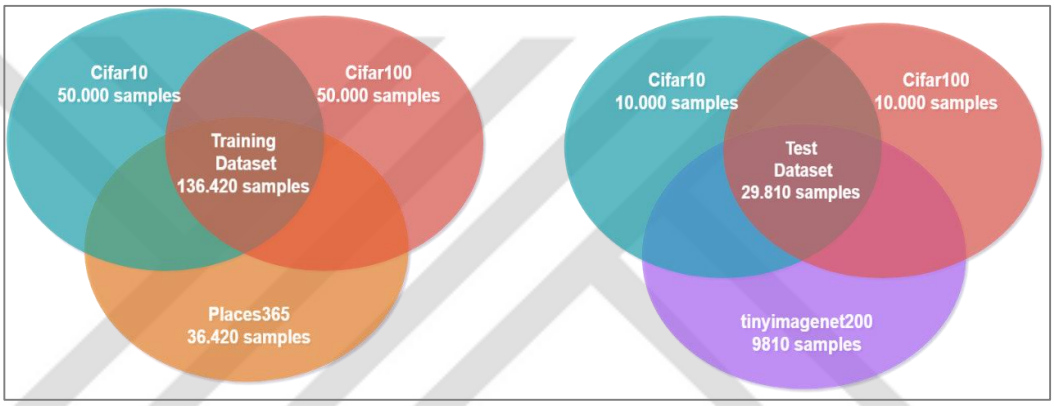

Figure 4.3. *Used training and test datasets*

Before training, the dataset must be synthetically modified. The used datasets are shown in Figure 4.3. An augmentation method is applied to boost the performance of the system and prevent the learning of unnecessary features. For example, if somebody tries to find features such like nose, ears and eyes of animals; lots of animal images are required. Training the network with only dataset of cats is a big mistake. The network can fit for shape of the cat and may not be successful for dogs or another animal. Therefore, various kind of animal datasets should be given as input to the network. It can be said that the dataset is important based on this example. In this thesis, the motion-blurred pictures are generated from Cifar10, Cifar100 and Places365 datasets. All these datasets are randomly blurred with the kernels which are specific to this thesis. Initial training is started to get an idea about the systems. VGG16 network and Cifar10 datasets are selected and monitored during coarse training. The results showed that the network memorizes the features of shapes because; Cifar10 has only ten different classes. To improve the validation accuracy and prevent the

overfitting, the Cifar10 dataset is used as training dataset and Cifar100 and Places365 is used as validation datasets. From that point, training dataset and test dataset are different classes of images. However, the only common point is PSF of the images. This tricky method increased the validation accuracy a bit but while the training accuracy goes to 100%, the validation accuracy is not increasing as quickly and saturating at some value without changing. Changing the learning rate, momentum, number of batch size or epoch did not help to overcome this problem. Training data always had overfit issue because the VGG16 network is designed for object recognition and classification. Furthermore, Cifar100 and Places365 datasets are used as training dataset but accuracy did not improve much more. It showed that, the number of classes is effective but not so much. The final state of art technique applied and the validation accuracy increased almost linearly with training accuracy. Cifar100 dataset is selected and every image is convolved with each PSF. An image now has 20 different motion blur corrupted samples, these means 50.000 x 20 = 1.000.000 images.

4.2. Using Dropout Layer

In early stages of the training, the overfit problem was usually happening. To fix that, dropout layer is used. Without dropout layer, the validation accuracy was increasing very fast but saturating at a point and did not improve much more. Dropout layer with 50% percentage drop rate before the softmax layer gave better results. Even if it helps to get better results, the time required to train increased. However, it is a method to resist the overfit problem so it significantly improves the performance.

4.3. Fine Tuning

In the beginning of train, SGD parameters are selected to obtain maximum training performance and reduce to time required for training process. For VGG16 and VGG19 networks, the initial momentum value is selected as 0.8 and learning rate 0.05. The training with these initial parameters, the loss did not converge. After that, coarse tuning is made with momentum=0.5 and learning rate=0.005. Accuracy increased slowly with this setup. At the end of coarse tuning, fine-tuning is made with learning rate=0.0001 and momentum=0.95.

These parameters are experimentally selected by monitoring the overfit or saturation point of the accuracy.

4.4. Trained Networks

Training the system with input blurry images, predicted value $\hat{y}$ at the output of softmax layer and y is the actual label. The mean square error loss function L that should be minimized as follows;

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i^2 + \hat{y}_i^2)^2 \tag{4.1}$$

The n parameter is the number of trained samples. While training the system, the weights are updated to minimize the loss function, but the loss function may be different depending on the selection.

Training with different architecture had some difficulties, such as input shapes, output shapes, training parameters etc. Each different architecture has different characteristics but VGG16 and VGG19 are similar. Therefore, the training required long time to obtain better results and overcome some problems. In this thesis, VGG16, ResNet50, VGG19 and Jian Sun's network model are focused. After data augmentation, VGG16, Resnet50, Jian Sun's network and VGG19 models are trained. During training, momentum and learning rates are edited to obtain maximum performance. In the beginning of training, coarsely optimization parameters such as momentum and learning rate are selected. When accuracy saturated, fine-tuning is made to improve the accuracy to get the best prediction performance.

VGG19 is an enhanced version of VGG16 network. The main purpose of using VGG19 is, investigating the performance of the VGG19 is better than VGG16 or not. The performance of VGG19 is expected to be better than VGG16, because VGG19 has three more convolution layers shown in Figure 4.4 [35]. The results demonstrate that the importance of depth related to this thesis.

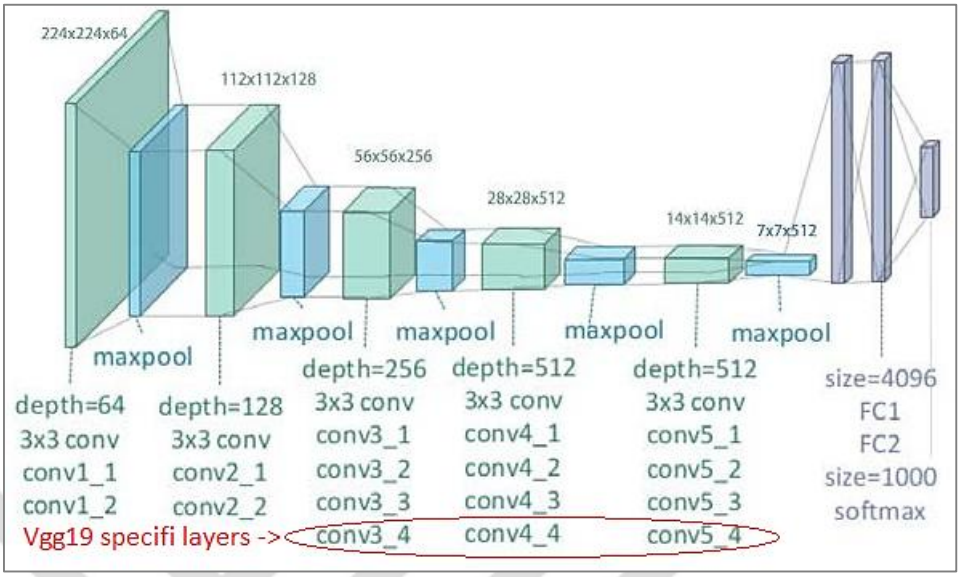

Figure 4.4. *Difference of VGG19 compared with VGG16*

Firstly, VGG19 is tried to be train without using transfer learning. The training needs more time compared with VGG16. The number of epoch is increased and fine-tuning tried but it requires much time. The validation accuracy of 97.56% is obtained. The accuracy could be increased with fine-tuning but in order to save the time, another method is applied. The idea was transfer learning with a different format. Because of the similarities of VGG19 and VGG16, all the previously trained layers are used except fully connected and softmax layers. The weights of these layers are transferred and extra three layers are added to the system. The transferred layers are set as non-trainable. On the other hand, newly added three layers, new fully connected and softmax layers are set as trainable.

5. TRAINING RESULTS

In order to obtain the main purpose of this thesis, various types of networks are trained. Different type of architecture has different behavior to our specific dataset and also different success rates are obtained. To be sure, new dataset called Caltech101 (http-16) is downloaded and tested with all the fine-tuned networks. The test accuracy results are shown in Figure 5.1.

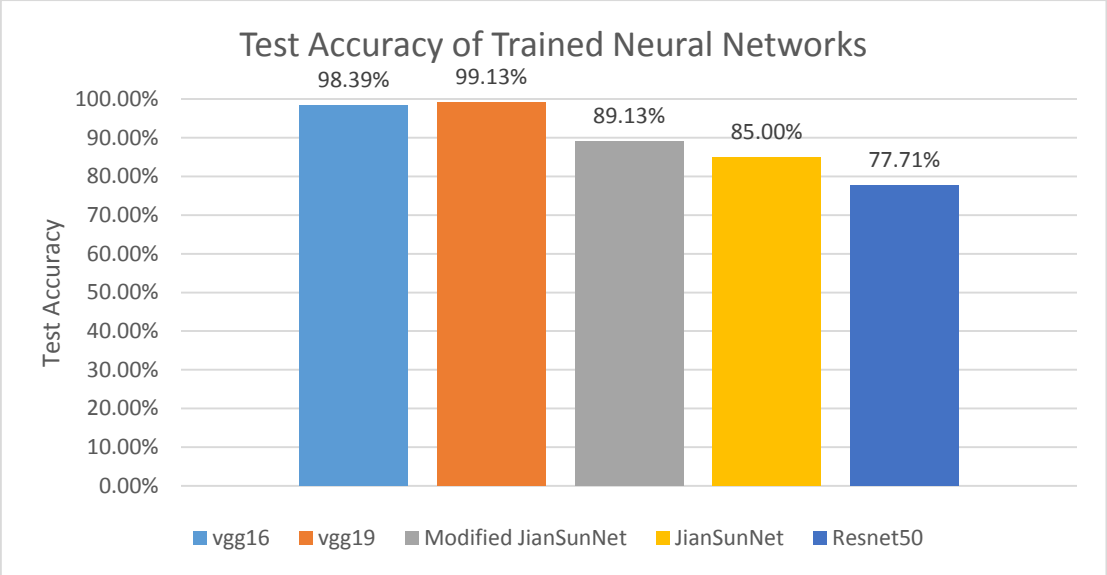

Figure 5.1. *Performance of trained networks using Caltech101 dataset*

During training, few issues are discovered. These were reducing the performance and the training accuracy. After solving these issues, performance improvement is achieved.

5.1. VGG16

VGG16 network is popular network and there are previously trained weights available which is produced by training of ImageNet dataset. The aim is obtaining better results with pre-trained network. In addition, the results will show the compatibility of pre-trained network filters with this application. From beginning of the network, up to fully connected layers, the weights are hold as non-trainable except the fully connected layers and softmax layer. The accuracy is saturated at maximum 30.32% so improvement did not happen. When

the layers are set as trainable, the maximum performance is obtained from Caltech101 dataset is 98.39%. This result is satisfying about the success.

5.2. ResNet50

With the training of ResNet50 model, firstly 73.58% accuracy is obtained. The same 32x32 images are used to train this model. Lots of training is done with 50 epoch and batch size is selected 32-64-128. While the training accuracy is 92.6%, the validation was maximum 73.58%. This shows overfitting. When the results become such like this (overfitting problem), the training parameters are changed for optimization. Sometimes, changing parameters is lead validation accuracy to go worse. The fine-tuning is done with the parameters; learning rate=0.0001 and momentum=0.8.

An article [36] shows that how to increase accuracy while training the ResNet50 with Cifar10 dataset. The article presents that the momentum should be between 0.9-0.99 and batch size should be increased to get better accuracies. These parameters are applied to previously trained network and fine-tuning is made. The batch size is selected as 128, momentum is 0.99 and learning rate is 0.0001. The final result increased to 77.71%.

5.3. Jian Sun's Network

Jian Sun's network model is implemented and trained to obtain maximum accuracy. Fine-tuning also is done and 85.0% accuracy is obtained. To show the importance of deep network, one convolution layer with 3x3 filter size is added. At the end of network, 2x2 maxpooling layer with stride number of 2 is inserted. The newly modified Jian Sun's network model is trained and maximum 89.13% accuracy is obtained. This is another proof of deep convolution layers in a network is better for obtaining the motion blur.

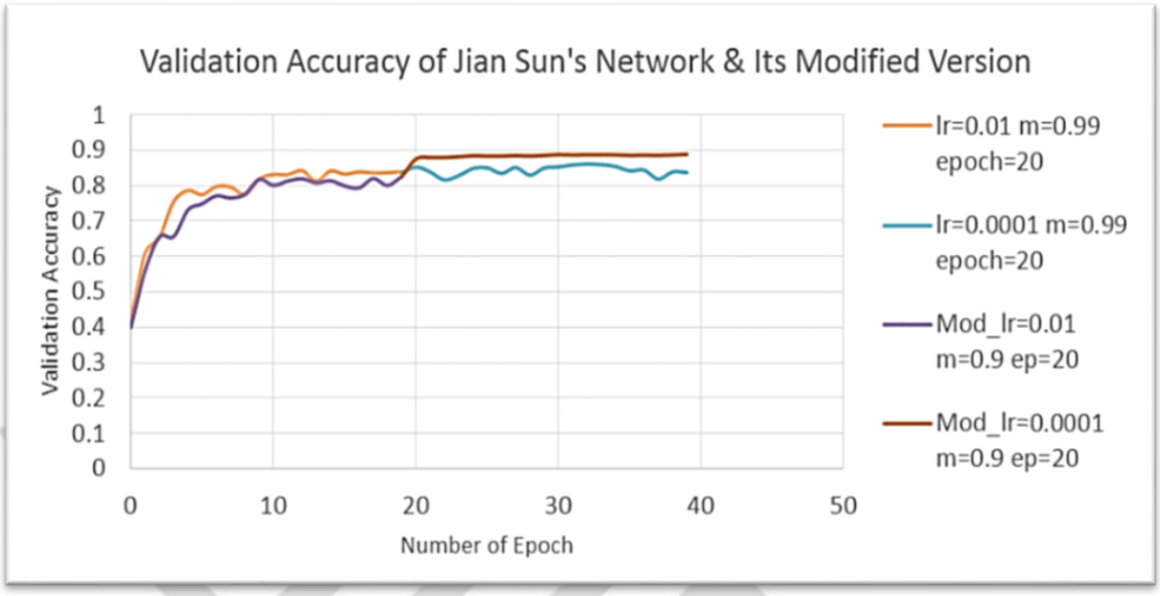

Figure 5.2. *Training result history of Jian Sun's model*

In Figure 5.2, the training parameters and accuracy history is shown. One million blurry images with their ground truth labels are trained. After 20 epochs, fine-tuning is applied. This network has a few layers so the number of trainable parameters is not much compared to VGG16. Due to small amount of trainable parameters exists in this model, the time required to train this network is also less than other models.

5.4. VGG19

Training VGG19 is made with initially random weights at every layers of model and then, transfer learning is applied from VGG16 to VGG19. The transfer learning process for this specific operation is described in detailed in Chapter 4. Without transfer learning, the training is started slowly while the other model with transfer learning rapidly becomes high accuracy. After 10 epochs, the momentum parameter of SGD is increased. In order to save the time, after 60 epochs, VGG19 without transfer learning is not trained. Accuracy graph is and SGD parameters are shown in Figure 5.3.

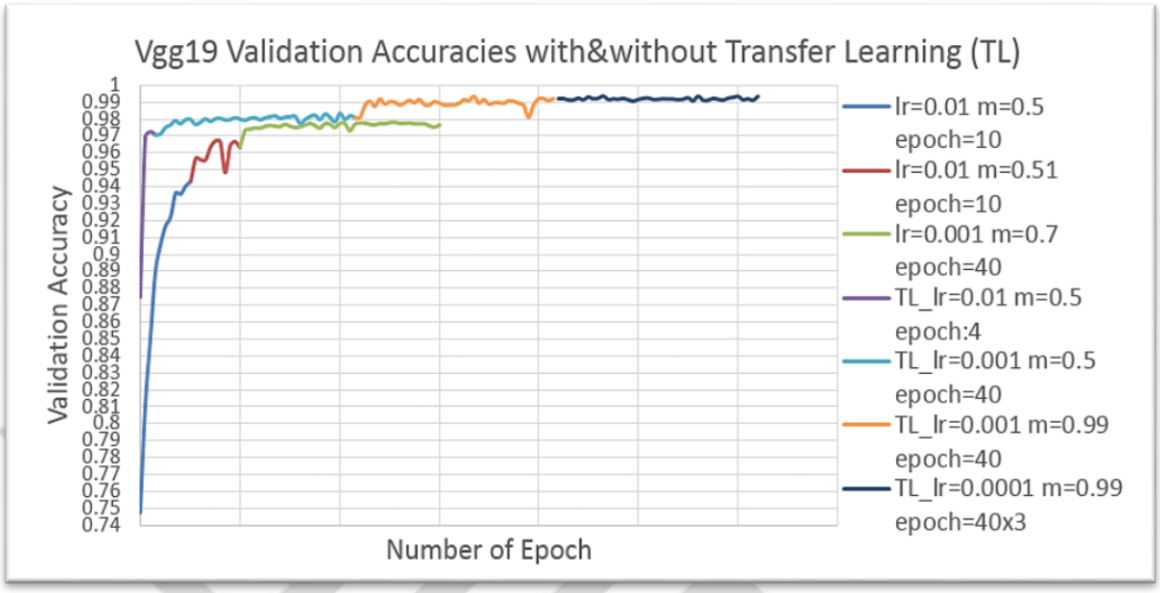

Figure 5.3. *Training results of VGG19 model with and without transfer learning*

From the graph in Figure 5.3, the transfer learning seems better about rapid rise of the accuracy and less time required to stop the training task. This proves that the operation is very useful and works to find motion blur kernel.

5.5. Found & Solved Problems Related with Training

In the beginning of this thesis, example Python codes including Keras library are searched and tried. A VGG16 example code is selected, modified and motion blurred datasets are trained. The maximum accuracy of training was 70%. Then, the 30% inaccurate part is analyzed, the classes which have 45º and 135º motion blur angles are detected as wrong. In addition, the most of the results were reverse which means 45º motion blurred image classes found as 135º and vice versa. Keras example codes were based on object recognition, so the codes are carefully analyzed and the whole documentation of Keras is read. To prevent overfit, the example has some options; flip horizontally, shear range, zoom range etc. These options are very useful when training process is based on object detection. In this thesis, zoom range leads to get wrong results about length of motion blur. The horizontally flip is used to reverses the image around the y axis so this was the main reason for founding the 45º and 135º angles in motion blurred image classes which were expected reverse. Shear range

option is led to obtain the both angle and length wrong. These options are disabled and high accuracy is obtained.

5.6. Increasing the Number of Blur Kernels

Obtaining %99+ successes with VGG19 network has showed that motion blur kernels are learned by the network. To test the accuracy of the system, more classes applied. Number of length is selected from 2 to 20 by 2 steps and angle resolution is decreased to 20º. The previously trained network which has 99% accuracy is transferred with its learned filter and it is trained. Using transfer learning, the training speed was incredibly fast. After fine-tuning, test accuracy is obtained with 87.93% success rate.

5.7. Restoration Results

A sample image is selected to observe realistic results and uniform-motion blurring is applied with a known PSF. This image is shown in Figure 5.4.

Figure 5.4. *Original image and motion blurred image with Θ =45º, L=8*

The motion-blurred images are divided into 32x32 patches and all of them are predicted using trained networks VGG19, VGG16, Jian Sun's model and Modified Jian Sun's model. The ResNet50 model is not used because it is not an appropriate model for this task. Degraded image is divided into patches with a window size of 32x32. The patch window is shifted 4 pixels in vertical and horizontal directions. The reason for shifting 4 pixel is to prevent wrong detection of PSF because of smooth surfaces. There may not be significant gradient changes, so the prediction will not be accurate. The results are shown in Figure 5.5 and Figure 5.6.

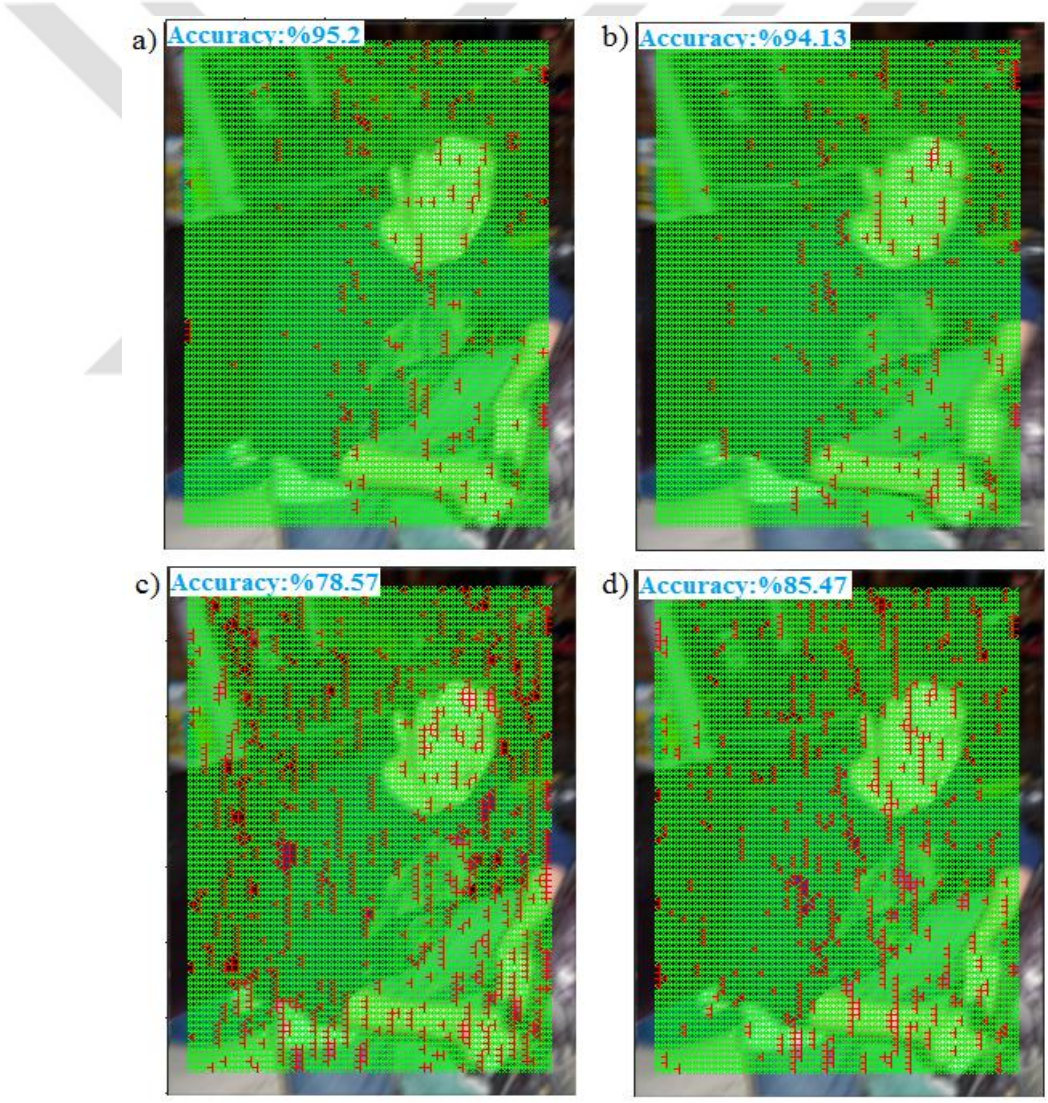

Figure 5.5. *Prediction results for the specific models: a) VGG19, b) VGG16, c) Jian Sun's model, d) Modified Jian Sun's model*

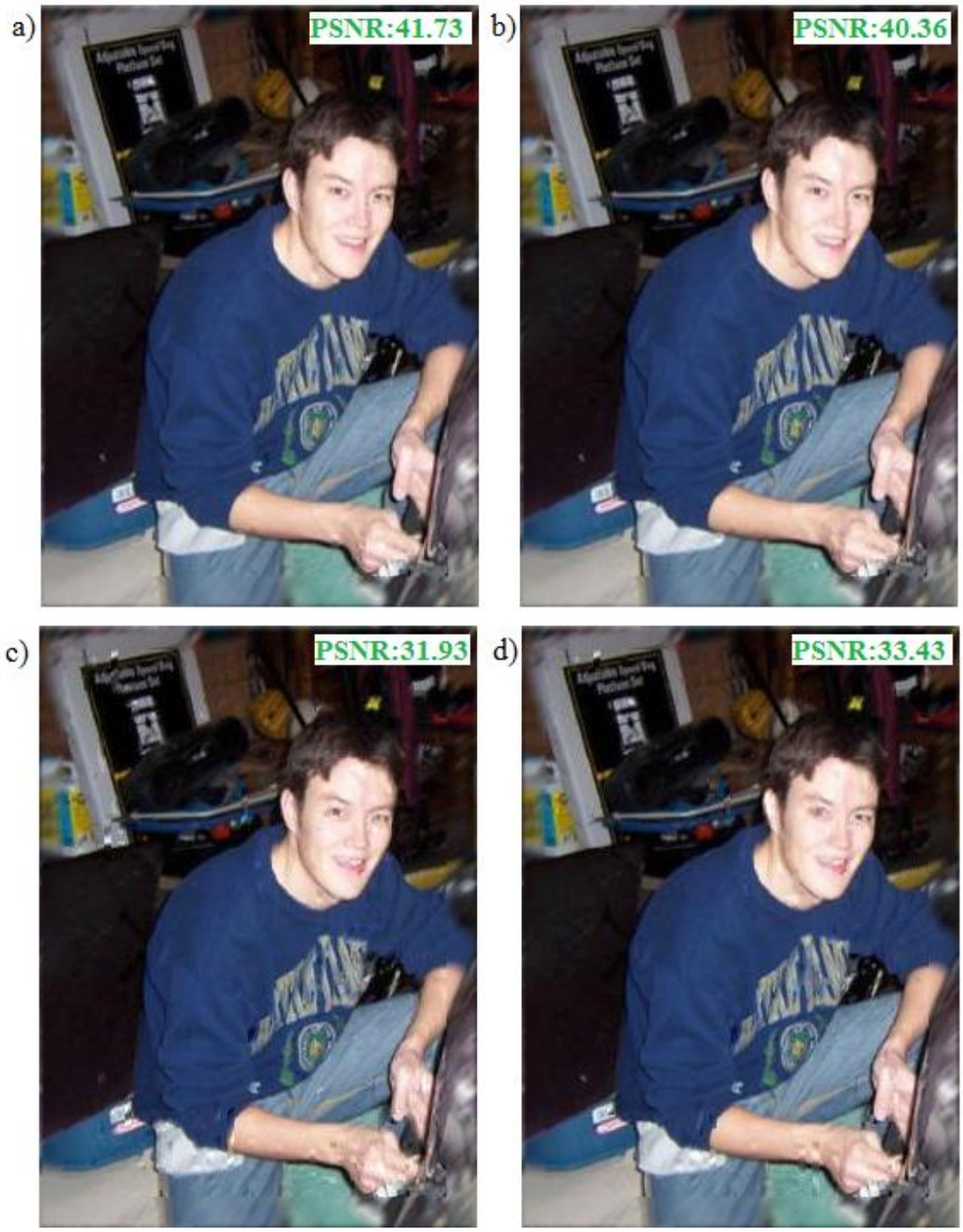

Figure 5.6. *Restoration results using predicted values of trained models a) VGG19, b) VGG16, c) Jian Sun's model, d) Modified Jian Sun's model*

In Figure 5.5, correctly predicted PSFs are marked with green plus sign and wrong prediction values are shown with red markers. All the green and red markers are inside the region of interest (ROI). The accuracy values are shown at the left top corner of the restored images. The region of interest is selected 32 pixels less than the original image size in both directions. Because of the deconvolution process, reducing the size is necessary. Otherwise, padding should be used but padding operation will cause bad quality of restored image at the output of deconvolution process.

In Figure 5.6, simple restoration task is done with using inverse Wiener filter and Lucy-Richardson deconvolution method [1]. Both restoration methods require a PSF and it is very critical parameter. Inverse Wiener filter directly process the motion-blurred image and no iteration is done. Therefore, any wrong selection of PSF decreases the PSNR of restored image and makes the restored image worse than the blurred image. However, iterative method has an advantage when the prediction results are wrong. In restoration process, when the PSNR value of deblurred patch is lower than 20 dB, Lucy-Richardson iterative deblurring method is applied, otherwise inverse Wiener filter is applied. This algorithm is chosen to achieve better quality of image for every patch in this simple restoration process.

Figure 5.6 proves that VGG19 has the best prediction accuracy. The performance of VGG16 is almost same with VGG19. Jian Sun's model has the lowest PSNR so the quality of image is not good compared to others. Modified Jian Sun's model has distortions in some patches but it is better than Jian Sun's original model. The distortions can be seem around the face, eyes, and wrist regions. In addition, the PSNR values (in dB) show the quality of the restored images and this helps to compare the performance of each neural network model.

Another test image is blurred with a PSF that has blur length=6 and direction=0 degree. This image is shown in Figure 5.7. In addition, length 4 and 2 are tried in some images but not showed in this thesis. The results are analyzed and it is concluded that when the motion blur length parameter of PSF decreases in the distorted images, the obtained accuracy from the results increases. This is discussed in Chapter 6 and explained using Figure 6.1. The accuracy results of test image for each network model are shown in Figure 5.8. The green markers means true detection and red markers shows wrong predictions of PSF in current 4x4 pixels of rectangular area. The same routine in the first restoration example is followed.

Figure 5.7. *Another original test image and its motion-blurred demonstration with Θ =0º, L=6*

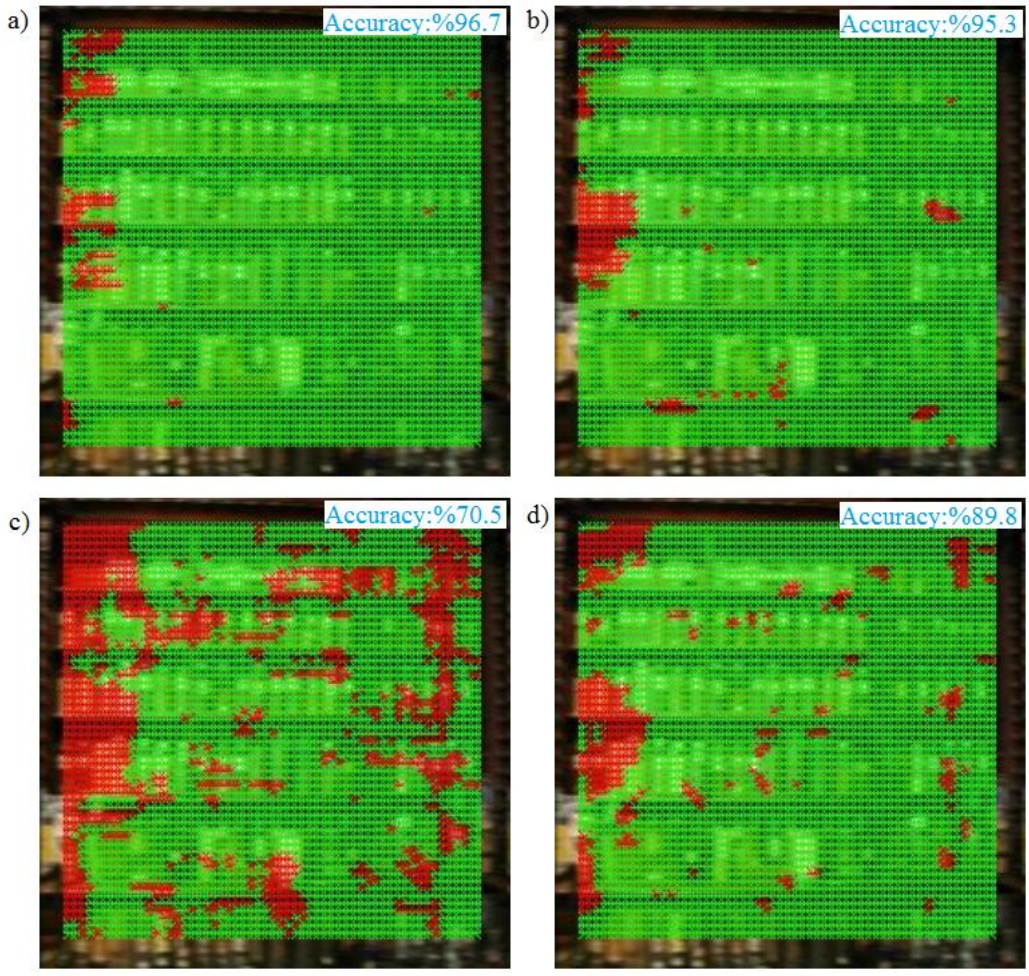

Figure 5.8. *Prediction results for the specific models: a) VGG19, b) VGG16, c) Jian Sun's model, d) Modified Jian Sun's model*

The window size of current network is 32x32 pixels. It is also known that high variation in gradient changes helps to detect the motion blur better. These variations are usually motion blurred versions of edges. If the test images that have smooth surfaces, the gradients become less detailed and prediction accuracy goes worse. In addition, wrongly selected ROI patches decreases the prediction accuracy. If the gradients exist in the middle of 32x32 window, it can be said that ROI is well chosen and due to this, the detection rate of motion blur parameters becomes accurate. In Figure 5.9, the ROI parts of restored images are shown. These results are found using the prediction results of different neural network models. In order to see the patch results more detailed, a region in the image is selected and scaled up. The scaled part is shown in the top right corner of the restored images with red box.

Figure 5.9. *Restoration results using predicted values of trained models a) VGG19, b) VGG16, c) Jian Sun's model, d) Modified Jian Sun's model*

6. FUTURE WORK

For future work, alpha pooling layer presented by [18] or other custom layers can be applied for a specific model to increase accuracy of training. Lots of top artificial neural networks are designed for object detection. Custom neural networks to detect motion blur are poorly designed and cannot give better result compared to VGG19. The proof is made in this work, VGG19 can be modified or new models can be designed.

Once the blur kernels are found successfully, both non-uniform and uniform motion blur problem can be solved. On the other hand, the found blur kernels (PSFs) should be precise to restore the quality of image. Furthermore, new methods will be applied to obtain the exact PSF. These can be done by applying an algorithm to degraded image which does contrast and luminosity normalization. Then, training the frequency spectrum of pre-processed motion blurred images is expected to increase blur angle and length resolution.

The additive noise and its effect on results can be estimated. This will show how robust the network models are against to the noise.

Transfer learning can be applied to network model and more PSF classes can be trained in order to reduce the time required to train the model. In the network layers, the previously learned filters are expected to be useful for training the new model which has large classes of PSF. The benefits will be observed.

An enhanced version of Generative Adversarial Networks (GANs) [37] can be used to generate the blur kernel itself as 2D image with feeding the kernel parameters which are found in this thesis. This provides to learn non-uniform distribution around the edges and find the non-uniform motion blur filters. In addition, it increases the restoration quality of mini patches.

In [9], non-uniform motion blurred images are divided into patches and the predicted parameters for each patch are showed on the image as vectors. This is a good idea to recover non-uniform image. The direction and length of the vectors obtained from the mini patches are expected to be changed smoothly over the image. In chapter 1, the reasons that create motion blur show that a camera cannot rotate or accelerate suddenly. After the results obtained from prediction of mini patches, instead of using MRF, the parameters can be corrected using extended Kalman filter or complementary filter. The performance of the

system will be compared. Similar problem happens at the output data of 9-axis inertial motion sensors.

In this work, neural network input image size is 32x32 and it is ideal to detect PSFs from 0 to 10 length. Because of the discontinuity in the borders, the neural network cannot predict very well. The blurry parts or features may be outside of the image, so it may not be possible to detect blurry features. In some filters that are shown in Figure 6.1, this problem or the ringing effects and motion blur artifacts is thought to be handled by discarding the features that are next to borders.

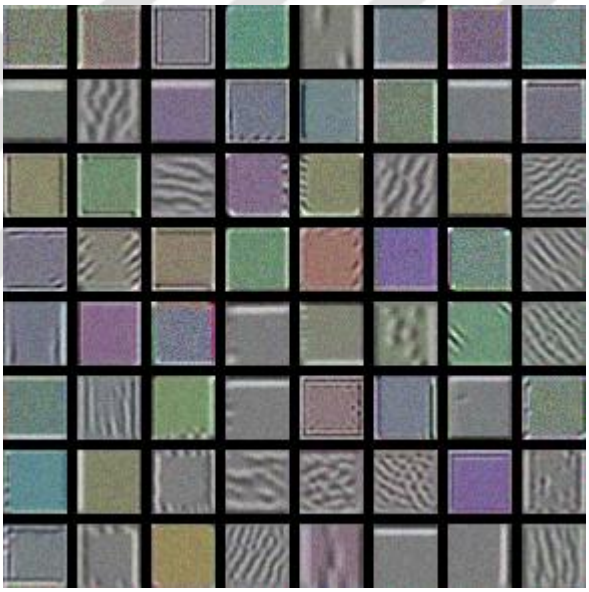

Figure 6.1. *A few learned filters taken from conv1 block of Vgg19*

The neural network learned a mechanism to handle these problems by selecting these filters itself without any intervention. A few filters have square borders up to 8 pixels width and height, so the blurry edges should be exist inside 8 to 24 pixels. Therefore, centering is required during the detection of higher blur lengths. A feature work can be investigated to detect the blurry edges and then the patches can be selected by centering the blurry edges. In addition, when the motion blur length is bigger than 10, the detection accuracy decreases because of centering problems. However, detection of higher blur lengths is possible by increasing the network input size from 32x32 upto 224x224 (max input size of VGG19).

7. CONCLUSION

In this work, the main purpose of this thesis is described; investigated, applied using neural networks and the results are successfully shown. There could be an advanced restoration process but simple restoration is made and this can be done in future work that is expressed in Chapter 6.

The results that are obtained from this work shows that VGG19 is the best model compared with other models, because when the depth of the network increases, the accuracy increases as well. According to the results, it can be said that VGG16 has learned less features compared to VGG19. This cannot be generalized for all types of studies that are related with neural networks but it meets the requirements of this project. Jian Sun's network is not much appropriate because it has only two convolution layers, not deep enough and does not have an ability to perform reliable results. It may be useful for only a limited number of images or limited type of classes. Using the trained network models in this thesis such as VGG19 and VGG16, they can be used to improve the performance of the Jian Sun's custom MRF method.

Transfer learning using constant ImageNet weights did not perform accurate results. It can be said that the filters learned during training of ImageNet dataset, are not appropriate filters for motion blur kernel classification. The main reason is, the features learned from ImageNet dataset are clean and sharp but not blurry. As it expected, the neural network has learned the color gradients, different types of blurry edges and maybe other types of operations that have never discovered yet.

ResNet is not appropriate model for the purpose of this thesis, if learning rate or momentum changes, the whole network effected badly and accuracy drops suddenly. The learning rate decreased and momentum changed for the purpose of fine tune, but system accuracy decreased. The main reason of this can be skip connections and batch normalization layers in this network. The results show that forward network models are more successful than residual network ResNet50 about kernel classification of motion blur.

The results obtained from this thesis will be useful for studies based on motion blur parameter estimation. The results and proofs will help to determine how network types or models are chosen and which modifications should be applied to the layers to obtain accurate motion blur kernels. These will also improve the performance of the restoration tasks.

REFERENCES

[1] Khan, M.K., Morigi, S., Reichel, L, Sgallari, F. (2013). Iterative Methods of Richardson-Lucy-Type for Image Deblurring. *Numerical Mathematics: Theory, Methods and Applications*, 6 (1), 262-275.

[2] Lang, L., Xu, Y. (2003). Adaptive Landweber method to deblur images, *IEEE Signal Proc. Lett.*, 10 (5), 129-132.

[3] Subashini, P. (2011). Image Deblurring Using Back Propagation Neural Network. World of Computer Science and Information Technology Journal (WCSIT), 1 (6) 277-282. http://wcsit.org/media/pub/2011/July/Image%20Deblurring%20Using%20Back%20Propagation%20Neural%20Network.pdf (accessed on 04.06.2019)

[4] Battiato, S., Giuffrida E., Rundo, F. (2008). A Cellular Neural Network for Zooming Digital Colour Images. *2008 Digest of Technical Papers - International Conference on Consumer Electronics*, Honolulu, USA, Jan 9-13, 2008, pp. 1-2.

[5] Wang, D., Dillon, T., Chang, E. (2002). Pattern learning based image restoration using neural networks. *Proceedings of the 2002 International Joint Conference on Neural Network,* Honolulu, USA, 2002, pp. 1481-1486.

[6] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR, abs/1409.1556*.

[7] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition,* Boston, 2016, pp. 770-778.

[8] MATLAB 2016b and Image Processing Toolbox, The MathWorks, Natick, 2016.

[9] Sun, J., Cao, W., Xu, Z., Ponce J. (2015) Learning a convolutional neural network for non-uniform motion blur removal. *IEEE Conference on Computer Vision and Pattern Recognition,* Boston, 2015, pp. 769-777.

[10] Dash, R., Sa, P. K., Majhi, B. (2009). RBFN based motion blur parameter estimation. *Proc. IEEE International Conference on Advanced Computer Control*, Singapore, 2009, pp. 327-331.

[11] Krahmer, F., Lin, Y., McAdoo, B., Ott, K., Wang, J., Widemann, D., Wohlberg, B. (2006). Blind image deconvolution: Motion blur estimation. *Tech Rep.*, Univ. of Minnesota.

[12] Yoshida, Y. S., Horiike, K., Fujita, K. (1993). Parameter estimation of uniform image blur using dct. *IEICE Trans. Fundamentals*, E76 (7), 1154-1157.

[13] Levin, A. (2007). Blind motion deblurring using image statistics, *Proc. Adv. Neural Inf. Process. Syst.*, 19, 841.

[14] Lokhande, R., Arya, K. V., Gupta, P. (2006). Identification of parameters and restoration of motion blurred images. *Proceedings of the ACM Symposium on Applied Computing*, 9, 301-305.

[15] Schuler, C.J., Hirsch, M., Harmeling, S., Schölkopf, B. (2016). Learning to Deblur. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38, 1439-1451.

[16] Al-amri, S.S., Ali, A.S. (2011). Restoration and Deblured Motion Blurred Images.

[17] Lucas, A., Iliadis, M., Molina, R., & Katsaggelos, A.K. (2018). Using Deep Neural Networks for Inverse Problems in Imaging: Beyond Analytical Methods. *IEEE Signal Processing Magazine*, 35, 20-36.

[18] Eom, H., Choi, H. (2018). Alpha-Pooling for Convolutional Neural Networks. *CoRR*, abs/1811.03436.

[19] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R. (2014). Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15, 1929-1958. http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf (accessed on 05.06.2019)

[20] Ba, J., Kiros, R., Hinton, G.E. (2016). Layer Normalization. *CoRR*, abs/1607.06450.

[21] Ulyanov, D., Vedaldi, A., Lempitsky, V.S. (2016). Instance Normalization: The Missing Ingredient for Fast Stylization. *CoRR*, abs/1607.08022.

[22] Wu, Y., He, K. (2018). Group Normalization. *ECCV*, 3-19.

[23] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *International Conference on Machine Learning*, Lille, France, July 6-11, 2015, pp. 448–456.

[24] Hardt, M., Recht, B., & Singer, Y. (2016). Train faster, generalize better: Stability of stochastic gradient descent. *International Conference on Machine Learning*, NY, USA, June 19-24, 2016, pp. 1225-1234

[25] Bengio, Y. (2011). Deep Learning of Representations for Unsupervised and Transfer Learning. *ICML Unsupervised and Transfer Learning*.

[26] Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *Neural Networks: Tricks of the Trade.* Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg

[27] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks : the official journal of the International Neural Network Society*, 12 (1), 145-151. https://www.ncbi.nlm.nih.gov/pubmed/12662723 (accessed on 05.06.2019)

[28] Masters, D., & Luschi, C. (2018). Revisiting Small Batch Training for Deep Neural Networks. *CoRR*, abs/1804.07612.

[29] Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, Florida, June 20-25, 2009, pp. 248-255.

[30] Witten, D.M. (2017). An Introduction to Statistical Learning: With Applications in R. New York: Springer-Verlag.

[31] Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., & Torralba, A. (2018). Places: A 10 Million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 40*, 1452-1464.

[32] Krizhevsky A. (2009). Learning multiple layers of features from tiny images. Master's thesis. Toronto: University of Toronto, Department of Computer Science.

[33] Veit, A., Wilber, M.J., Belongie, S.J. (2016). Residual Networks Behave Like Ensembles of Relatively Shallow Networks. *Proceedings of the 30th International*

Conference on Neural Information Processing Systems, Barcelona, Spain, December 05-10, 2016, pp. 550-558.

[34] Vedaldi, A., Lenc, K. (2015). MatConvNet: Convolutional Neural Networks for MATLAB. *ACM Multimedia*.

[35] Zheng, Y., Yang, C., Merkulov, A.G. (2018). Breast cancer screening using convolutional neural network and follow-up digital mammography. *Commercial + Scientific Sensing and Imaging*.

[36] Smith, S.L., Kindermans, P., Ying, C., Le, Q.V. (2018). Don't Decay the Learning Rate, Increase the Batch Size. *CoRR, abs/1711.00489*.

[37] Kupyn, O., Budzan, V., Mykhailych, M., Mishkin, D., Matas, J. (2018). DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, Utah, June 18-23, 2018, 8183-8192.

http-1: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/ (accessed on: 23.05.2019)

http-2: http://caffe.berkeleyvision.org/ (accessed on: 11.04.2019)

http-3: https://en.wikipedia.org/wiki/Motion_blur (accessed on: 23.05.2019)

http-4: https://forums.mtbr.com/photography-mountain-bikers/panning-shots-motion-blur-740285.html (accessed on: 23.05.2019)

http-5: https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/ (accessed on: 23.05.2019)

http-6: https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f (accessed on: 23.05.2019)

http-7: https://machinelearningmastery.com/why-training-a-neural-network-is-hard/ (accessed on: 23.05.2019)

http-8: https://www.cs.toronto.edu/~kriz/cifar.html (accessed on: 23.05.2019)

http-9: http://yann.lecun.com/exdb/mnist/ (accessed on: 23.05.2019)

http-10: http://cs231n.stanford.edu/tiny-imagenet-200.zip (accessed on: 22.06.2019)

http-11: https://www.robots.ox.ac.uk/~vgg/ (accessed on: 22.06.2019)

http-12: http://www.image-net.org/challenges/LSVRC/2014/index (accessed on: 22.06.2019)

http-13: https://keras.io/ (accessed on: 22.06.2019)

http-14: https://www.amazon.com/Nvidia-Tesla-GDDR5-Cores-Graphic/dp/B00Q7O7PQA (accessed on: 22.06.2019)

http-15: https://colab.research.google.com/notebooks/welcome.ipynb (accessed on: 22.06.2019)

http-16: http://www.vision.caltech.edu/Image_Datasets/Caltech101/ (accessed on: 22.06.2019)