ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

Mehmet SARIGÜL

A NEW DEEP LEARNING APPROACH: DIFFERENTIAL CONVOLUTIONAL NEURAL NETWORK

DEPARTMENT OF COMPUTER ENGINEERING

ADANA-2019

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

A NEW DEEP LEARNING APPROACH: DIFFERENTIAL CONVOLUTIONAL NEURAL NETWORK

Mehmet SARIGÜL

PhD THESIS

DEPARTMENT OF COMPUTER ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of degree of the Doctor of Philosophy by the board of jury on 26/04/2019

…….	……..	…….
Prof. Dr. Mutlu AVCI	Prof. Dr. Vasif NABİYEV	Prof. Dr. Selma Ayşe ÖZEL
SUPERVISOR	MEMBER	MEMBER
…….	…….	
Assoc. Prof. Dr. Mustafa ORAL	Assoc. Prof. Dr. Serdar YILDIRIM	
MEMBER	MEMBER	

This PhD Thesis is written at the Department of Institute of Natural And Applied Sciences of Çukurova University.
Registration Number:

Prof. Dr. Mustafa GÖK
Director
Institute of Natural and Applied Sciences

ABSTRACT

PhD THESIS

A NEW DEEP LEARNING APPROACH: DIFFERENTIAL CONVOLUTIONAL NEURAL NETWORK

Mehmet SARIGÜL

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING

Supervisor : Prof. Dr. Mutlu AVCI
Year: 2019, Pages: 97
Jury : Prof. Dr. Mutlu AVCI
: Prof. Dr. Vasif NABİYEV
: Prof. Dr. Selma Ayşe ÖZEL
: Assoc. Prof. Dr. Mustafa ORAL
: Assoc. Prof. Dr. Serdar YILDIRIM

Deep learning structures have achieved unprecedented success rates in many scientific research areas. The well-known deep structure, convolutional neural network, is commonly used in pattern recognition studies. Convolutional neural network structures are composed of a feature extractor consisting of convolution and pooling layers and a fully connected network used as a classifier. In the first study of the thesis, GCNN, a strong kernel-based classifier, was adapted to CNN structure to increase the classification performance. This adaptation led a relative performance increase up to 44.45%, 39.69% and 43.57% for precision, recall, and F1-score, respectively. Although this adaptation yielded a significant increase in performance, it was observed that the convolutional part was weak in terms of representation. Therefore, the idea of developing a convolution technique with higher learning performance has emerged. A novel convolution technique named as Differential Convolution which considers directional changes among a pixel and its neighbors is proposed. Deep structures applying Differential Convolution are named as Differential Convolutional Neural Networks. These structures made a relative performance boost up to 55.29%, 58.43%, 41.75% and 56.43% for accuracy, precision, recall, and F1-score, respectively.

Key Words: Deep learning, convolutional neural network, convolution techniques, general regression neural network, image classification, pattern recognition, artificial intelligence, machine learning.

ÖZ

DOKTORA TEZİ

YENİ BİR DERİN ÖĞRENME YAKLAŞIMI: FARKSAL KONVOLÜSYONEL SİNİR AĞI

Mehmet SARIGÜL

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Prof. Dr. Mutlu AVCI
Yıl: 2019, Sayfa: 97
Jüri : Prof. Dr. Mutlu AVCI
: Prof. Dr. Vasif NABİYEV
: Prof. Dr. Selma Ayşe ÖZEL
: Doç. Dr. Mustafa ORAL
: Doç. Dr. Serdar YILDIRIM

Derin öğrenme yapıları birçok bilimsel çalışma alanında önceden görülmemiş başarı oranları yakalamıştır. En çok bilinen derin öğrenme yapısı olan konvolüsyonel sinir ağı genel olarak örüntü tanıma çalışmalarında kullanılmaktadır. Konvolüsyonel sinir ağ yapıları konvolüsyon ve havuzlama katmanları içeren bir özellik çıkarıcı yapısı ve tam bağlı bir sınıflayıcı ağ yapısından oluşur. Bu tezin ilk çalışmasında, güçlü bir çekirdek tabanlı sınıflayıcı olan GCNN, sınıflandırma performansını arttırmak için CNN yapısına adapte edilmiştir. Bu adaptasyon kesinlik, duyarlılık ve F1-skor değerlerinde sırası ile %44.45, %39.69 ve %43.57 değerlerine kadar bağıl bir performans artışı sağlamıştır. Bu adaptasyon performansı önemli ölçüde arttırmasına karşın, konvolüsyonel kısmın temsil yeteneği anlamında zayıf kaldığı gözlemlenmiştir. Bu nedenle, daha yüksek bir öğrenme performansına sahip bir konvolüsyon tekniği geliştirme fikri ortaya çıkmıştır. Yeni bir konvolüsyon tekniği olan, bir piksel ve komşuları arasındaki yönlü değişim farklarını ele alan farksal konvolüsyon önerilmiştir. Farksal konvolüsyon uygulayan derin öğrenme yapıları farksal konvolüsyonel sinir ağları olarak adlandırılmıştır. Bu yapılar doğruluk, kesinlik, duyarlılık ve F1-skor değerlerinde sırası ile 55.29%, 58.43%, 41.75% ve 56.43% değerlerine kadar bağıl bir performans artışı sağlamıştır.

Anahtar Kelimeler: Derin öğrenme, konvolüsyonel sinir ağı, konvolüsyon teknikleri, genel regresyon sinir ağı, resim sınıflama, örüntü tanıma, yapay zeka, makine öğrenmesi.

EXPANDED ABSTRACT

Nowadays, deep convolutional neural network (DCNN) structures are used extensively in many areas of science. The main reason behind this popularity and success is the DCNN structure which combines the feature extractor and classifier in a single learning methodology. The feature extractor part of these structures generally consists of a number of successive convolution and pooling layer pairs, while classifier part is mostly a fully connected neural network structure. In this thesis, two novel convolutional neural network structures are proposed. The first structure is based on the Generalized Classifier Neural Network (GCNN) adaptation over the CNN structure. GCNN is a strong kernel-based classifier structure and its adaptation requires an improvement over the traditional back-propagation algorithm. Therefore, an altered training algorithm based on back-propagation is also proposed. Although this adaptation significantly increased the classification performance, it also raised awareness on the need for a method to increase the representativeness and training quality of the convolutional part. A novel convolution technique named as Differential Convolution is proposed for convolutional structures to provide these capabilities. This technique extracts the activation differences between the neighbors and allows the signal changes on the feature maps to be transferred to the next layer. In addition, each pixel is also provided an error fed through neighborhood pixels, allowing the training to be more reliable. Convolutional structures applying differential convolution are named as Differential Convolutional Neural Networks. Both proposed structures achieved higher classification performance values in all measures for all experiments.

The classifier part of the DCNN structures usually consists of a fully connected network. The main disadvantage of this structure is to decide an effective classifier structure. There are many parameters to be decided such as the number of the hidden layers, the number of the neurons in each layer. Also, an

effective activation function for each layer must be decided for a fully connected network. In the literature different classifying methods such as SVM and kNN were adapted as a classifier part of CNN to overcome these disadvantages. In this thesis, a novel convolutional neural network structure named as Deep Convolutional General Classifier Neural Network (DC-GCNN) and its training algorithm is proposed. DC-GCNN adapts a high-performance kernel-based classifier, Generalized Classifier Neural Network, to convolutional layers. A learning algorithm based on the error back-propagation method is developed and proposed for DC-GCNN. DC-GCNN solves the effective classifier decision problem with a fixed classifier structure and by the aid of the kernel-based classifier, it also provides much higher performance values.

Performance of DC-GCNN is compared with that of DCNN on the 10 different datasets. In the evaluation process, 5-fold cross-validation was used in the experiments. Therefore, average performance values are given to present the efficiency of the results. Three standard performance measures, precision, recall, and F1-score, are used to evaluate the performance of the proposed network structures. DC-GCNN provided a relative performance boost up to 44.45% for precision, 39.69% for recall and 43.57% for F1-score. These results show the efficiency of the proposed structure. Although this adaptation resulted in a significant rise in classification accuracy, it was realized that the convolutional part limits the overall performance. Since the extracted attributes are crucial during classification, the most suitable attributes mean the most effective classification. All attributes are fed by convolution part to the classifier. Therefore, the idea of developing a convolution technique with higher learning performance has emerged.

The most important feature that distinguishes DCNN structures from other neural network structures is the inclusion of the convolutional layers. These layers apply convolution operation to extract features from large-scale input data. Each convolutional layer contains a number of filters, also called kernels, slid over the input image to generate a number of feature maps. These generated feature maps

can be scaled down with pooling operation and then passed through another convolution process. When the dimensions of the feature maps are reduced sufficiently, they are converted into a vector and entered into the classifier structure. There are two important features that separate a convolutional layer from a fully connected neural network layer. The first one is the weight-sharing mechanism. A convolutional filter utilizes the same weight values within the whole image. This provides the ability to represent the same feature on the entire image. The second important feature is the local receptive field. The local receptive field of a neuron is the particular area which the neuron is connected in the previous layer. This area is in equal size with the corresponding convolution filter. These two features considerably reduce the number of parameters that need to be trained and significantly increase network performance, especially on pattern recognition.

The success of the convolutional neural network structures led to different convolution techniques to be proposed. In this thesis, a novel convolution technique named as Differential Convolution and its updated error back-propagation algorithm are suggested. The aim of the proposed technique is transferring feature maps containing directional activation changes to the next layer. This implementation takes into consideration that the idea of how convolved features change on the feature map. In a sense, this process is the adaptation of the mathematical differentiation operation into the convolutional process. Proposed improved back-propagation algorithm also considers neighborhood activation errors. This property raises the classification performance without changing the number of filters. In this thesis, deep learning structures utilizing the differential convolution operation are named as Differential Convolutional Neural Networks.

Four different experiment sets were carried out to observe the performance and adaptability of the differential convolution. In the first experiment set, differential convolution utilized CNN structure made a relative performance boost up to 55.29%, 58.43%, 41.75% and 56.43% for the accuracy, precision, recall, and F1 score, respectively. In the second experiment set differential convolution

adaptation raised the top1 and top5 test accuracy of AlexNet up to 5.30% and 4.75% on ImageNet dataset, relatively. In the third experiment set differential convolution utilized model outperformed all compared convolutional structures. In these experiment sets, it was observed that the differential convolution technique outperformed with respect to traditional convolution and other compared improved convolution techniques. Differential CNN had better top1 and top5 performances on CIFAR-10 and CIFAR-100 datasets with 4.37% and 2.74% more accuracy than the existing highest performing model. Relative performance increases were 7.72% and 4.09% for top1 and top5 accuracy values, respectively. In the fourth experiment set differential convolution adaptation increased the top1 accuracies of the NIN and VGGNet models up to 4.12% and 5.01%, respectively. The relative increase in the top1 accuracies were 6.01% and 7.15% for these models. These results prove the effectiveness and adaptability of the proposed technique.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Prof. Dr. Mutlu AVCI for his supervision, guidance, encouragements, patience, motivation, useful suggestions and valuable time for this work.

I would like to thank members of the PhD thesis jury, Prof. Dr. Vasif NABIYEV, Prof. Dr. Selma Ayşe ÖZEL, Assoc. Prof. Dr. Mustafa ORAL and Assoc. Prof. Dr. Serdar YILDIRIM for their suggestions and corrections.

I would like to express my sincere appreciation to TUBİTAK for their financial support in the process of my PhD education.

Special thanks to Asst. Prof. Buse Melis ÖZYILDIRIM for her support, patience, and motivation.

Finally, I would like to thank my family supported me throughout the entire process, both by their love and encouragements.

CONTENTS **PAGE**

LIST OF TABLES **PAGE**

LIST OF FIGURES **PAGE**

LIST OF ALGORITHMS **PAGE**

LIST OF ABBREVIATIONS

ANN	: Artificial Neural Network
CNN	: Convolutional Neural Network
DC-GCNN	: Deep Convolutional Generalized Classifier Neural Network
DCNN	: Deep Convolutional Neural Network
GCNN	: Generalized Classifier Neural Network
GPU	: Graphics Processing Unit
GRNN	: Generalized Regression Neural Network
ILSVRC	: ImageNet Large Scale Visual Recognition Competition
kNN	: k-Nearest Neighbours
LGCNN	: Logarithmic Generalized Classifier Neural Network
MLP	: Multilayer Perceptron
MSE	: Mean Squared Error
NCA	: Neighborhood Component Analysis
NIN	: Network-in-Network
OGCNN	: One Pass Generalized Classifier Neural Network
PNN	: Probabilistic Neural Networks
RBF	: Radial Basis Function
RNN	: Recurrent Neural Networks
SVM	: Support Vector Machines

1. INTRODUCTION

In the last decade, deep learning methods and structures have been used popularly in all areas of artificial intelligence with very high success. This success has been increasing the number of studies related to deep learning method. Although a huge amount of studies is being done on deep learning, existing large and complex models still require new training methods and techniques to reach higher performance values by realizing more effective training process.

In this thesis, two different types of contributions are made on the deep learning methodology. First of all, the classifier part of the CNN is taken into consideration. A high-performance kernel-based classifier named as Generalized Classifier Neural Network (GCNN) is adapted to CNN topology. This new hybrid structure is introduced as Deep Convolutional Generalized Classifier Neural Network (DC-GCNN). A training algorithm based on error back-propagation is also developed for this structure. Although DC-GCNN improves the efficiency of the CNN and provides higher classification rates, all improvements are limited due to the convolutional part of the neural network. It is noticed that revolutionary improvements must be realized in the convolutional part. According to this fact, a novel convolution technique named as Differential Convolution and its learning algorithm named accumulative backpropagation algorithm are proposed. This novel technique is designed as an adaptable module for readily adaptation to all CNN models. The neural network structures utilizing this technique are named as the Differential CNNs. It is observed that these networks are able to make higher depth inferences with a smaller number of layers and achieve higher performance rates.

In this section, detailed information about deep learning methodology and up-to-date deep learning research topics are given. Afterward, information about the proposed structures and their contribution to existing knowledge are introduced.

1.1. General Overview of the Artificial Neural Network

The idea of an artificial neural network was inspired by the interaction of neurons within the human brain (McCulloch and Pitts, 1943). Biological neurons receive signals with dendrites and transmit the signal from the terminal called the axon when the signal level exceeds a certain threshold. This outgoing signal is transmitted to other neurons. An artificial neuron, can be seen in Figure 1.1, is the mathematical model approach for the biological neuron. Each neuron generates an activation value depending on its input values and an activation function (Rosenblatt, 1958).

Neural network structures consist of layers and each of these layers generally contains a predefined number of neurons. These layers are input layer, output layer, and the other layers between these two layers are the hidden layers. The input layer receives data directly from the environment, while the other layers take the output of the previous layers as input. There are weight values on the links connecting the neurons. In a traditional neural network structure, each neuron in a hidden layer has connections with all neurons in the previous layer and all of the neurons in the next layer. The input values of a neuron are multiplied by the connection weights and these products are summed. Each neuron also has a bias value and this bias value is added to the sum of products before passing through the activation function. The inclusion of activation functions allows non-linearity in learning. In the most general way, training an artificial neural network means adjusting the weight and bias values to give the desired output value for the given input data. The topology of an artificial neural network, number of layers, number of neurons, type of activation functions may vary according to the work to be done. Artificial neural networks with high-performance applications are extensively used for regression and classification purposes. A neural network structure with five input neurons, two hidden layers, and one output neuron can be seen in Figure 1.2. In order to train this network, input data with five attributes and a corresponding target data must be used.

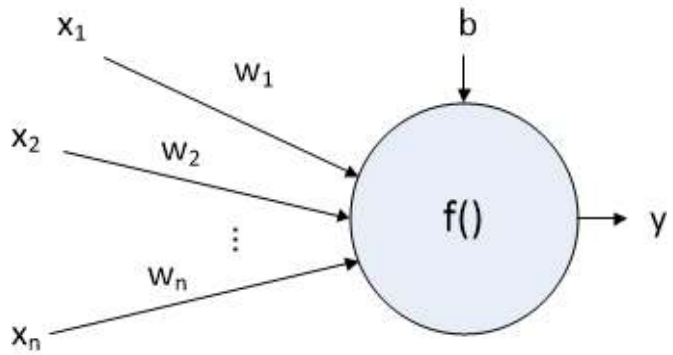

Figure 1.1 Artificial Neuron

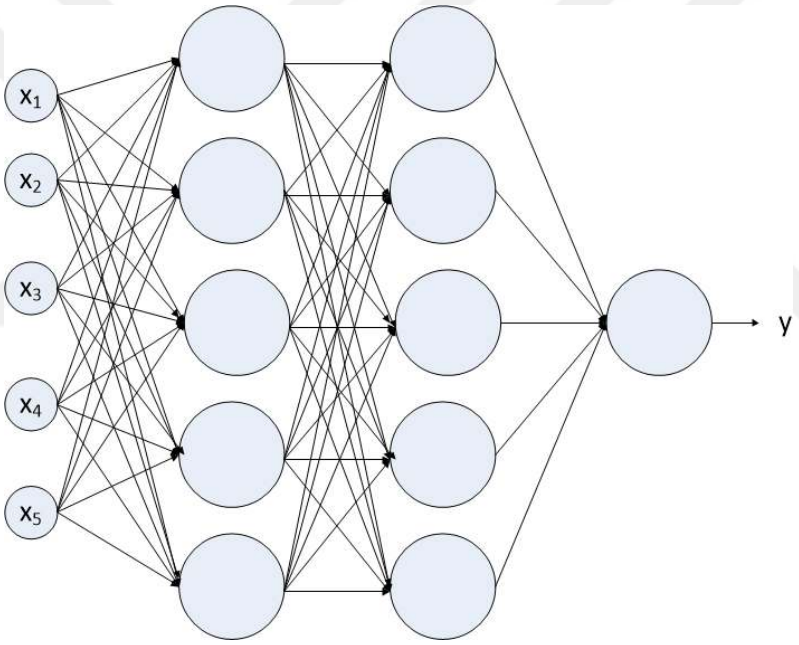

Figure 1.2 A general artificial neural network structure

Back-propagation (Rumelhart et. al.,1986), an effective gradient descent algorithm, is a breakthrough for artificial neural network training. It became possible to propagate the error among layers of multi-layered neural network by the back-propagation. In this way, parameters of each layer can be optimized by calculating corresponding error values for each parameter. After calculating the

error at the output layer, the derivative of the square error function is calculated and the calculated gradient values are transmitted backward. Errors collected on output neurons are propagated backward after multiplying them with the derivative of the activation function of the corresponding neurons. During the error back-propagation process, the errors are multiplied with the weight values on the corresponding connections; therefore, the only amount of the error caused by each neuron is taken into account. In each back-propagation step, all parameters of the network are optimized at a certain rate by using the computed individual gradient values. This rate is called the learning rate. It has been observed that a constant learning rate may limit performance in some cases. Therefore, many learning rate optimization methods have been developed. One of these methods is to decrease the learning rate over time (Zeiler, 2012). In some applications, a different learning rate is defined for each parameter. While the learning coefficient is kept small for continuously changing parameters, a higher learning coefficient can be used for the rarely changing parameters. This allows each part of the artificial neural network to learn at the same level (Tieleman and Hinton, 2012). In addition, methods for accelerating learning called momentum techniques, deciding how much of the previous knowledge must be forgotten in each step, have been developed (Nesterov, 1983; Kingma and Ba, 2014; Dozat, 2016). Recently, the increase in computing power has enabled the training of much larger network structures. The development of graphic cards increased parallel processing power, which allowed much faster neural network training. Nowadays, huge neural network structures containing millions of parameters can be trained with a single GPU. These developments have increased the popularity of neural networks.

Convolutional neural networks which are based on animals' visual cortex model suggested by Lecun in 1998 (Lecun et al., 1998). One of the biggest differences of this network from other fully connected network structures is the inclusion of the convolution operation. Convolution operation is carried out by additional convolutional layers. In these layers, a neuron is connected to neurons in

a much smaller area, called the receptive field of the neuron, rather than connecting with all neurons in the previous layer. Another important novelty of the convolutional layers is the weight sharing mechanism. This mechanism allows the same pattern to be searched in the entire input image. While these two features increase the power of representation and the performance of the network, the number of parameters needed to be trained are also decreased. The hardware limitations at the start-up of the convolutional layer addition did not allow this structure to be utilized for larger data sets. The development of GPUs and the increase in the number of parallel synchronous operations have revealed the opinion that convolutional neural networks structures can be trained on GPUs (Steinkraus et al., 2005). Finally, convolutional neural network structures implemented on GPU gave impressive results (Ciresan et al., 2011). They exhibited the best performance values in the literature on very popular multi-class image datasets such as MNIST and CIFAR (Ciresan et al., 2012). These high-performance rates led to an increase in the popularity of convolutional neural networks. This popularity allowed the development of many techniques not only for convolution but also for each subarea of the artificial neural networks.

1.2. What should be improved

A general deep convolutional neural network structure contains three different types of layers which are convolutional layer, pooling layer, and fully connected layer. Convolutional layers and pooling layers are located in the feature extractor, while the fully connected network serves as a classifier. Convolutional layers are used to perform feature extraction over input data, while the task of the pooling layer is to reduce the size of the data with minimal data loss. New convolution and pooling techniques are required to increase the efficiency of the deep structures. In addition to improvements that can raise the performance of a fully connected classifier, different types of classifiers can be optimized for the deep structure. Different classifier methodologies such as kNN and SVM have

already utilized to achieve higher classification performance. Training of these huge structures can be problematic. The trained parameters can reach very large values so that the network can diverge from the training. Therefore, studies dealing with optimization and regularization of the training have been carried out. Additionally, different loss functions and activation functions must be utilized to increase the performance.

1.3. The Aims and Objectives

Deep convolutional neural network structures are composed of two parts which are feature extractor and the classifier. Feature extractor part of the CNN is composed of convolutional layers and pooling layers, while the classifier part is mostly a neural network structure composed of fully connected layers. It is needed to determine the correct number of layers and the correct number of neurons for each layer for a fully connected classifier. Also, the loss function and activation function for each layer must be decided. Additionally, the fully connected classifier can also be problematic in performance. Therefore, it is intended to eliminate these problems and to achieve higher performance values by adapting GCNN, a high-performance kernel-based classifier, instead of this fully connected neural network structure. This novel structure is named as Deep Convolutional Generalized Classifier Neural Network. A training algorithm based on error back-propagation is suggested for the training of DC-GCNN. This adaptation provided a significant increase in classification performance. During this study, it was noticed that the traditional convolution structure may struggle to learn with the error produced by the GCNN classifier. Therefore, the idea of developing a new and more powerful convolution technique has emerged. It is required to develop methods accelerating the training, increasing the representation power of the convolution and improving the performance of the neural network structure. A novel convolution technique named as Differential Convolution is proposed to achieve these goals. Unlike the traditional method, this technique produces additional feature maps that include

neighborhood activation changes. During the learning process, errors fed from the neighbor neurons are also taken into consideration. Structures utilizing differential convolution are named as Differential Convolutional Neural Networks. Differential convolution required an upgrade on the standard back-propagation algorithm. A novel training algorithm named as accumulative back-propagation is proposed for the training of Differential CNN structures. This method has resulted in a significant increase in performance compared to that of the traditional method.

1.4. Contribution to Existing Knowledge

In this thesis, two novel deep convolutional neural network structures are proposed together with their training algorithms. The proposed first neural network structure, Deep Convolutional General Classifier Neural Network, is designed with a view of increasing the classification performance of the DCNN. In this structure, GCNN structure is adapted instead of a fully connected neural network structure. The proposed approach includes the statistical variance calculation method of One-pass Generalized Classifier Neural Network (OGCNN) and iteratively trained convolutional layers. Since GCNN structure composed of a fixed number of layers, DC-GCNN also has this structural advantage. This advantage prevents the structure decision problem of a fully connected classifier and provides kernel-based high classification performance. This adaptation has also improved the learning algorithm. A new learning algorithm based on error back-propagation for the training of the convolutional part is designed and implemented for DC-GCNN structure.

Recommended second artificial neural network structure, Differential Convolutional Neural Network, contains a novel convolutional technique named as differential convolution which let the convolutional layer generate four additional feature maps for each convolution filter. Proposed convolution technique improves the standard convolution by considering directional changes among a pixel and its neighbors. Differential convolution extracts pattern orientation of a pixel and its

neighbors by calculating the additional change amounts. As in mathematical differentiation, change amount calculations take sequential changes into consideration by computing the difference among the activations of the pixels. Since images can be defined as a form of a superposition of waves with different frequencies, differential convolution generates a new feature map representing signal changes. These new feature maps lead detection of the big differences in the patterns which are valuable information for classification problems. Four pre-defined constant filters are realized to perform this operation. Each of these filters is used for calculating the differences in one of four directions. The proposed technique applies these filters on a single feature map generated by the traditional convolution. This operation is named as Differential Convolution. The efficiency of the proposed technique is provided by preserving neighboring activation difference information. Additionally, neighboring activation difference is frequently used information in image processing especially for determining edges and corners. Although mathematical operations of the proposed method are simple, it requires updates on the traditional error back-propagation method. Hence, a new error back-propagation method for differential convolution adapted structures named as accumulative error back-propagation is suggested. In the accumulative method, each neuron receives an error fed over the neighbor activations during the back-propagation.

Four different experiment sets were carried out to demonstrate the effectivity of the proposed technique. In the first experiment set differential convolution was adapted to a CNN structure and the performance increase was observed on ten different image datasets. In the second experiment, differential convolution was adapted to popular deep structure, AlexNet, and the performance increase was measured on ImageNet dataset. ImageNet is a very popular large-scaled dataset containing 1.3 million images divided into 1000 different classes. In the third experiment set, six different CNN structures were tested on CIFAR-10 and CIFAR-100 datasets. Three of these structures are different traditional CNN

structures and the others are Differential CNN, Tiled CNN, and Dilated CNN. Differential CNN performed higher performance values than all of the compared models on validation data. In the fourth experiment set, differential convolution was adapted to NIN and VGGNet models and performance increases of the models were measured on CIFAR-10 and CIFAR-100 datasets. In addition to improvements in the performance, it had also been shown that the decrease in the error value of the proposed technique is higher than that of the other compared methods at the same time period. These results show the importance and effectiveness of the proposed convolution technique.

1.5. Main Contributions of the Research

The literature review and evaluation of the traditional deep convolutional methods produced the following findings:

1. Using different classifier structures within the deep learning structure can increase the classification performance of the model.
2. Classifier structure used in the deep learning model changes the whole training algorithm with considerable effect on the success of the training.
3. Although convolution is a demanding process, it reduces the number of the parameters of the neural network model and provides a great success to deep learning structures. Therefore, it is very essential to develop more effective convolution techniques.
4. Improving the convolution technique with different aspects may increase performance perceptibly.
5. There is a trade-off between the number of parameters and performance of the model. The important thing is increasing the number of parameters by gaining the ability of distinct feature extraction.

6. As deep learning structures are quite large and costly to train, it is essential to develop structures and techniques that will speed up training and improve the quality of the results.

1.6. Outline of the Thesis

This thesis is organized as follows:

In Section 2, the related studies in the literature are reviewed and general information is given about these studies. Their advantages and disadvantages are discussed.

Section 3, covers materials such as structures, algorithms and training methodologies used at each step of the study. Firstly, the traditional Deep Convolutional Neural Network structure that this thesis is based on is explained in detail. Then, information about the use of the deep learning structure, training of the structure, the problems and solutions that arise in these stages are given. In the last part of this section, the datasets used for testing of the suggested neural network structures, and the evaluation functions used for measuring the performance are explained in details.

In Section 4, the proposed two different neural network structures to boost the performance of DCNN are presented in details. Structure of the networks, ideas behind the implementation of these structures, important equations and training algorithms are given in detail in this section.

In Section 5, details of all experiments, comparison of the proposed neural network structures with classical structure, performance analysis of the models, obtained classification results, the discussions on test results, and future works are given.

The last part of the thesis covers a detailed conclusion about all the studies performed in this thesis.

2. PREVIOUS WORKS

In this section, a detailed literature search is given. Firstly, information about the history and development of DC-GCNN which is the first proposed structure is given and similar approaches are explained in details. Then, the details of similar techniques to the proposed novel convolution technique are given and the advantages and disadvantages of similar techniques are discussed.

The convolutional part of the DCNN consists of convolution and pooling layer pairs that are generally followed by a fully connected classifier. Using a fully connected network structure as a classifier has several problems. The number of layers and the number of neurons for each layer for an effective network structure must be determined. In addition, effective activation and loss functions must be determined. Even if all these parameters are set correctly, the performance of the fully connected classifier may be poor. Some techniques such as DropOut or DropConnect must be utilized for the stability of the fully connected classifier. Therefore, it is necessary to adapt easy and effective classifiers to CNN structure to overcome these problems. There exist studies using different methods such as k-nearest neighbor (kNN) and support vector machines (SVM) for the classifier part of the DCNN structure (Ren et al., 2014; Tang, 2013). Ren et al. used kNN as the classification part of DCNN. The loss function of the proposed method was neighborhood component analysis (NCA), and stochastic gradient descent algorithm was used for training. The suggested structure consisted of three convolutional layers which each one is followed by a pooling layer and two fully connected layers. It was evaluated on MNIST and CIFAR-10 datasets. According to the given test results, while 1-NN reached an error value of 0.83 on MNIST dataset, the error value of 10-NN was 0.75. The accuracy of the proposed structure was 75.2% with 1-NN and 78.3% with 10-NN on CIFAR-10 dataset. Tang used SVM instead of SoftMax part of the DCNN structure. Unlike most of the studies in the literature used SVM after obtaining the high-level features, SVM was

accompanying the training process in that work. L2-SVM was used as a loss function. While the structure using the softmax function reached an error rate of 0.99% and 14%, the structure using linear SVM model reached an error rate of 0.87% and 11.9% on MNIST and CIFAR-10 dataset, respectively.

General Classifier Neural Network structure is a high-performance neural network structure based on the Nadaraya-Watson kernel (Özyıldırım and Avcı, 2013). GCNN employs RBF in the pattern layer similar to the generalized regression neural network (GRNN) and probabilistic neural network (PNN). The only parameter needed to be tuned is the variance, σ, as in other RBF-based network structures. Three different methods were suggested for tuning the efficient variance value. The first method using stochastic gradient descent approach for error back-propagation requires a long time especially for large datasets (Özyıldırım and Avcı, 2013). The second method called logarithmic learning for GCNN(LGCNN) uses the maximum likelihood estimation to decrease the value of logarithmic cost function (Özyildirim and Avcı, 2014). Although the second method is quite fast compared to the first method, it is still slow in terms of use in real-time applications. The third approach suggested was named as one pass GCNN (OGCNN). This one pass method uses the data statistics to calculate the efficient variance value. All versions of GCNN utilizing a term called diverge effect term to separate the data belonging to different classes. Test results showed that OGCNN is much faster than the other two methods, but also gives more successful results.

In this thesis, a novel deep classifier structure Deep Convolutional General Classifier Neural Network (DC-GCNN) is proposed. DC-GCNN inherits the kernel-based learning methodology of the GCNN family. DC-GCNN contains the statistical variance calculation approach of OGCNN in the classifier part and iteratively trained convolutional layers. The inclusion of the GCNN required an alteration on the training algorithm. An algorithm based on error back-propagation is also given. DC-GCNN structure provided a great performance increase in all the

tests for all measures.

The success of convolutional neural networks has led to many studies to be carried out on these structures. The high representativeness of them, which is the key point of their success, largely comes from the convolution operation. Many novel convolutional structures have been proposed in recent years. One of these structures is Tiled Convolutional Neural Network (Ngiam et al., 2010). Tiled CNN uses different filters for neurons having close receptive fields in the convolution process. This is because nearby domains will give similar results for the same filters. More feature extraction can be done without increasing the number of feature maps with tiled convolution. There is an extra parameter, k, denotes the tile size. If n filters are used, the number of feature maps to be generated will be n / k. Tiled CNN showed a performance of 73.1% on the CIFAR-10 data set. Dilated Convolutional Neural Network (Yu and Koltun, 2015) is another convolutional structure. The main idea behind this structure is to increase the filter's receptive field without increasing the number of parameters by inserting cells with the value zero values into the convolution filters. If the size of the filter is nxn and the dilation amount of the layer is k, the size of the filter's receptive area will be $n + k(n-1)$. This structure led to a performance boost in some studies (Sercu and Goel, 2016). Deconvolution (Noh et al., 2015) is another convolution technique aims to increase the size of the input image. This technique is mostly used for image segmentation and image reconstruction tasks inside auto-encoder structures. The deconvolution process produces multiple values for a single input value, unlike the convolution process that produces a single activation value from multiple values. It achieved a performance of 72.5% on Pascal VOC 2012 dataset (Everingham et al., 2010). Another successful convolutional structure, named as Network In Network (NIN) (Lin et al., 2013), suggests the usage of micro MLP structures as filters. The filters in this structure have multiple layers and the error is propagated from the last layer to the first layer between its own layers as in

ordinary back-propagation. This provided a stronger representation power for the convolutional filters. Models with NIN filters achieved higher performance values than previous methods in many data sets such as CIFAR-10, CIFAR-100 etc. Another well-known and popular convolution technique is Inception (Szegedy et al., 2015). In this method, different sized filters can be found in the same convolutional layer. There may even be a pooling process in the convolution layer. It was shown that more successful results can be obtained with fewer parameters by using Inception modules. GoogleNet (Szegedy et al, 2015) structure created with inception modules won the ILSVRC competition in 2014. ILSVRC 2012 winner AlexNet structure contains 60 million parameters, while a number of parameters in the GoogleNet is only 4 million.

The main idea behind the alternative techniques is to increase the number of obtained features from a single convolution layer. Similar to this idea, in this thesis, a novel convolution technique, differential convolution, is proposed. There are some novelties and advantages of the proposed method over the other approaches. While Tiled convolution increases the number of filters and keeping the number of feature maps constant, the number of feature maps is increased without increasing the number of filters in the Differential convolution. Dilated convolution attempts to increase the receptive field of a filter, on the contrary, differential convolution increase the efficiency of the filter without changing the receptive field size. The purpose of the deconvolution process is to increase the size of the input which is quite different from the proposed technique. Differential convolution has a different perspective than the NIN method and its implementation is much simpler. It can be used inside the inception module as traditional convolution. While Tiled CNN and Dilated CNN recommend new convolution techniques, the NIN method does not change the convolution method but the filter structure; deconvolution has a much different objective, while the inception module offers a structure where more than one technique can be used together. Therefore, the Differential CNN structure may be presented as an

alternative to conventional CNN, Tiled CNN and Dilated CNN structures. Differential CNN allows more feature extraction by using neighboring activation differences which provides many advantages in use. Differential convolution adapted CNN structures showed impressive performance result in all experiments and outperformed the alternative structures.

3. MATERIALS

3.1. Introduction

This section contains detailed information about Deep Convolutional Neural Network methodology. The structure of these networks and the techniques used in their training process are given in detail. Later in this section, an introduction about a set of popular data sets used to evaluate the performance of the proposed structures is included. After that, popular convolutional neural network structures are given. Finally, the evaluation metrics and methods that are used to evaluate and compare the quality of the classification results of the experiments are explained.

3.2. Deep Convolutional Neural Networks

Convolutional neural networks (LeCun et al, 1998) have become one of the most popular and successful approaches in the area of pattern recognition. In these structures, visual features are extracted by sliding locally trained filters over the input image. The size of generated feature maps is reduced by the pooling operation and then these maps are transferred to the next convolutional layer. This process is repeated until deep features are extracted. After that, a classifier is used to make a decision over these features (Ravi et al, 2017). A deep convolutional neural network structure generally composed of convolutional layers, pooling layers and a fully-connected neural network (Schmidhuber, 2015). In this structure, convolutional and pooling layers are found in the feature extractor part, whereas the fully connected network is used as a classifier. There are many popular structures created with this order such as AlexNet (Krizhevsky et al, 2012), OverFeat (Sermanet et al, 2013), GoogLeNet (Szegedy et al, 2015). AlexNet was proposed for the classification task of ImageNet LSVRC-2010 competition images. Structure of the AlexNet is composed of five convolutional layers, three pooling layers, and three fully connected layers. First and second convolutional layers are

followed by pooling layers and local response normalization is applied. Third pooling layer is applied after the last convolutional layer. The fully connected part of the network ends with a SoftMax classifier. Moreover, dropout and data augmentation, two techniques prevent overfitting, are also utilized in AlexNet and in many other popular convolutional neural network structures. DropConnect is another novel technique used to prevent overfitting. These successful structures are also used for feature extraction tasks in many different works. The fine-tuning operation may be required in some cases when these structures are used to classify or extract features of a new dataset.

3.2.1 Convolutional Layer

The convolutional layer is the layer applies convolution operation in the deep neural network. Convolution is performed by applying the complex functions to the input image using filters (Ravi et al., 2017). Convolutional filters are fixed sized kernels. These kernels are trained locally according to the application and they are slid over the entire input image during the convolution operation and generates an activation value of each position. In this structure, each neuron is bound by a limited number of neurons in a particular area in the previous layer. This area is called the receptive field. There are numerous convolutional filters on each convolutional layer and each filter is trained to recognize a specific pattern on input images. A filter has the same weight and bias values when sliding over the image. This is called the weight-sharing mechanism. In this way, the same patterns can be scanned on the entire input image by the corresponding filter (Chandrakumar and Kathirvel, 2016; Ravi et al., 2017; Sankar et al., 2016). The number of filters of a convolution layer must be carefully determined. If few filters are used, performance may be reduced as few patterns are recognized. If a large number of filters are utilized, the number of parameters will increase and the training of the network will slow down. Another issue to be considered is the size

of the filters. Generally, large filters are used in the first layers and smaller filters are used in the other layers. Especially in the first layers, it may be necessary to use striding to effectively reduce the image size.

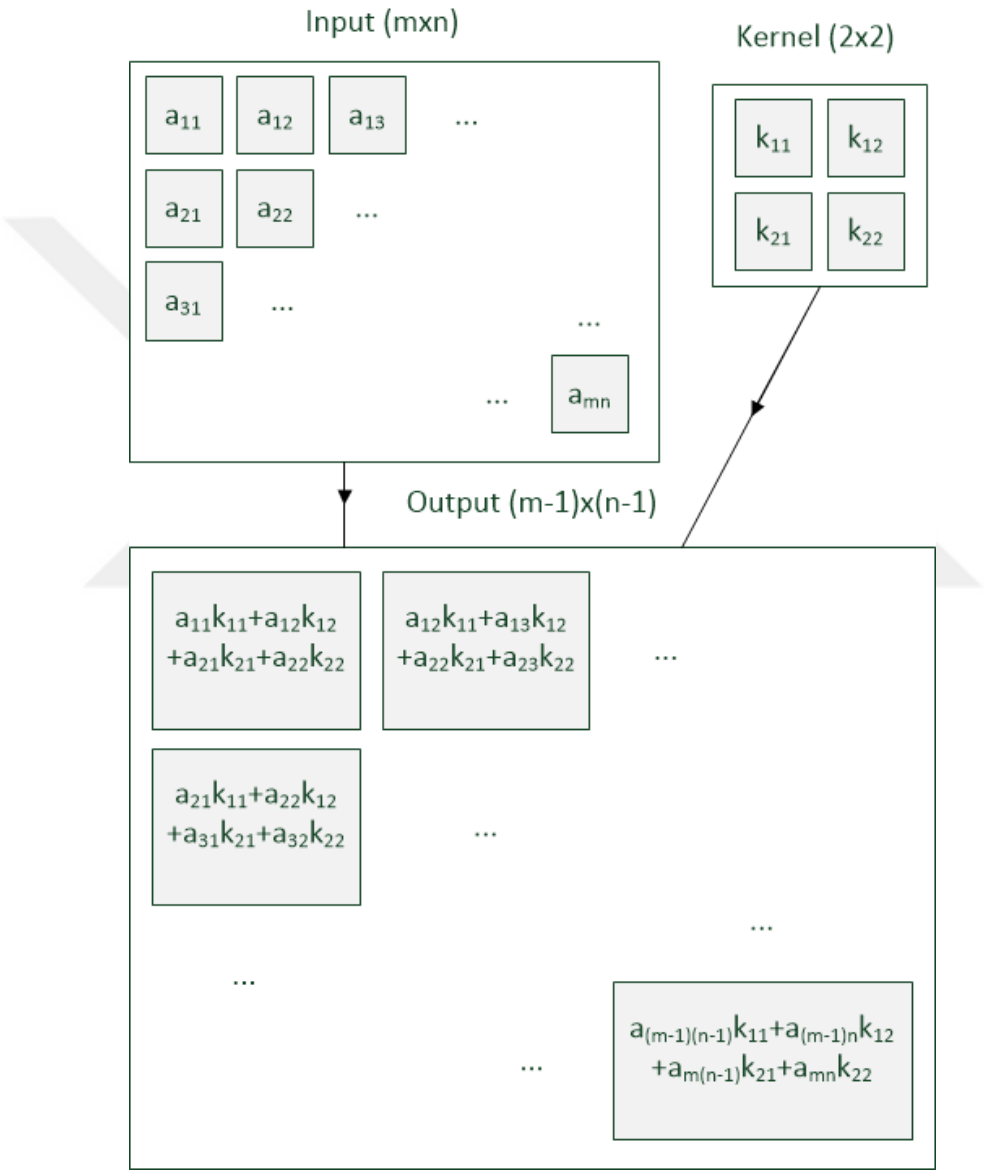

Figure 3.1 Convolution Operation

Let mxn be the size of the input, cxc be dimensions of the filter, i represent the input, w and b be shared weight and bias values, then the output of the $o_{0,0}$th neuron can be calculated as in (3.1) where f is an activation function such as Rectified Linear Unit (ReLu), sigmoid, etc. (Nielsen, 2015).

$$o_{0,0} = f(b + \sum_{t=0}^{c}\sum_{r=0}^{c} w_{t,r} i_{0+t,0+r}) \quad 3.1.)$$

When the filter is sliding over the image and the activation is calculated, sliding can be multiple pixels at each step. In this way, the size of the feature map will be reduced. Number of pixels to be shifted in each step of convolution is called the striding length. Also, the input image can be expanded with 0-valued pixels before the convolution operation. This operation is called padding. Let s be the stride length, pxp be the padding size and IxI be the input size, the size of the output image can be calculated as $(I - c + 2p)/s + 1$. ReLu, the activation function which is popular with deep learning, gives zero for negative inputs and it is linear for positive inputs (Nielsen, 2015).

3.2.2. Pooling Layer

After the feature maps are produced by the convolution process and passed through an activation function, the pooling operation is applied. The purpose of this process is to reduce the size of the feature maps with minimum data loss. Reducing the size of the feature maps allows a faster learning process with less parameter. This operation is also carried out by sliding filters over the image. But this time a filter applies a pooling operation.

The most popular pooling operations are maximum pooling, average pooling and L2 pooling (Nielsen, 2015). Maximum pooling chooses the maximum value, average pooling calculates the average, while L2 pooling takes the L2-norm

of the input values in the kernel sized input area. The pooling process, together with accelerating the training, also allows the pattern to be recognized independently of its location in the picture (Nielsen, 2015). Let IxI be the size of the input, cxc be dimensions of the pooling filter, the pixel size for each dimension can be calculated with (I/c) .

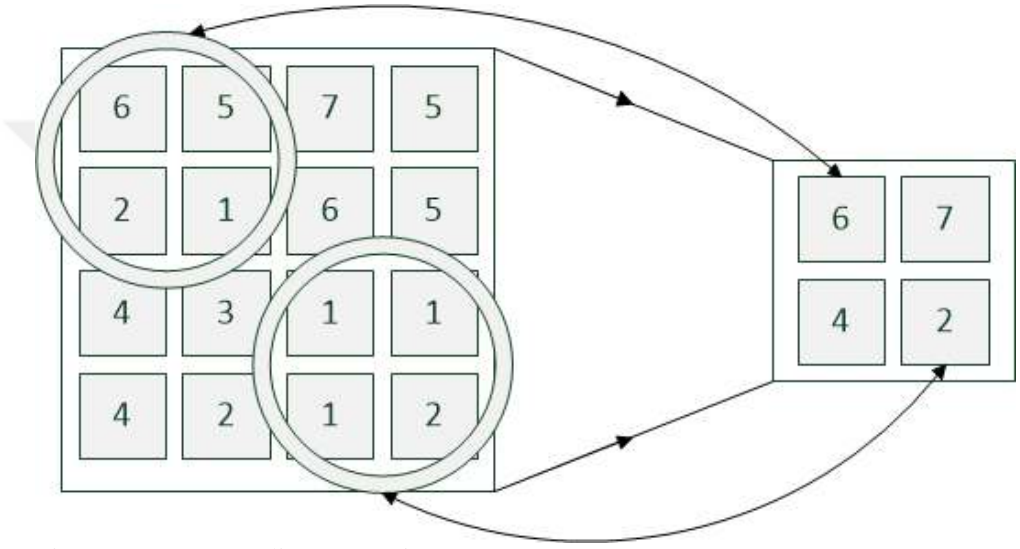

Figure 3.2 Max Pooling Operation

3.2.3. Local Response Normalization

It is reported that local response normalization raises the performance of the network and provides a better generalization (Krizhevsky et al, 2012). Let $o_{0,0}^{j}$ be the output of the jth kernel's neuron after the ReLu, the output of response normalization, $o_{0,0}^{j'}$, can be calculated as in (3.2.), where h_1, h_2, h_3 and h_4 are predetermined hyper parameters. h_2 represents number of adjacent kernels at the same spatial position and K is the number of kernels in the corresponding layer.

$$o_{0,0}^{j'} = \frac{o_{0,0}^{j}}{h_1 + h_3 \sum_{q=\max(0,j-h_2/2)}^{\min(K-1,j+h_2/2)} (o_{0,0}^{q})^{h_4}} \quad 3.2.)$$

3.2.4. Fully Connected Layer

The neurons in the last layer of the feature extractor, including the convolution and pooling layers, are converted into a one-dimensional vector. The layers that come after this process are fully connected layers. Fully connected layer neurons perform the following calculation, (3.3):

$$f_{c_1} = f(b + \sum_{q=1}^{M} w_{1,q} * o_q) \quad 3.3.)$$

where f is the activation function and M is the number of previous layer neurons aligned in a vector, o_q is the qth neuron's value, f_{c_1} is the value of the first neuron in the first hidden layer of the fully connected network and $w_{1,q}$ is the weight value between the neurons o_q and f_{c_1}. Fully connected neural network structure composed of fully connected layers is generally used as a classifier in deep learning structures. This structure is usually followed by a SoftMax output layer for classification purposes.

3.2.5. DropOut

DropOut technique is used for avoiding overfitting. In this technique, some of the hidden layer neurons are determined to be dropped out by using a dropout probability. Selected neuron activities are zeroed during the forward and backward step for a phase of the training process. As different neurons become active for each iteration, more consistent and reliable learning is provided, although the training time is extended (Krizhevsky et al, 2012).

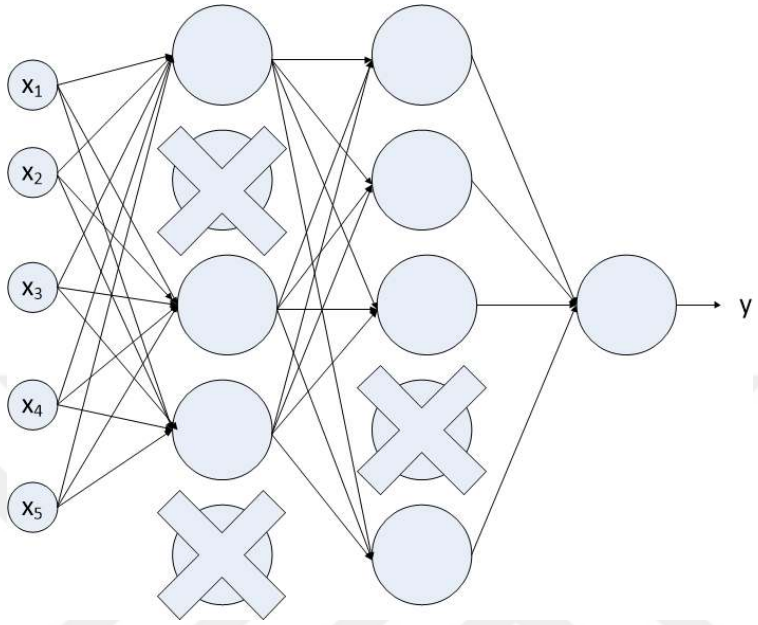

Figure 3.3 Dropout Technique

3.2.6. SoftMax

SoftMax function is the general form of logistic regression for multiple classes. It takes an input vector of N real numbers and outputs a probability distribution consisting of N probabilities. All computed probability values are in the range of (0,1) and the sum of these probability values is equal to 1. Higher input values will result in higher probability values. In the majority of the work, deep convolutional neural networks end up with a SoftMax layer for classification purposes. It is defined as in (3.4).

$$class_j = \frac{\exp(sf_j)}{\sum_q \exp(sf_q)} \qquad 3.4.)$$

where $class_j$ is the value of jth output, sf_j is the jth input value to softmax activation function and sf_q shows each neuron in SoftMax layer.

3.2.7. DropConnect

DropConnect is another technique used for preventing overfitting (Wan et al, 2013). While randomly selected neurons are deactivated during a phase of training in DropOut, the weights on randomly selected links are taken as zero in DropConnect. Links between different neurons are deactivated in each iteration. This process avoids memorizing, provides reliable training and raises the performance of the neural network. DropConnect technique is given in Figure 3.4. In the figure, 20% of the connections are removed randomly.

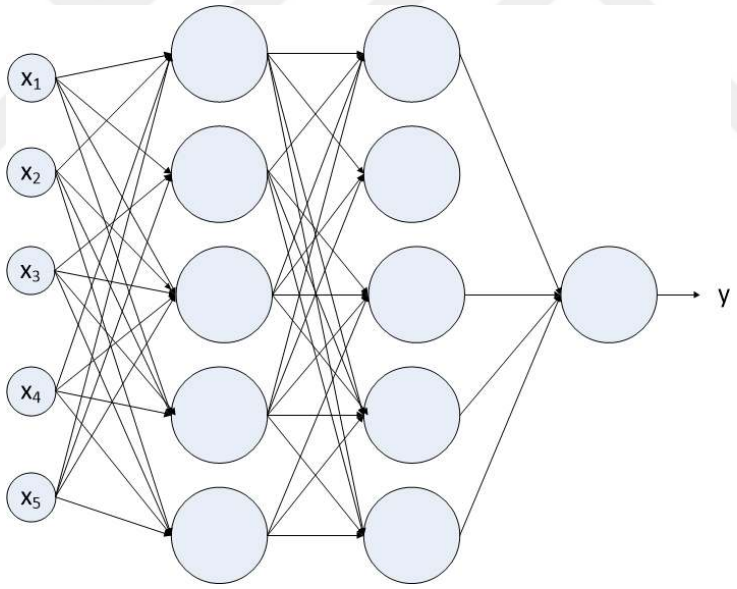

Figure 3.4 DropConnect Technique

3.2.8. Data Augmentation

Data augmentation is based on increasing the number of instances by artificially altering the data. This process can be done by making various

transformations on the picture. Flipping, rotating, cropping, altering the size, changing the lighting conditions are some of these transformations. Artificially produced data is used in network training together with other data. This prevents the network from overfitting and increases the recognition ability of the network. In addition, pattern recognition is carried out independently of environmental factors. Illumination invariant features are provided by this process.

3.2.9. Fine-Tuning

Deep neural networks are tremendous structures requiring a long time and a large number of samples for the training process. Therefore, utilizing a pre-trained neural network structure for some tasks can be more preferable. This process is named as transfer learning. There are different transfer learning approaches. Pre-trained models are trained by using large datasets, for this reason, these structures can be utilized as a feature extractor for similar problems. In this way, the pre-trained deep model is utilized for feature extraction of a new dataset and then generated features are used for classification. Therefore, only the classification operation requires a training step.

On the other hand, fine-tuning is another transfer learning approach. In this method, the pre-trained network is re-trained with the new dataset. The pre-trained weights can be used as the initial weight values for the whole network and can be retrained, or some parts of the network can be kept constant and only some parts of the network can be re-trained. There are some issues must be considered when using a pre-trained neural network. The first one is the similarity between the new dataset and the one used in the pre-training process. The other issue is the size of the new dataset. If contents of two datasets are similar, fine-tuning operation will not be preferred for the small datasets before the feature extraction, however, fine-tuning operations can be profitable for the large datasets. On the contrary, if the contents of the two datasets are different, the whole structure should be trained for

the large dataset while fine-tuning from earlier layers can be more efficient for small ones. Additionally, the learning rate is an important issue in this operation (Karpathy, 2015).

There are many popular pre-trained neural network structures, such as GoogLeNet, OverFeat, and AlexNet. These networks were trained with ImageNet datasets, contains over one million high-resolution images, for ImageNet Large Scale Visual Recognition Challenges. These networks are very popular in transfer learning area (Szegedy et al., 2015; Krizhevsky et al., 2012; Sermanet et al.,2013; Deng et al., 2009).

3.3. OGCNN

GCNN is a classifier artificial neural network structure using RBF kernel to utilize regression-based classification. GCNN is composed of five layers which are input layer, pattern layer, summation layer, normalization layer and output layer as shown in Figure 3.5. As in other RBF kernel-based classifier neural network structures such as GRNN and PNN, the only parameter should be tuned is the smoothing parameter for this structure. While GCNN utilizes gradient descent approach to adjust the smoothing parameter, LGCNN uses logarithmic learning approach and OGCNN computes the smoothing parameter by using statistical information of data (Ozyildirim and Avci, 2013; Ozyildirim and Avci, 2014; Ozyildirim and Avci, 2016). The proposed study is based on OGCNN, therefore, in this section, OGCNN is explained in detail.

GCNN receives a vector of numerical data as input. Pattern layer consists of neurons calculating the squared Euclidean distance between the input vector x and training data vector t found in the corresponding neuron as in (3.5) where p is the number of training data at the pattern layer. RBF kernel function is utilized to compute squared Euclidean distance as in (3.6) where σ is the smoothing parameter and $r(j)$ denotes the output of j^{th} neuron. Output values of the pattern

layer are the RBF kernel functions' outputs. Additionally, this layer holds N target values for each training data defined by the corresponding class. If a training data belongs to i^{th} class, i^{th} target value of this data will set to 0.9 and all other $N-1$ target values will be 0.1 as given in (3.7).

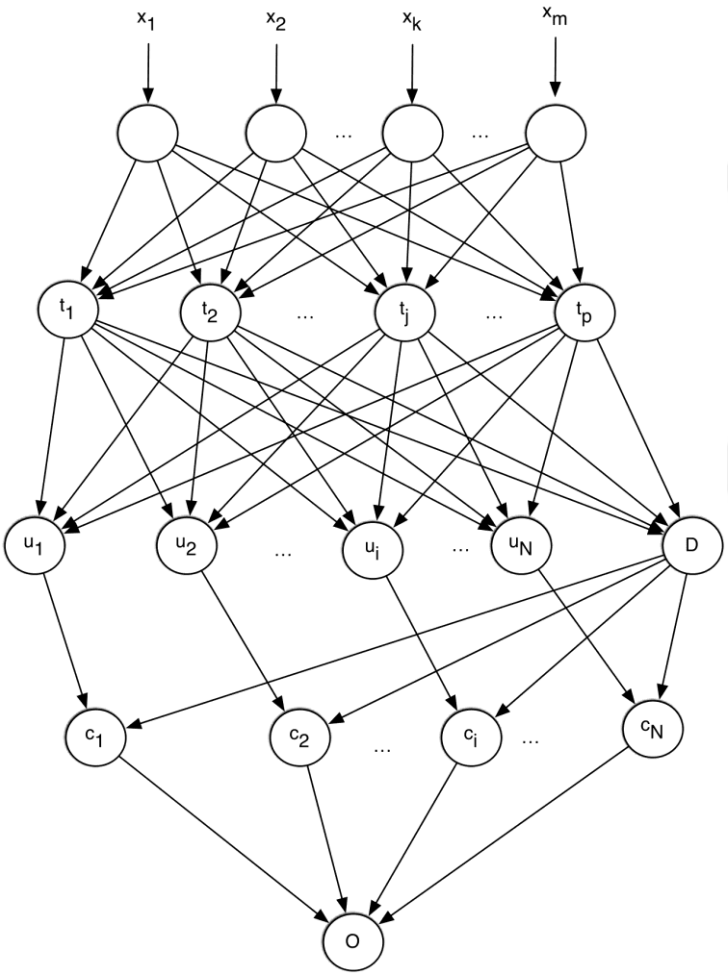

Figure 3.5: GCNN Structure

$$dist(j) = ||x - t_j||^2, 1 \leq j \leq p \tag{3.5.}$$

$$r(j) = e^{(-1*\frac{dist(j)}{2\sigma^2})}, 1 \leq j \leq p \tag{3.6.}$$

$$y(j,i) = \begin{cases} 0.9\ t_j\ belongs\ to\ i^{th}\ class \\ 0.1\ else \end{cases} \quad 1 \leq i \leq N, 1 \leq j \leq p \tag{3.7.}$$

Summation layer contains $N + 1$ neurons where N is the total number of classes and one extra neuron to calculate the denominator value. Diverge effect term is utilized to increase the distance among the classes at the summation layer is calculated as in (3.8) where $d(j,i)$ denotes the diverge effect term of j_{th} training datum i_{th} class. y_{max}, initially is set to 0.9, is the maximum value of $y(j,i)$. In each iteration, y_{max} value is set to the output value of the winner neuron. Each of the N neuron calculates the diverge effect term, separately. While the usage of the exponential form of $y(j,i) - y_{max}$ raises the effect of $y(j,i)$, it also ensures the convergence to a minimal error within the limits and avoidance of the problem of overfitting.

$$d(j,i) = e^{(y(j,i)-y_{max})} * y(j,i) \tag{3.8.}$$

After the calculation of diverge effect terms, nominator values are computed at N neurons of the summation layer as in (3.9) where u_i denotes nominator value of i^{th} neuron. There is an extra neuron calculating the denominator value at this layer. Denominator value is computed as in (3.10) where D denotes the denominator value.

$$u_i = \sum_{j=1}^{p} d(j,i) * r(j), 1 \leq i \leq N \tag{3.9.}$$

$$D = \sum_{j=1}^{p} r(j) \tag{3.10.}$$

The normalization layer consists of N number of neurons where each neuron represents one of the N classes. Each neuron divides the corresponding nominator value by the denominator value calculated at the summation layer as in (3.11) where c_i denotes the normalized output of i^{th} class.

$$c_i = \frac{u_i}{D}, 1 \leq i \leq N \tag{3.11.}$$

Winning detection mechanism is applied to determine the class of the input vector at the output layer. It is given in (3.12.) where c denotes the output vector of the normalization layer, c_{id} shows the winner neuron's value and id represents the indice of the winner class, respectively.

$$[c_{id}, id] = \max(c) \tag{3.12.}$$

Unlike GCNN, OGCNN calculates the smoothing value in a non-iterative one-step way. The most important factor for computation of smoothing parameter, ensures handling different data distributions of the dataset, is the standard deviation. OGCNN also deals with different data distributions in the dataset. OGCNN utilizes two functions to compute the smoothing parameter and defines the active function through thresholding.

Let V_i^k and M_i^k shows the standart deviation and mean of k^{th} feature of i^{th} class, P_i denotes the number of training data belonging to i^{th} class and t_l^k

represents k^{th} feature of l^{th} training data belonging to i^{th} class. First, V_i^k and M_i^k are computed as in (3.13.) and (3.14.). Then each of the features are examined separately to determine the distribution of the class. Standard deviation of the maximum valued feature can be calculated to find the limit of the corresponding class. Therefore, the limits of the i^{th} class ($Vmin_i$ and $Vmax_i$) are determined by calculating the standard deviation of minimum and maximum spread attributes. (3.15.) denotes the representative standard deviation of the i^{th} class, where $maxft_i$ is the index of the attributed whose values are most widely spread.

$$M_i^k = \frac{1}{P_i}\sum_{l=1}^{P_i} t_l^k, 1 \le i \le N, 1 \le k \le m \tag{3.13.}$$

$$V_i^k = \sqrt{\frac{1}{P_i - 1}\sum_{l=1}^{P_i}(t_l^k - M_i^k)^2} \tag{3.14.}$$

$$Vmax_i = \max(V_i), Vmin_i = min(V_i), maxft_i = argmax(V_i) \tag{3.15.}$$

OGCNN considers two different conditions for the standard deviation of a class. As the distribution of the data approaches their mean, the standard deviation approaches 0, otherwise standard deviation rises. Therefore, OGCNN checks whether the maximum standard deviation is in the range of [0,1]. If the standard deviation is in the range of [0,1], it is assumed that the data distribution of the corresponding class is balanced. Otherwise, the coefficient of variation (CV), the ratio of standard deviation to mean value, is used to evaluate the uniformity of the data. While the closeness to 1 denotes raise in variability, the closeness to 0 denotes raise in uniformity. CV thresholding is utilized for the classes with a standard deviation greater than 1 to determine the active function to be used for computing the smoothing parameters of these classes. CV values are calculated as

in (3.16). While the threshold value for standard deviation is taken as 1, it is decided as 0.1 for the CV.

$$CV_i = \frac{Vmax_i}{M_i^{maxft_i}} \tag{3.16.}$$

Next step is the use one of the two suggested function to calculate the smoothing parameter values according to the thresholding for each class, separately. OGCNN adjust the effect of attributes by utilizing the lower limit of the one standard deviation of data, as in (3.17.), and by using the strength of the logarithmic function for the classes with large data distribution determined by threshold function. The 68-95-99.7 rule, defined for normally distributed data, is the idea behind this balancing. According to this rule, approximately 68.27% of the data placed in the range of $[M - V, M + V]$ where V represents the standard deviation and M denotes the mean value, respectively (Pukelsheim, 1994).

$$VM_i^k = V_i^k - M_i^k \tag{3.17.}$$

Finally, the difference between the maximum and minimum standard deviation values of the corresponding class is computed and divided by two to find the value of the smoothing parameter of the class having a small standard deviation. If the class has high variability, lower limits of attributes of the corresponding class calculated in (3.17.), are summed. In addition, OGCNN utilizes logarithmic function over the computed sum to balance the effects of attributes with high variability. OGCNN's formula is given in (3.18.) where N is the number of classes, P_i is the number of the instances in i_{th} class and m is the number of the features of each datum.

$$\sigma_i^l = \begin{cases} |\ln(\sum_{k=1}^{m} |V_i^k - M_i^k|)|, & (Vmax_i > 1) \text{ and } \mathrm{CV}_i > 0.1 \\ \frac{1}{2}(Vmax_i - Vmin_i), & else \end{cases} \quad \begin{matrix} 1 \le i \le N \\ 1 \le l \le P_i \\ 1 \le k \le m \end{matrix} \qquad (3.18.)$$

Algorithm 3.1 OGCNN training algorithm

Input: training data t, test_data
outputs: class

for each class i
find training data belonging to i^{th} class, tp_i
for each feature k
calculate mean value, M_i^k
calculate standard deviation, V_i^k
end for
find maximum and minimum standard deviation $Vmax_i$ and $Vmin_i$
find the indice of the maximum one $maxftr$
calculate σ according to standard deviation thresholding and assign to smoothing parameter of training data tp_i

$$\sigma_i^l = \begin{cases} |\ln(\sum_{k=1}^{m} |V_i^k - M_i^k|)|, & (Vmax_i > 1) \text{ and } \mathrm{CV}_i > 0.1 \\ \frac{1}{2}(Vmax_i - Vmin_i), & else \end{cases}$$

end for

for each training datum; t_j
compute the Euclidean distance between test and training data, $dist(j)$
apply RBF activation function, $r(j)$ with σ_j
for each class; i
compute diverge effect term,
$d(j,i) = e^{(y(j,i)-y_{max})} * y(j,i)$
compute $u_i = \sum_{j=1}^{p} d(j,i) * r(j)$ and $D = \sum_{j=1}^{p} r(j)$
compute values of normalization layer neurons; $c_i = \frac{u_i}{D}$
end for
determine the winner neuron and its value; $[o, id] = \max(c)$
end for

Alg. 1 shows the OGCNN algorithm, where $dist(j)$ denotes Euclidean distance, $r(j)$ represents pattern layer's output and $d(j,i)$ is the diverge effect term for i^{th} class and j^{th} training datum, respectively. Summation layer neurons are denoted with u_i and D. Output of the normalization layer is c_i and id is the label of the class. y_{max} is fixed to 0.9.

3.4. Datasets

In this thesis, each experiment to measure the validity of the developed methods was tested at least on 10 different data sets. The total number of datasets used in this thesis is 16. The characteristics of these datasets can be seen in Table 3.1. In this sub-section, necessary information is given about the contents of these datasets.

Table 3.1. Class distribution of Datasets

Dataset Name	Number of images	Number of classes
Abnormal Objects	626	6
ETHZ Shapes	255	5
Glassense Banknotes	492	5
JF-30	1479	30
KTH-Animals	1740	19
Leeds Butterflies	832	10
MSRC-v1	240	8
MSRC-v2	591	20
Oxford Flowers	1360	17
Ponce Birds	600	6
Ponce Butterflies	619	7
Swedish Leaf	1125	15
Tud Leaf	186	3
CIFAR-10	60000	10
CIFAR-100	60000	100
ImageNet	1.3M	1000

3.4.1. Abnormal Objects Dataset

Abnormal Objects Dataset (Saleh et al., 2013) contains unusual images of various objects. It includes 626 images divided into 6 different classes. These classes can be seen in Table 3.2.

Table 3.2. Abnormal objects dataset contents

Class Name	Number of images
Airplane	100
Boat	114
Car	110
Chair	109
Motorbike	85
Sofa	108

3.4.2. ETHZ Shapes Dataset

ETHZ Shapes Dataset (Ferrari et al., 2010) includes 255 images of 5 different classes. These classes can be seen in Table 3.3.

Table 3.3. ETHZ shapes dataset contents

Class Name	Number of images
Apple	40
Bottle	48
Giraffe	87
Mug	48
Swan	32

3.4.3. Glassense Banknotes Dataset

Glassense Banknotes Dataset contains 492 images of 5 different euro banknotes. These images are different in terms of the positions and visible areas of the banknotes. The types of banknotes in this data set are given in Table 3.4.

Table 3.4. Glassense banknotes dataset contents

Class Name	Number of images
5 euro	114
10 euro	145
20 euro	110
50 euro	71
100 euro	52

3.4.4. KTH-Animals Dataset

KTH-Animals Dataset (Afkham et al., 2008) includes 1740 images of 19 different animal species. These species are given in Table 3.5.

.

Table 3.5. KTH-Animals dataset contents

Class Name	Number of images
Bear	105
Cougar	100
Cow	97
Coyote	100
Deer	100
Elephant	100
Giraffe	84
Goat	99
Gorilla	83
Horse	100
Kangaroo	90
Leopard	100
Lion	98
Panda	97
Penguin	80
Sheep	68
Skunk	62
Tiger	100
Zebra	77

3.4.5. Jena Flowers Dataset

Jena flowers dataset (Seeland et al., 2017) contains 1479 images of 30 flowers species found on the city of Jena. These species can be seen in Table 3.6.

Table 3.6. Jena flowers dataset contents

Class Name	Number of images
Ophrys insectifera	11
Knautia arvensis	65
Hypericum perforatum	41
Hippocrepis comosa	66
Scabiosa columbaria	54
Centaurea scabiosa	70
Pimpinella saxifrage	64
Anthericum ramosum	28
Centaurea jacea	46
Fragaria viridis	60
Lotus corniculatus	36
Prunella grandiflora	48
Origanum vulgare	66
Anemone sylvestris	54
Aster amellus	13
Ononis repens	24
Stachys recta	45
Bulpeurum falcatum	63
Sanguisorba minor	55
Polygala comosa	61
Ajuga genevensis	38
Euphorbia cyparissias	49
Inula hirta	67
Salvia pratensis	60
Geranium sanguineum	56
Polygala amarella	35
Trifolium montanum	61
Teucrium chamaedrys	68
Inula salicina	63
Asperula cynanchica	12

3.4.6. Leeds Butterflies Dataset

Leeds Butterflies Dataset (Wang et al., 2009) contains 832 images of 10 different butterfly species. These species are given in Table 3.7.

Table 3.7. Leeds butterflies dataset contents

Class Name	Number of images
Danaus plexippus	82
Heliconius charitonius	93
Heliconius erato	61
Junonia coenia	90
Lycaena phlaeas	88
Nymphalis antiopa	100
Papilio cresphontes	89
Pieris parae	55
Vanessa atalanta	90
Vanessa cardui	94

3.4.7. MSRC-v1 Dataset

MSRC-v1 dataset (Winn et al., 2005) includes 240 images divided into 8 different classes. These classes are as given in Table 3.8.

Table 3.8. MSRC-v1 dataset contents

Class Name	Number of images
Grass	30
Tree	30
Building	30
Airplane	30
Cow	30
Human Face	30
Car	30
Bicycle	30

3.4.8. MSRC-v2 Dataset

MSRC-v2 dataset (Beck and Teboulle, 2009) contains 591 images divided into 20 classes. These classes are as given in Table 3.9.

Table 3.9. MSRC-v2 dataset contents

Class Name	Number of images
Grass	30
Tree	30
Building	30
Airplane	30
Cow	30
Human Face	30
Car	30
Bicycle	30
Sheep	30
Flower	32
Signboard	30
Bird	34
Book	30
Seat	30
Cat	24
Dog	30
Street	30
Boat	30
Human	30
Sea	21

3.4.9. Oxford Flowers Dataset

Oxford flowers dataset (Nilsback and Zisserman, 2006) contains 1360 different images of 17 flower species. These species are as given in Table 3.10.

Table 3.10. Oxford flowers dataset contents

Class Name	Number of images
Cowslip	30
Tulip	30
Tiger lily	30
Crocus	30
Bluebell	30
Lily valley	30
Snowdrop	30
Windflower	30
Sunflower	30
Pansy	32
Iris	30
Fritillary	34
Dandelion	30
Daisy	30
Daffodil	24
Coltsfoot	30
Buttercup	30

3.4.10. Ponce Birds Dataset

Ponce Birds Dataset (Lazebnik et al., 2005) contains 600 images of 6 different bird species. These species are as given in Table 3.11.

Table 3.11. Ponce birds dataset contents

Class Name	Number of images
Egret	30
Mandarin	30
Snowy Owl	30
Puffin	30
Toucan	30
Wood Duck	30

3.4.11. Ponce Butterflies Dataset

Ponce Butterflies Dataset (Lazebnik et al., 2004) includes 619 images of 7 different butterfly species. These species are given in Table 3.12.

Table 3.12. Ponce butterflies dataset contents

Class Name	Number of images
Admiral	111
Swallowtail	42
Machaon	83
Monarch 1	74
Monarch 2	84
Peacock	134
Zebra	91

3.4.12. Swedish Leaf Dataset

Swedish Leaf Dataset (Söderkvist, 2001) contains 1125 images of leaves of 15 different tree species. These tree species are given in Table 3.13.

Table 3.13. Swedish leaf dataset contents

Class Name	Number of images
Ulmus carpinifolia	75
Acer	75
Salix aurita	75
Quercus	75
Alnus incana	75
Betula pubescens	75
Salix alba 'Sericea'	75
Populus tremula	75
Ulmus glabra	75
Sorbus aucuparia	75
Salix sinerea	75
Populus	75
Tilia	75
Sorbus intermedia	75
Fagus silvatica	75

3.4.13. Tud Leaf Dataset

Tud Leaf Dataset (Weber, 1999) includes 186 images of leaves of 3 different tree species found in Caltech.

3.4.14. CIFAR-10

CIFAR-10 dataset (Krizhevsky and Hinton, 2009) is a very popular image dataset containing 60000 32x32 images divided into 10 different classes. These classes are given in Table 3.14.

Table 3.14. CIFAR-10 dataset contents

Class Name	Number of images
Airplane	6000
Automobile	6000
Bird	6000
Cat	6000
Deer	6000
Dog	6000
Frog	6000
Horse	6000
Ship	6000
Truck	6000

3.4.15. CIFAR-100

CIFAR-100 dataset (Krizhevsky and Hinton, 2009) contains 60000 32x32 images divided into 100 different classes.

3.4.16. ImageNet Dataset

ImageNet dataset (Krizhevsky et al., 2012) is a very popular dataset containing about 1.3 million images divided into 1000 different classes. It is used in traditionally organized ILSVRC competitions.

Figure 3.6 A few images from ImageNet Dataset (Krizhevsky et al., 2012)

3.5. Popular Deep Learning Structures

In this subsection popular deep learning structure LeNet, AlexNet, OverFeat, GoogLeNet, and VGGNet are given.

3.5.1. LeNet

LeNet (LeCun et al., 1998) is the first published CNN structure. It contains two convolutional and pooling layer pairs and used for classifying MNIST dataset (LeCun et al., 1995). LeNet structure is given in Figure 3.7. Because the computers were not strong enough to train larger networks for larger datasets at that time, they could not be tested for a long period of time.

3.5.2. AlexNet

AlexNet (Krizhevsky et al., 2012) is the winner model of ILSVRC-2012 competition with a 16.4% top5 error rate on the ImageNet dataset. AlexNet consists of a fully connected network structure that follows two parallel feature

extractor modules as given in Figure 3.8. Each module contains 5 convolutional layers and 3 pooling layers. AlexNet's high performance increased the interest in convolutional structures.

3.5.3. OverFeat

OverFeat (Sermanet et al., 2013) is the winner of the localization task in ILSVRC-2013 competition. It contains 5 convolutional layers, 3 pooling layers, and 3 fully connected layers. OverFeat is popularly used in transfer learning applications. OverFeat structure is given in Figure 3.9.

3.5.4. GoogLeNet

GoogLeNet is the winner of ILSVRC2014 competition with a top5 error rate of 6.67. It has a similar structure to LeNet; however, it includes inception modules which provide convolution operations with different sized filters and pooling operation in a single structure. GoogLeNet achieved significantly higher performance values with far fewer parameters with the inclusion of inception modules.

3.5.5. VGGNet

VGGNet is another important model participated in ILSVRC2014. It is composed of 16 convolutional layers, 5 pooling layers, and 3 fully connected layers. All convolutional layers of this structure apply 3x3 convolution operations. Although it has very successful performance values, having a large number of parameters makes it difficult to use.

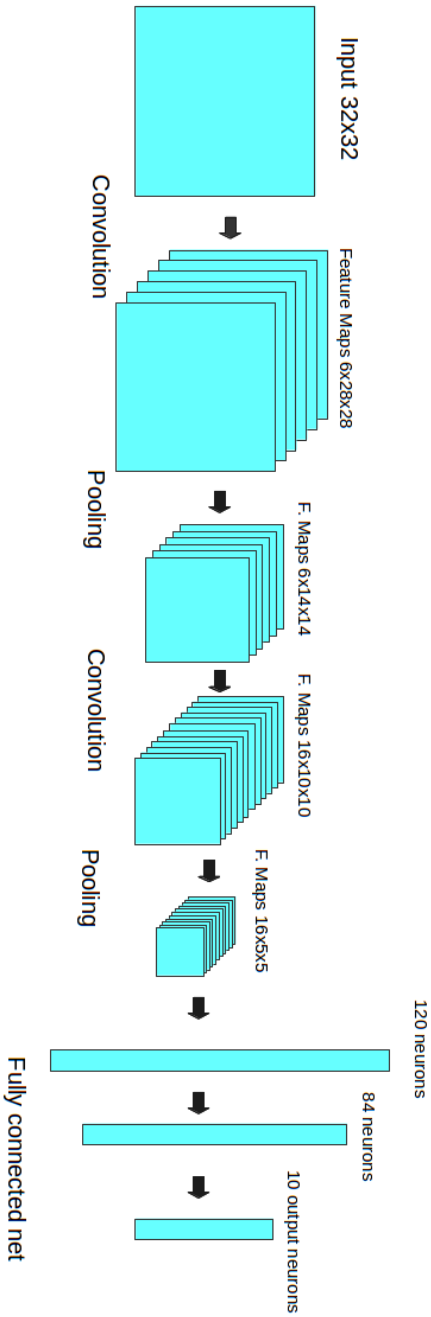

Figure 3.7 LeNet Structure

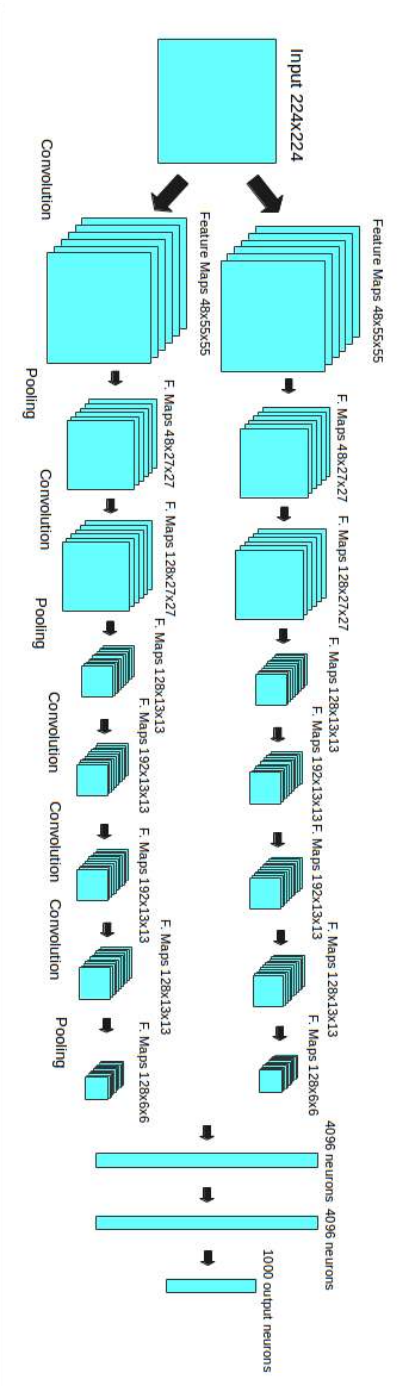

Figure 3.8 AlexNet structure

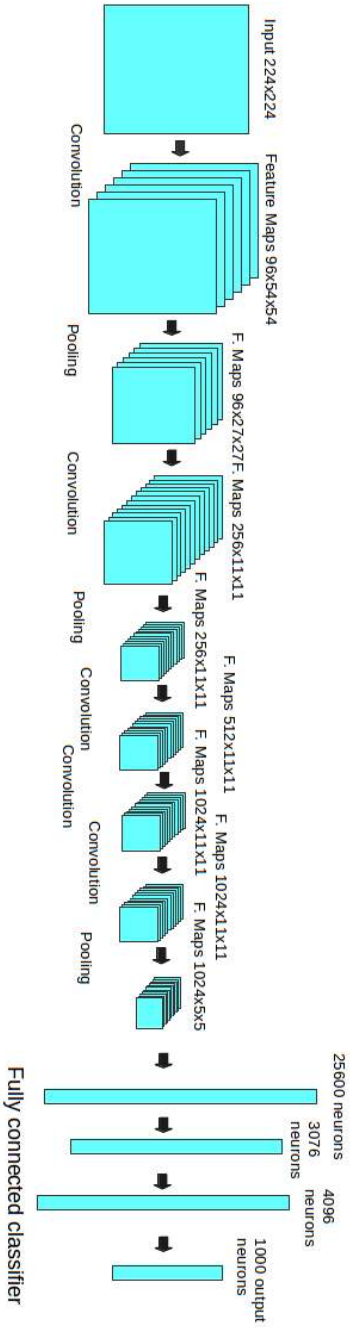

Figure 3.9 OverFeat structure

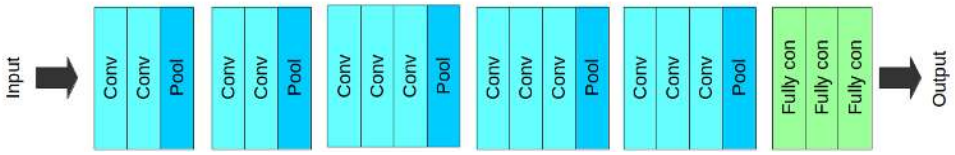

Figure 3.10 VGGNet Structure

3.6. Performance Evaluation Methods

N-fold cross-validation is a technique used for increasing the validity of the results in machine learning experiments. Accuracy, precision, recall, and f1-score are popular performance metrics for classification tasks.

3.6.1. N-Fold Cross-Validation

N-fold cross validation is a technique used for assuring the quality of the experiment results. In this technique, N is determined intuitively. After that, the dataset is divided into N groups randomly. While $N-1$ of the groups are utilized for training the model, the other group is used for test purpose. After that, another group is taken as validation data. After all, N groups are used as test data, the average performance value of N experiments are taken as the final score of the experiment. In this way, the accuracy of the results is assured. N is usually determined as 5 or 10.

3.5.2. Evaluation Metrics

There are four standard error measures utilizing to evaluate the classification performance of a model which are accuracy, precision, recall, and F1-score. Accuracy is the most popular metric and it is the ratio of correctly classified instances among all instances in a dataset. Accuracy value can be seen in (3.19) where t is the number of correctly classified instances and n is the number of instances in a dataset. In this thesis, two different accuracy values are calculated which are Top1 and Top5 accuracy values. When calculating the Top1 accuracy

value, the class with the highest probability given by the network must be the correct class of the test data. In Top5 accuracy, it is sufficient for the class of test data to be in the 5 classes that the network gives the highest probability. Top5 accuracy value can be used in addition to or instead of Top1 accuracy value in some cases.

$$Acc = \frac{t}{n} \tag{3.19.}$$

Precision is the ratio of correctly classified instances among all instances in a class. Precision value of class c, P_c, can be seen in (3.20.) where n_c is the total number of the instances in class c and t_c is the number of correctly classified instances in class c. If there is an equal number of instances in each class within a dataset, average precision value of the dataset will be equal to the accuracy value.

$$P_c = \frac{t_c}{n_c} \tag{3.20.}$$

The recall is the ratio of correctly classified instances among all instances classified in that class. Recall value of class c is shown in (3.21) where k_c is the total number of the instances classified as belong to class c and t_c is the number of correctly classified instances in class c.

$$R_c = \frac{t_c}{k_c} \tag{3.21.}$$

F1-score is a combination of precision and recall measure. F1-score of class c is shown in (3.22.) where P_c is the precision of class c and R_c is the recall of class c.

$$F_c = \frac{2(P_c * R_c)}{P_c + R_c} \tag{3.22.}$$

Precision, recall, and F1-score of each class are computed for each fold and the average values of these values are taken to measure the performance of the method for a dataset.

4. METHODS

In this section, the structures and algorithms of the Differential Convolutional Neural Network and Deep Convolutional General Classifier Neural Network are given in detail. Main ideas behind the proposed structures are demonstrated.

4.1. Differential Convolutional Neural Network

Differential Convolutional Neural Network, utilizes a novel convolution technique named as Differential Convolution. This technique allows producing five feature maps instead of one from a single convolutional filter. This reduces the number of filters required for each layer and increases the number of features produced by a single filter. The aim of the differential convolution is to extract pattern orientation of a filter for each direction. This method not only improves the performance but also provides more effective learning by utilizing the errors on the neighborhoods during the error back-propagation. In this subsection, details, and implementation of the Differential Convolution, its effects on the neural network structure and its training algorithm are given.

4.1.1. Differential Convolution

Convolution operation is one of the most important factors in the success of deep learning structures. This operation allows the recognition of the visual patterns by sliding a number of kernels over the input image. The idea behind the proposed convolution technique is the consideration of directional changes among a pixel and its neighbors. Differential convolution analyses the pattern orientation of a pixel and its neighbors by the additional change amount calculations. As in the mathematical differentiation, sequential changes are taken into consideration by calculating the difference among the activations of the pixels. While considering the images in the form of a superposition of waves with different frequencies an

additional successive differential calculation must improve the pattern orientation ability of the standard convolution. This fact leads the proposed method containing an improved convolution approach reinforced with a map of difference signal addition. These additional differential maps are produced from the feature map generated with the traditional convolution by applying pre-defined constant filters. The fixed filters are shown in Figure 4.1. Each of these filters is used to calculate differences in one direction. As a result, additional feature maps containing signal differences for each direction are generated.

Let g_1 denotes the feature map generated with traditional convolution and g_2, g_3, g_4 and g_5 represents the additional four feature maps. The values of the neurons of these feature maps are computed as in (4.1.), (4.2.), (4.3.) and (4.4.). If the size of g_1 is assumed as MxN, sizes of the additional maps g_2, g_3, g_4 and g_5 will be $(M-1)xN$, $Mx(N-1)$, $(M-1)x(N-1)$ and $(M-1)x(N-1)$. Since the size of the feature maps must be the same, these produces additional feature maps are padded with zeros and brought to the size of the first one. Additionally, the receptive field of the filter is enlarged by one unit along two axes with this operation. Examples of the differential feature maps are given in Figure 4.2.

$$g_{2,i,j} = g_{1,i,j} - g_{1,i+1,j}, 1 \leq i < M, 1 \leq j \leq N \quad (4.1.)$$

$$g_{3,i,j} = g_{1,i,j} - g_{1,i,j+1}, 1 \leq i \leq M, 1 \leq j < N \quad (4.2.)$$

$$g_{4,i,j} = g_{1,i,j} - g_{1,i+1,j+1}, 1 \leq i < M, 1 \leq j < N \quad (4.3.)$$

$$g_{5,i,j} = g_{1,i+1,j} - g_{1,i,j+1}, 1 \leq i < M, 1 \leq j < N \quad (4.4.)$$

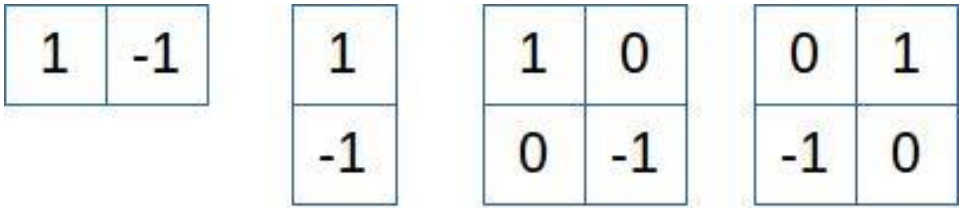

Figure 4.1 Differential filters

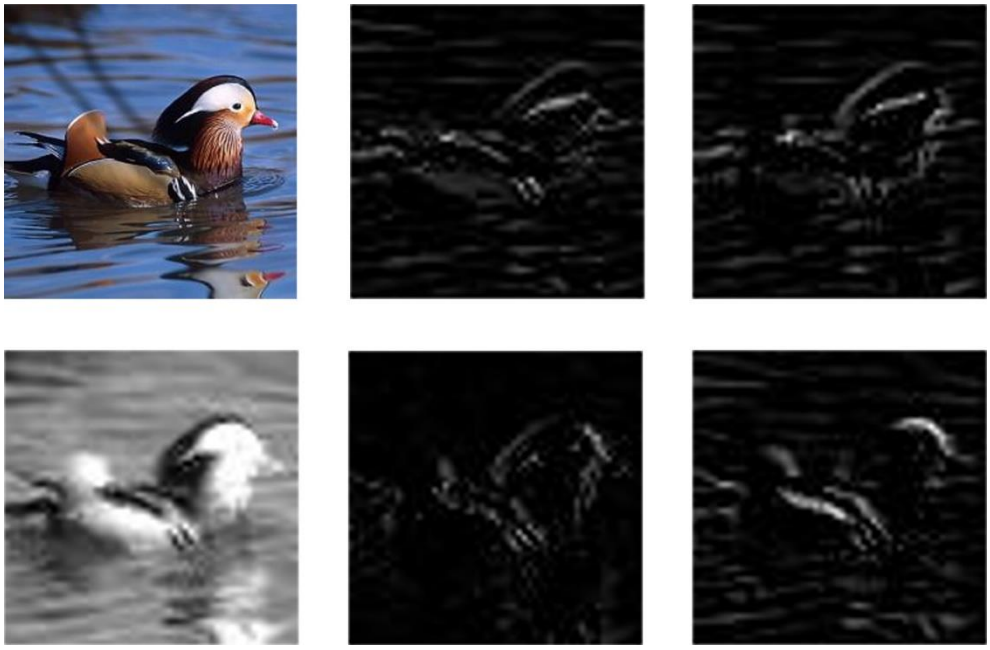

Figure 4.2. Differential feature map examples. Top Left: Original image; Bottom Left: Traditional feature map; Others: Differential feature maps

When performing a differential convolution process, the first feature map produced by the traditional method is subjected to an additional convolution process with predefined constant filters. This increases the convolution depth performed on a single differential convolution layer from 1 to 2. Moreover, it is not necessary to define any new parameters while increasing this depth. The predefined filters used in differential convolution are also useful to find edges and corners on images. Such filters have been successfully used for the same purposes in structures such as cellular neural networks (Chua and Yang, 1988). Increasing the

depth of convolution in this process reduces the number of convolutional layers that should be utilized.

Increasing the number of feature maps will increase the number of parameters in the next layer at the same rate. Although it may seem like a disadvantage in this respect, it should be considered that a differential convolutional layer acts as two convolutional layers. In addition, as it produces feature maps via existing filters, there is no need for new parameters to be optimized.

Generally, a convolutional layer is followed by a pooling layer. While average pooling passes the average difference, maximum pooling transfers the largest difference on the additional maps to the next layer. Therefore, pooling operation appears to be reasonable and advantageous in the differential feature maps as for the general maps. Other pooling operations, such as L2 pooling, may also be used over differential maps.

4.1.2. Structure

Differential convolution generates four additional feature maps for each kernel. While the first map is produced by general convolution, the additional four feature maps are produced by computing the neighborhood activation differences by utilizing constant filters. The size of the additional maps produced becomes smaller than the first map. This mismatch is fixed by padding the additional maps with zeros. This whole operation can be seen in Figure 4.3. This figure shows a differential convolution operation with a 3x3 filter on an 8x8 image. Differential convolution operation also increases the receptive area by one pixel for each dimension.

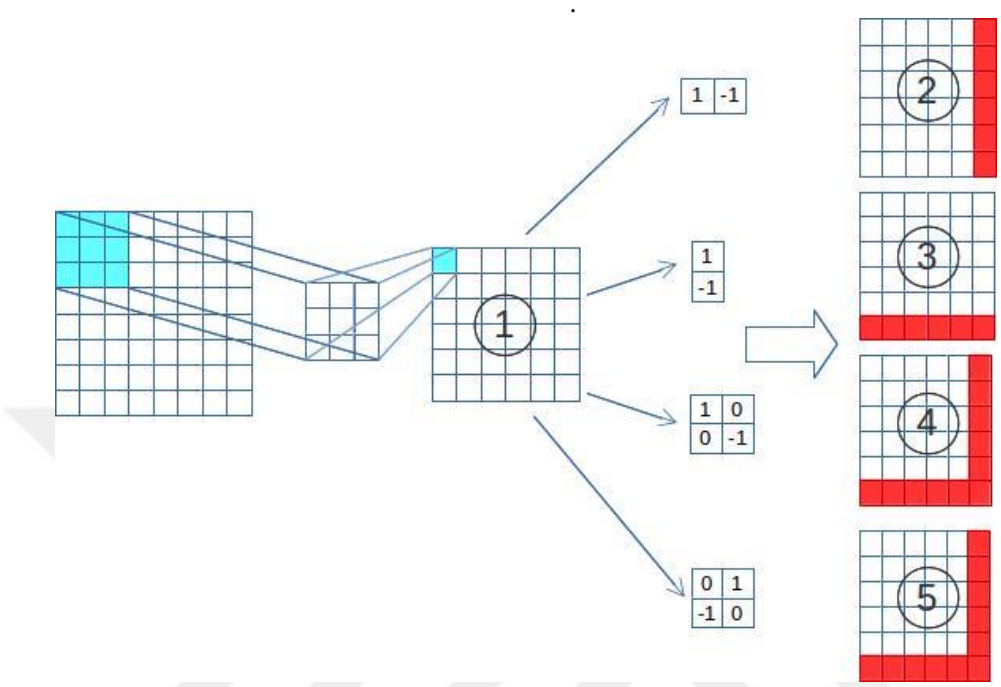

Figure 4.3 Differential CNN Structure

4.1.3. Accumulative Back-propagation Algorithm

Differential convolution with new feature maps requires improvement on the back-propagation algorithm. During the training of this structure, errors will be transmitted to the generated five feature maps from the next layer. The main issue is how to handle the error values on the five feature maps to compute the errors needed for the training of the relevant filter and the errors to be transmitted to the previous layer. Therefore, a back-propagation based learning algorithm called accumulative error back-propagation is proposed for the Differential Convolutional Neural Network structures. In this algorithm, the errors on each feature map are multiplied by the corresponding constant valued filter and summed with the errors on the first feature map. This calculated error matrix is used to train the relevant filter and propagated the error through the filter.

Let h_1 denotes the error matrix transmitted to the first feature map, h_2, h_3, h_4, h_5 represents the error matrixes transmitted to the four additional feature maps and E denotes the error matrix to be calculated. Assume the size of the feature maps be MxN, elements of E matrix can be calculated as in (4.5.), (4.6.), and (4.7.).

$$E_{i,j} = h_{1,i,j} - h_{2,i,j-1} + h_{2,i,j} - h_{3,i-1,j} + h_{3,i,j} - h_{4,i-1,j-1} + h_{4,i,j} - h_{5,i-1,j} + h_{5,i,j-1}, 1 < i < M, 1 < j < N \quad (4.5.)$$

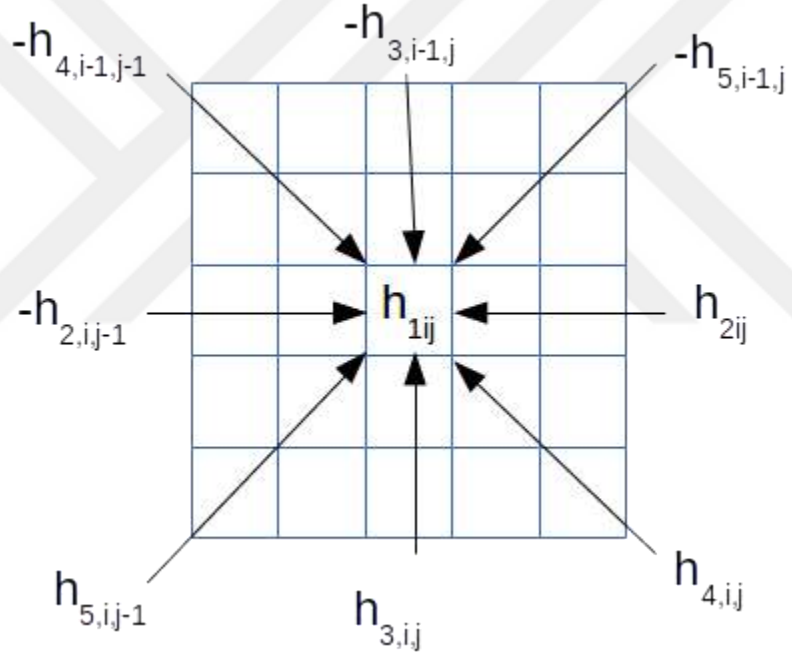

Figure 4.4 Error transmission through neighbors with directions

Neurons which are not placed at the corners or the edges get an error signal from all the adjacent neurons as it is seen in (4.5.). Corner neurons get an error signal from three adjacent neurons as in (4.6). Finally, the neurons on the edges receive a fault signal from five adjacent neurons as in (4.7.).

$$E_{i,j} = \begin{cases} h_{1,i,j} + h_{2,i,j} + h_{3,i,j} + h_{4,i,j} & i = 1, j = 1 \\ h_{1,i,j} - h_{2,i,j-1} + h_{3,i,j} + h_{5,i,j-1} & i = 1, j = N \\ h_{1,i,j} + h_{2,i,j-1} - h_{3,i-1,j} - h_{5,i-1,j} & i = M, j = 1 \\ h_{1,i,j} - h_{2,i,j-1} - h_{3,i-1,j} - h_{4,i-1,j-1} & i = M, j = N \end{cases} \quad (4.6.)$$

$$E_{i,j} = \begin{cases} h_{1,i,j} - h_{2,i,j-1} + h_{2,i,j} + h_{3,i,j} + h_{4,i,j} + h_{5,i,j-1} & i = 1, 1 < j < N \\ h_{1,i,j} - h_{2,i,j-1} + h_{2,i,j} - h_{3,i-1,j} - h_{4,i-1,j-1} - h_{5,i-1,j} & i = M, 1 < j < N \\ h_{1,i,j} + h_{2,i,j} - h_{3,i-1,j} + h_{3,i,j} + h_{4,i,j} - h_{5,i-1,j} & 1 < i < M, j = 1 \\ h_{1,i,j} - h_{2,i,j-1} - h_{3,i-1,j} + h_{3,i,j} - h_{4,i-1,j-1} + h_{5,i,j-1} & 1 < i < M, j = N \end{cases} \quad (4.7.)$$

4.1.4. SoftMax Utilized Accumulative Backpropagation

Accumulative backpropagation approach yields good results; however, it has brought some problems because of the direct summation of the errors. The first problem is the need for using a small learning rate since the accumulated error can be relatively big for error back-propagation. The second important problem is the equality of the effects of the errors transmitted from each one of the neighbors and the main feature map on the calculated total error. This causes the effect of the error transmitted from the main feature map to be very low. Therefore, it is necessary to balance the errors transmitted from neighbors by multiplying the errors with specific weights. In this algorithm, the effect of the error coming from neighbors was restricted by using SoftMax function and the effect of the total neighboring error and effect of the error transmitted from the main feature map were equalized. Each neighborhood error is multiplied by a calculated rate by using the calculated differences in the forward pass and all the neighborhood errors are collected and the total neighborhood error is calculated. Then, the error transmitted from the first feature map and the computed neighborhood error is summed. This last computed error is utilized to train the relevant filter and it is propagated to the previous layer. First, the denominator values of the SoftMax function is calculated as in (4.8), (4.9) and (4.10).

$$D_{i,j} = e^{g_{2,i,j-1}} + e^{g_{2,i,j}} + e^{g_{3,i-1,j}} + e^{g_{3,i,j}} + e^{g_{4,i-1,j-1}} + e^{g_{4,i,j}} + e^{g_{5,i-1,j}} + e^{g_{5,i,j-1}}, 1 < i < M, 1 < j < N \quad (4.8.)$$

$$D_{i,j} = \begin{cases} e^{g_{2,i,j}} + e^{g_{3,i,j}} + e^{g_{4,i,j}} & i = 1, j = 1 \\ e^{g_{2,i,j-1}} + e^{g_{3,i,j}} + e^{g_{5,i,j-1}} & i = 1, j = N \\ e^{g_{2,i,j-1}} + e^{g_{3,i-1,j}} + e^{g_{5,i-1,j}} & i = M, j = 1 \\ e^{g_{2,i,j-1}} + e^{g_{3,i-1,j}} + e^{g_{4,i-1,j-1}} & i = M, j = N \end{cases} \quad (4.9.)$$

$$D_{i,j} = \begin{cases} e^{g_{2,i,j-1}} + e^{g_{2,i,j}} + e^{g_{3,i,j}} + e^{g_{4,i,j}} + e^{g_{5,i,j-1}} & i = 1, 1 < j < N \\ e^{g_{2,i,j-1}} + e^{g_{2,i,j}} + e^{g_{3,i-1,j}} + e^{g_{4,i-1,j-1}} + e^{g_{5,i-1,j}} & i = M, 1 < j < N \\ e^{g_{2,i,j}} + e^{g_{3,i-1,j}} + e^{g_{3,i,j}} + e^{g_{4,i,j}} + e^{g_{5,i-1,j}} & 1 < i < M, j = 1 \\ e^{g_{2,i,j-1}} + e^{g_{3,i-1,j}} + e^{g_{3,i,j}} + e^{g_{4,i-1,j-1}} + e^{g_{5,i,j-1}} & 1 < i < M, j = N \end{cases} \quad (4.10.)$$

After calculating the denominator values, the SoftMax rates are determined by dividing the exponential values of the forward pass activation differences which are located in the additional feature maps by the denominator values as in (4.11).

$$P_{k,i,j} = \frac{e^{g_{k+1,i,j}}}{D_{i,j}}, 1 \le i \le M, 1 \le j \le N, 1 \le k \le 4 \quad (4.11.)$$

The neighborhood error values are multiplied by the calculated rates and total neighboring error values are computed as in (4.12).

$$c_{k,i,j} = h_{k+1,i,j} P_{i,j}, 1 \le i \le M, 1 \le j \le N, 1 \le k \le 4 \quad (4.12.)$$

Finally, the error matrix to be used for the training of the relevant filter and to be propagated back is calculated as in (4.13), (4.14) and (4.15). These three equations are the same as the previous accumulative equations, but only the neighborhood errors have replaced with new calculated errors. In the previous method, while the corner neurons received an error computed with a summation of

4 error values and the edge neurons received an error computed with a summation of 6 error values, the remainder neurons received an error calculated with a summation of 9 error values. This caused problems with the error density on the neurons. The amount of error transmitted over each neuron is equalized with this new method. Because the errors transmitted from the neighbors were normalized and the total amount of error on each neuron become equal.

$$E_{i,j} = h_{1,i,j} - c_{2,i,j-1} + c_{2,i,j} - c_{3,i-1,j} + c_{3,i,j} - c_{4,i-1,j-1} + c_{4,i,j} - c_{5,i-1,j} + c_{5,i,j-1}, 1 < i < M, 1 < j < N \quad (4.13.)$$

$$E_{i,j} = \begin{cases} h_{1,i,j} + c_{2,i,j} + c_{3,i,j} + c_{4,i,j} & i = 1, j = 1 \\ h_{1,i,j} - c_{2,i,j-1} + c_{3,i,j} + c_{5,i,j-1} & i = 1, j = N \\ h_{1,i,j} + c_{2,i,j-1} - c_{3,i-1,j} - c_{5,i-1,j} & i = M, j = 1 \\ h_{1,i,j} - c_{2,i,j-1} - c_{3,i-1,j} - c_{4,i-1,j-1} & i = M, j = N \end{cases} \quad (4.14.)$$

$$E_{i,j} = \begin{cases} h_{1,i,j} - c_{2,i,j-1} + c_{2,i,j} + c_{3,i,j} + c_{4,i,j} + c_{5,i,j-1} & i = 1, 1 < j < N \\ h_{1,i,j} - c_{2,i,j-1} + c_{2,i,j} - c_{3,i-1,j} - c_{4,i-1,j-1} - c_{5,i-1,j} & i = M, 1 < j < N \\ h_{1,i,j} + c_{2,i,j} - c_{3,i-1,j} + c_{3,i,j} + c_{4,i,j} - c_{5,i-1,j} & 1 < i < M, j = 1 \\ h_{1,i,j} - c_{2,i,j-1} - c_{3,i-1,j} + c_{3,i,j} - c_{4,i-1,j-1} + c_{5,i,j-1} & 1 < i < M, j = N \end{cases} \quad (4.15.)$$

4.2. Deep Convolutional Generalized Classifier Neural Network

Deep convolutional generalized classifier neural network (DC-GCNN) structure is another novel neural network structure proposed in this thesis. DC-GCNN is composed of convolutional layers and back-propagation adapted OGCNN classifier. OGCNN is a one pass kernel-based classifier neural network structure (Ozyildirim and Avci, 2016). OGCNN part requires training features in the pattern layer before the training of convolutional part weights. Therefore, a feature extraction method must be utilized to obtain these features. Convolutional part of OverFeat pre-trained neural network structure is used as feature extractor in all experiments since the application data are images in this work. OverFeat

(Sermanet et al., 2013) is a popular pre-trained neural network structure which is frequently utilized for feature extraction and transfer learning. ImageNet (Deng et al., 2009) image database including 1.3 million images of 1000 different classes was used for training of OverFeat structure. At the beginning of the training, each training image is passed through the convolutional part of OverFeat neural network structure and features are obtained. Fine-tuning operation can be performed before the feature extraction. Each training image is denoted by a neuron in the pattern layer of DC-GCNN. Image attributes acquired from OverFeat are stored in the related neuron of the DC-GCNN. Structure of DC-GCNN is given in Figure 4.5.

DC-GCNN requires tuning the convolutional filter weights of the convolutional part and the optimizing the smoothing parameter of the classifier. OGCNN methodology is used for tuning the smoothing parameter; therefore, the classification part does not require any optimization of parameters during the training process. The features produced with OverFeat are used for statistical calculations of OGCNN and the correct smoothing parameters are calculated for each dataset. Additionally, a method must be utilized to train the convolutional part weights by considering the error at the output of the classifier. For this to happen, the error must be propagated back over the classifier in order to be used in the training of the convolutional part filters. A backward calculation from the classifier output to classifier input is required. In this work, a method propagating the error in a backward direction by utilizing the gradient descent approach is proposed. After that, the convolutional part is trained with error back-propagation using the calculated errors.

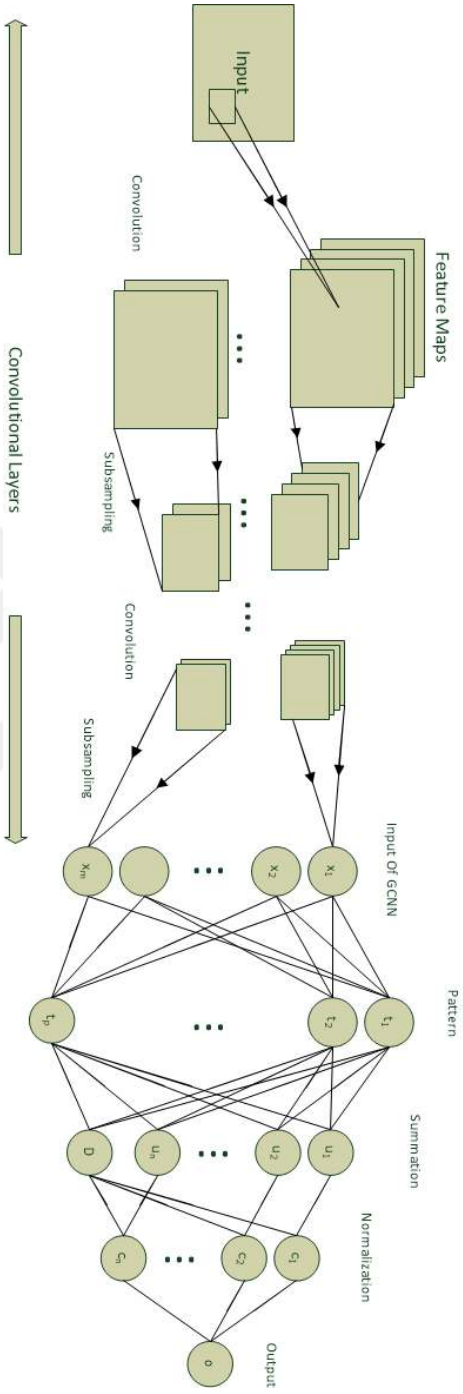

Figure 4.5. DC-GCNN Structure

x_k represents the output vector of the convolutional part of the network where $1 \leq k \leq P$, P is the feature count, the output of DC-GCNN is computed as in (3.5.) to (3.12.). The error back-propagation process is given in (4.16) to (4.20). DC-GCNN has as many outputs as the number of classes. The training samples are labelled as in (4.8.) where $y(i)$ represents the label for i^{th} class and N is the number of the classes. If the training sample belongs to i^{th} class then its i^{th} label will be 0.9 and others will be 0.1. The errors at the output layer are calculated as given in (4.9.) where c denotes the output vector of the DC-GCNN.

$$y(i) = \begin{cases} 0.9 \; if \; datum \; belongs \; to \; i^{th} \; class \\ 0.1 \; otherwise \end{cases} \quad 1 < i < N \tag{4.16.}$$

$$e = (y(i) - c(i))^2 \tag{4.17.}$$

The derivative of the error function to be propagated is given in (4.18.) where x represents the weights of the convolutional layers to be optimized.

$$\frac{\partial e}{\partial x} = 2 * \big(y(i) - c(i)\big) * \frac{\partial c}{\partial x} \tag{4.18.}$$

Therefore, the amount of the error accumulated at the end of the j^{th} pattern layer neuron is computed as in (4.19.) where N denotes the number of output layer neurons. In the forward pass, the activation values were multiplied by diverge effect term of the corresponding class. Therefore, on the backward pass, the errors are also multiplied by these terms.

$$\frac{\partial e}{\partial x} = \left(\sum_{i=1}^{N} 2\big(y(i) - c(i)\big) * div(j,i) + \sum_{i=1}^{N} 2\big(y(i) - c(i)\big) \right) \frac{\partial p}{\partial x} \tag{4.19.}$$

In (4.19.), the first sum represents the error comes from the numerator neuron, while the second sum represents the error comes from the denominator. Since the error transmitted from the denominator affects all the pattern layer neuron equally, it can be neglected. Hence, the following step is to take the derivative of distance function according to each neuron in the pattern layer. In this derivative, each feature of the input layer is considered separately. The amount of error accumulated for k^{th} feature which will be propagated to the convolutional part of the neural network is given in (4.20.) where P denotes the number of pattern layer neurons. This calculated error is back-propagated from each feature output of the convolutional layer and weights of the convolutional filters are optimized.

$$\frac{\partial e}{\partial x_k} = \sum_{j=1}^{P} \left(-\frac{(x_{kj} - t_{kj})}{\sigma_j^2} * \exp(-dist(j)/\sigma_j^2) * \sum_{i=1}^{N} 2\big(y(i) - c(i)\big) * div(j,i)\right) \qquad (4.20.)$$

Before the DC-GCNN structure is used to classify a data set, the features of the data set are obtained by using a feature extractor. If the data set is different or a large data set, the fine-tuning process can be applied to improve the quality of the features. Using the features produced, the effective smoothing parameter is calculated. Produced feature values are placed in DC-GCNN classifier and neural network training is started. After this stage, standard artificial neural network training is performed. Whole training algorithm can be seen in

Algorithm 4.1.

Another important feature of the DC-GCNN structure is its ability to decide the features to be learned. The traditional CNN structure is literally a black box. But using the DC-GCNN structure allows the features produced with the desired feature extraction algorithm to be taught to a CNN structure. In the studies, it was seen that the applicability of the features produced with a much deeper structure to a simpler structure is possible by using this methodology.

Algorithm 4.1. DC-GCNN training algorithm

Input: training data t, test_data, class count N **outputs:** class
apply fine-tuning operation to the pre-trained neural network if required.
pass all training images through the pre-trained network, generate all features
normalize all features in the range of $[-1,+1]$
for each class i
find training data belonging to i^{th} class, tp_i
for each feature k
calculate mean value, M_i^k and standard deviation, V_i^k
end for
find the maximum and minimum standard deviation $Vmax_i$ and $Vmin_i$
find the indice of the maximum one $maxftr$
calculate σ according to standard deviation thresholding and assign to smoothing parameter of training data tp_i

$$\sigma_i^l = \begin{cases} |\ln(\sum_{k=1}^{m} |V_i^k - M_i^k|)|, & (Vmax_i > 1) \text{ and } \mathrm{CV}_i > 0.1 \\ \frac{1}{2}(Vmax_i - Vmin_i), & else \end{cases}$$

end for
initialize convolutional part weights randomly
while total_error in the test data decreases
for each training image; img_m
pass img_m through the convolutional part and calculate the features
for each pattern node;
compute the Euclidean distance between node features and generated features, $dist(i)$
apply RBF activation function, $r(j)$ with σ_j
for each class; i
compute diverge effect term,
$d(j,i) = e^{(y(j,i)-y_{max})} * y(j,i)$
calculate $u_i = \sum_{j=1}^{p} d(j,i) * r(j), 1 \le i \le N$
and $D = \sum_{j=1}^{p} r(j)$
compute values of normalization layer neurons; $c_i = \frac{u_i}{D}$
end for
end for
error of each output neuron is calculated with $e = (y(i) - c(i))^2$
where $y(i) = \begin{cases} 0.9\ if\ datum\ belongs\ to\ i^{th}\ class \\ 0.1\ otherwise \end{cases}$ $1 < i < N$
error of the k_{th} output neuron of convolutional part is calculated with

$$\frac{\partial e}{\partial x_k} = \sum_{j=1}^{P} \left(-\frac{(x_{kj} - t_{kj})}{\sigma_j^2} * \exp(-dist(j)/\sigma_j^2) * \sum_{i=1}^{N} 2\big(y(i) - c(i)\big) * div(j,i)\right)$$

back propagate the error through the convolutional layers
end for
end while

DC-GCNN is a high-performance classifier neural network structure, but the major problem of the methodology is the high complexity of the calculations in the pattern layer. If the pattern layer includes a high number of neurons, this may increase the training time of the network. In such cases, the reduction algorithms used in the GRNN methodology must be utilized.

5. RESULTS AND DISCUSSIONS

In this section, all experiments and the experiment results are discussed in detail. Differential CNN was evaluated with four different experiment sets and the results were compared with the classical method and other comparable techniques. DC-GCNN was tested on 10 different datasets and its performance was compared with the DCNN structure with the fully connected classifier. The proposed test results are clearly showed that the two proposed neural network structures provide a noticeable increase in the performance of the convolutional neural network.

5.1. Experimental Details

In this sub-section, the experimental setup is explained in details.

5.1.1. Differential CNN Experimental Details

Four different experiment sets were carried out to demonstrate the effectiveness of differential convolution technique. In the first experiment set, differential convolution operation was adapted to a traditional CNN structure and the rise of the classification performance was measured on ten different image datasets. CNN structure with 4 convolution layers was used in these experiments. Since some of the datasets were unbalanced, precision, recall, and F1-score values are also calculated in addition to the accuracy value. In this experiment set, 5-fold cross-validation was utilized.

In the second experiment set, differential convolution was adapted to AlexNet, a well-known CNN structure, and the performance increase was measured on ImageNet dataset with and without data augmentation operation, separately. The network was trained using mini-batches of size 128. The training process started with an initial learning rate, $1x10^{-2}$. It continued until the accuracy on the training data stops increasing, and after the learning rate is lowered. This

process was repeated multiple times. The learning rates used in this experiment were $1x10^{-2}$, $5x10^{-3}$, $1x10^{-3}$, $5x10^{-4}$, and $1x10^{-4}$, respectively.

In the third experiment set, the Differential CNN was compared with 5 convolutional neural networks in different structures. All compared structures had the same number of convolutional layers. The tested Differential CNN structure contains 5 convolutional layers. The numbers of filters in each convolution layer were 64, 64, 128, 128, and 128, respectively. The filters in the first two convolutional layers were $5x5$, while the other convolutional layers contained $3x3$ filters. First and fourth convolutional layers were followed by $2x2$ max-pooling layers. The first traditional CNN structure, CNN_No1, contains the same number of filters as the Differential CNN model. In other words, Differential CNN model is differential convolution equipped version of the CNN_No1 model. The second traditional CNN structure, CNN_No2, contained the same number of filters, while the filters were one unit larger for each axis. Since the differential CNN increases the receptive area by 1 unit in each axis, the second model was added to the comparison. The third traditional CNN structure, CNN_No3, had more filters to include the same number of parameters as the Differential CNN structure for each convolutional layer. Since the differential convolution produces additional feature maps, it increases the number of parameters of the following layer. The third model with more filters to contain exactly the same number of parameters in each layer was implemented to avoid additional parameter advantage. CNN_No3 had a total of 1280 traditional convolution filters, while the Differential CNN model contained only 512 differential convolution filters. The fourth structure was a Tiled CNN structure (Ngiam et al., 2010). The number of tiles, k, was defined as 2 for the Tiled CNN structure. Therefore, each of the two filter pairs in the structure would create only one feature map, Tiled CNN structure was given twice as many filters than differential CNN structure. The fifth structure was the Dilated CNN structure (Yu and Koltun, 2015) with the same number of filters. Dilated CNN structure utilized 2-dilated convolution. Decaying learning rate policy was used in the all

tests. While the initial learning rate was $1x10^{-2}$, the final learning rate was $1x10^{-4}$.

In the fourth experiment set differential convolution was adapted to NIN and VGGNet models and the performance increase was measured on CIFAR10 and CIFAR100 datasets. NIN model contains 3 convolutional layers applying 3 layered mlpconv (Lin et al., 2013). Differential convolution adaptation was applied to the first layer of each mlpconv filters. VGGNet model contains 13 convolutional layers applying 3x3 convolution operation. Batch normalization and Dropout techniques were applied to all models in this experiment set.

5.1.2. SoftMax Utilized Accumulative Backpropagation Experiment Details

Differential CNN with SoftMax utilized accumulative backpropagation was compared with a traditional CNN structure and the increase in the classification performance was measured on ten different image datasets. Precision, recall, and F1-score values are calculated for all experiments. In this experiment, 5-fold cross-validation was used.

5.1.3. DC-GCNN Experiment Details

DC-GCNN was compared with a traditional CNN structure containing exactly the same feature extractor structure and the rise of the classification performance was measured on ten different image datasets. Since some of the datasets were unbalanced, precision, recall, and F1-score values are calculated. In this experiment set, 5-fold cross-validation was utilized.

5.2. Results

In this subsection, all experimental results for Differential CNN and DC-GCNN are given in detail.

5.2.1. Differential CNN Results

In the first experiment set, a convolutional neural network structure composed of a feature extractor with four convolutional layers and three pooling layers and a fully connected classifier with two hidden layers is designed for the experiment. The input size of this artificial neural network was determined as 231x231. Therefore, all images in all datasets were resized into $231x231$ pixels. In the experiments, Differential convolution was used in the first three convolutional layers because the last convolutional layer's output is $1x1$ in size which is not compatible with Differential Convolution. The structure of the artificial neural network structure utilized in the experiments is shown in Table 5.1.

In the first experiment set, results showed that Differential Convolution utilized model provides better performance in any measures for all datasets. It achieved a relative performance increase up to 55.29% for accuracy. It had a relative performance boost up to 58.43% for precision. Relative performance increase for recall measure was up to 41.75%. It led a relative performance boost up to 56.43% for F1-score. The overall consideration of all the experiments showed that the average relative increase in accuracy, precision, recall, and F1 score were 20.91%, 21.26%, 16.19%, and 20.55%. Differential convolution exhibits a limited performance increase in some data sets, while this performance increase is very high for some other data sets. The reason behind this is the intensity of the directional changes in the pictures. While the increase in the Swedish leaf data set was around 1% for the performance measures, this increase was observed much higher in datasets such as KTH-ANIMALS or Abnormal Objects containing more complex images. Results of the first experiment were given in Table 5.2,

Table 5.3. Differential CNN with Accumulative BP results for each fold and Table 5.4.

In the second experiment set, differential convolution was adapted to AlexNet structure. AlexNet is a popular deep model trained with different policies in the literature. In the first experiment, two structures which one is differential

convolution adapted AlexNet and the other one is original AlexNet was trained following the same training policy and performances were measured on ImageNet dataset without data augmentation. This utilization increased the top-1 classification accuracy by 5.12% and top-5 classification accuracy by 4.40%. When data augmentation was used, the relative performance increase values were 5.3% and 4.75% for top1 and top5 accuracy values, respectively. Results of the second experiment set can be seen in Table 5.5.

In the third experiment set, differential CNN was compared with 5 different CNN structures. The differential CNN model achieved higher classification rates than all compared models. The differential CNN model showed better top1 and top5 performances on CIFAR-10 and CIFAR-100 datasets with 4.37% and 2.74% more accuracy values than the existing highest performing model. Relative performance increases were 7.72% and 4.09% for top1 and top5 accuracy values, respectively. Results of the third experiment set are given in Table 5.6. The training data error values and test accuracy rates of all models during the CIFAR-10 experiment are shown in Figure 5.1 and Figure 5.2.

In the fourth experiment set, differential convolution was adapted to NIN and VGGNet models. Top1 accuracy values of differential convolution adapted NIN (Differential NIN) model were 92.44.58% and 72.65% for CIFAR-10 and CIFAR-100 datasets, respectively. Differential NIN model had a relative increment on the top1 classification accuracy for CIFAR-10 and CIFAR-100 datasets by 3.20% and 6.01%, respectively. Top1 accuracy values of differential convolution adapted VGGNet (Differential VGGNet) model were 93.58% and 75.06% for CIFAR-10 and CIFAR-100 datasets, respectively. Relative increments in the top-1 classification accuracy of the Differential VGGNet model were 1.33% and 7.15% on CIFAR-10 and CIFAR-100 datasets, respectively. In addition, while the VGGNet model achieves more successful results than the NIN model, the Differential NIN model exceeds the performance of VGGNet model. The number of parameters of the VGGNet model is 15 million while the number of parameters

of the Differential NIN model is less than 1.4 million. This can be regarded as another indicator of the effectiveness of the differential convolution operation. Results of the fourth experiment set are given in Table 5.7. Test accuracy rates of all models during CIFAR-100 experiment are shown in Figure 5.3.

Table 5.1. Network structure of the first experiment set

Layer No	Network Structure
Input Layer	3x231x231 (RGB)
2	11x11 sized 128 filters with stride 4
3	2x2 max-pooling
4	5x5 sized 256 filters with no stride
5	3x3 max-pooling
6	3x3 sized 512 filters with no stride
7	2x2 max-pooling
8	3x3 sized 4096 filters with no stride
9	1024 neuron ReLu
10	128 neuron ReLu
11	Softmax Output

Table 5.2. Differential CNN with Accumulative BP and DCNN results comparison

Dataset Name	Differential CNN with Accumulative BP				DCNN			
	Acc	P	R	F1	Acc	P	R	F1
MSRC-v1	**78.76**	**78.76**	**79.73**	**78.97**	61.66	61.67	67.61	64.25
Oxford Flowers	**69.18**	**69.18**	**70.89**	**70.11**	58.46	58.46	60.25	59.34
MSRC-v2	**49.53**	**49.34**	**53.61**	**51.47**	34.12	33.99	38.25	35.20
Ponce Birds	**75.16**	**75.16**	**75.36**	**75.26**	66.66	66.67	69.81	67.32
Ponce Butterflies	**84.16**	**83.09**	**85.91**	**84.48**	75.21	74.21	78.07	75.55
Leeds Butterflies	**86.22**	**85.23**	**86.88**	**85.97**	77.64	77.43	78.89	78.08
KTH Animals	**45.36**	**45.12**	**49.67**	**46.68**	29.21	28.49	35.05	29.84
Swedish Leaf	**91.78**	**91.78**	**91.92**	**91.88**	90.66	90.67	91.07	90.87
JF-30	**92.12**	**91.98**	**95.10**	**93.26**	87.12	86.43	89.39	87.88
Abnormal Object	**43.36**	**43.12**	**54.31**	**45.12**	36.17	35.79	49.87	37.37

Table 5.3. Differential CNN with Accumulative BP results for each fold

Dataset Name	Measure	Fold1	Fold2	Fold3	Fold4	Fold5	Average
Ponce Birds	P	72.5	81.03	74	73.25	75	75.156
	R	72.52	81.62	74.18	71.92	76.58	75.364
	F1	72.51	81.37	74.06	72.51	75.84	75.258
Ponce Butterflies	P	80.44	90.79	80.93	80.68	82.6	83.088
	R	83.86	91.7	84	84.16	85.84	85.912
	F1	82.45	91.29	82.52	82.62	83.52	84.48
Leeds Butterflies	P	86.46	85.2	86.14	83.71	84.62	85.226
	R	87.77	86.82	86.22	86.54	87.04	86.878
	F1	87.01	85.76	86.18	85.24	85.64	85.966
Oxford Flowers	P	70.52	70.22	70.22	69.12	65.81	69.178
	R	69.88	72.1	72.27	72.93	67.28	70.892
	F1	70.12	71.25	71.21	71.42	66.54	70.108
Swedish leaf	P	92	92.23	90.67	92.44	91.55	91.778
	R	91.97	91.51	91.64	92.87	91.62	91.922
	F1	91.98	92.01	91.15	92.66	91.59	91.878
KTH-Animals	P	44.93	45.48	44.18	44.73	46.26	45.116
	R	48.57	49.53	48.71	51.69	49.89	49.678
	F1	45.82	47.52	45.1	47.03	47.92	46.678
Abnormal Objects	P	40.52	43.1	42.51	46.43	43.05	43.122
	R	61.97	51.21	51.71	53.73	52.93	54.31
	F1	45.17	44.35	43.65	47.31	45.1	45.116
MSRC-v1	P	81.25	77.72	78.74	79.02	77.08	78.762
	R	82.15	78.14	80.42	80.84	77.11	79.732
	F1	81.5	77.83	79.02	79.41	77.1	78.972
MSRC-v2	P	46.32	55.99	47.23	49.27	47.89	49.34
	R	51.65	59.79	53.27	49.97	53.35	53.606
	F1	49.82	57.83	50.26	49.22	50.22	51.47
JF-30	P	93.24	89.26	92.5	92.74	92.16	91.98
	R	96.98	93.84	94.01	95.69	94.97	95.098
	F1	94.61	91.51	93.09	93.68	93.42	93.262

Table 5.4. DCNN results for each fold

Dataset Name	Measure	Fold1	Fold2	Fold3	Fold4	Fold5	Average
Ponce Birds	P	61.67	63.33	65	72.5	70.83	66.666
	R	68.48	66.62	67.74	73.98	72.21	69.806
	F1	61.56	64.94	65.38	73.23	71.51	67.324
Ponce Butterflies	P	80.26	78.95	68.96	72.3	70.57	74.208
	R	81.13	81.04	74.11	76.92	77.16	78.072
	F1	80.63	79.98	71.22	74.13	71.81	75.554
Leeds Butterflies	P	83.09	79.55	77.3	74.99	72.23	77.432
	R	84.96	81.27	79.5	76.62	72.08	78.886
	F1	84.02	80.34	78.08	75.8	72.15	78.078
Oxford Flowers	P	51.1	56.25	56.25	58.46	58.46	56.104
	R	51.64	58.34	58.49	61.26	60.25	57.996
	F1	50.97	56.39	57.35	61	59.34	57.01
Swedish leaf	P	90.67	90.22	91.56	89.78	91.11	90.668
	R	91.13	90.72	91.89	90.26	91.36	91.072
	F1	90.9	90.47	91.72	90.02	91.24	90.87
KTH-Animals	P	27.92	25.79	32.28	27.22	29.23	28.488
	R	31.93	35.59	39.44	32.23	36.05	35.048
	F1	29.39	28.89	32.63	27.89	30.4	29.84
Abnormal Objects	P	41.12	31.21	36.05	35.18	35.41	35.794
	R	44.87	52.92	61.99	42.95	46.6	49.866
	F1	41.22	35.02	39.43	35.98	35.19	37.368
MSRC-v1	P	60.41	56.25	66.67	66.67	58.33	61.666
	R	65.69	63.67	77.92	71.2	59.55	67.606
	F1	62.95	59.73	71.85	68.86	57.85	64.248
MSRC-v2	P	33.34	33.86	28.56	32.17	42.05	33.996
	R	37.21	36.54	33.22	38.93	45.37	38.254
	F1	34.29	33.73	30.72	34.39	42.89	35.204
JF-30	P	89.29	85.98	88.95	87.07	80.88	86.434
	R	90.88	89.77	90.04	91.83	84.45	89.394
	F1	90.07	87.83	89.48	89.38	82.63	87.878

Table 5.5. Differential CNN second experiment set results

Dataset/Network	AlexNet		AlexNet with Diff. Convolution	
	Top1	Top5	Top1	Top5
ImageNet	55.12	78.23	**57.94**	**81.67**
ImageNet with data augmentation	62.50	83.00	**65.81**	**86.94**

Table 5.6. Differential CNN third experiment set results

	CIFAR-10		CIFAR-100	
	Top1	Top5	Top1	Top5
Differential CNN	**78.53**	**98.28**	**60.97**	**85.21**
CNN_No1	74.68	97.96	55.62	81.86
CNN_No2	74.92	98.04	55.92	82.13
CNN_No3	75.27	98.15	56.60	82.47
Tiled CNN	74.82	98.06	55.73	82.05
Dilated CNN	72.14	94.86	53.25	80.27

Table 5.7. Differential CNN fourth experiment set results

	CIFAR-10		CIFAR-100	
	Top1	Top5	Top1	Top5
Differential VGGNet	**93.58**	**99.87**	**75.06**	**93.12**
Differential NIN	92.44	99.83	72.65	92.56
VGGNet	92.35	99.81	70.05	88.92
NIN	89.57	99.72	68.53	87.75

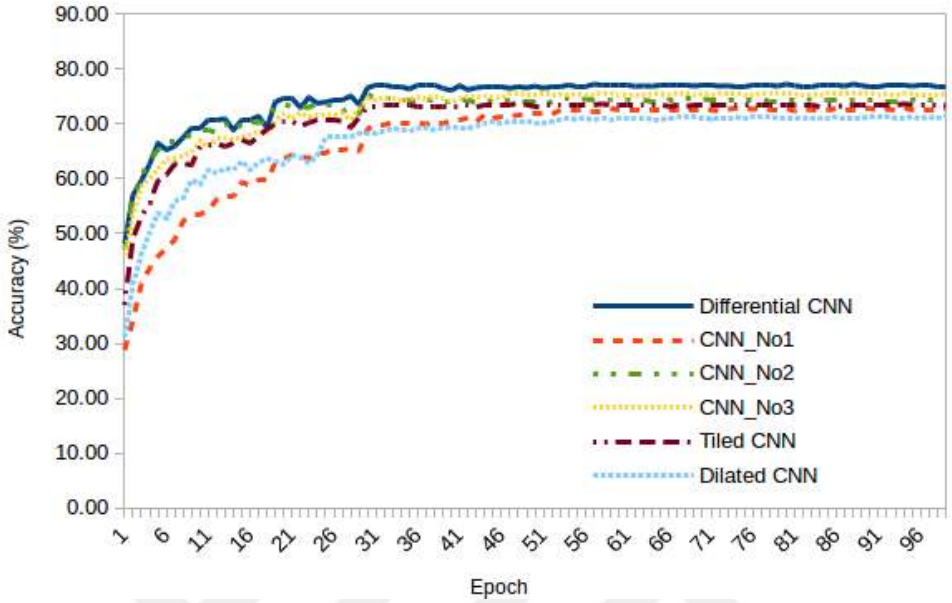

Figure 5.1. Third experiment set models' test accuracies for CIFAR-10

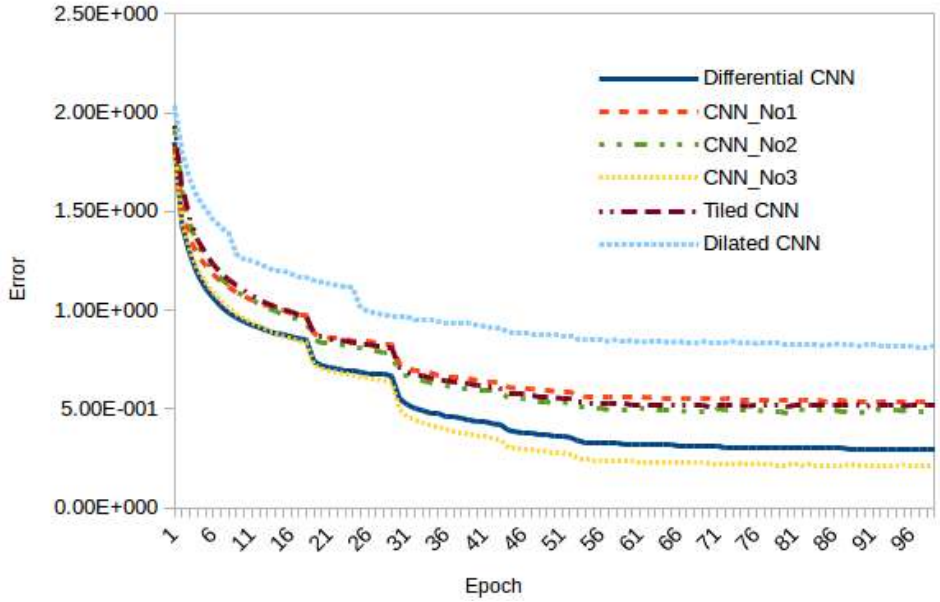

Figure 5.2. Third experiment set models' training error values for CIFAR-10

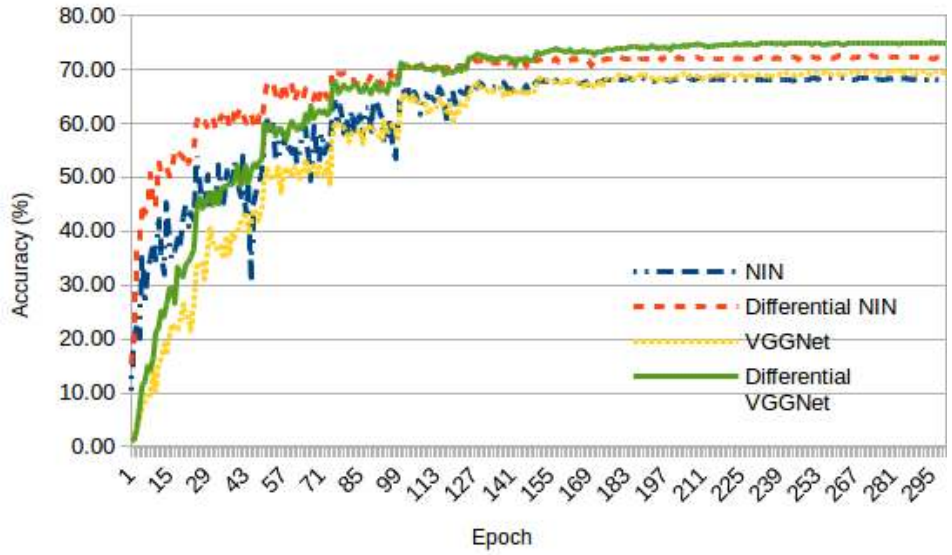

Figure 5.3. Fourth experiment set models' test accuracy values for CIFAR-100

5.2.2. Differential CNN with SoftMax utilized Backpropagation results

The accumulative backpropagation method increases the classification performance but it also has led to some problems. Collecting all the errors transmitted from neighbors with the error coming from the main feature map has increased the value of the error to be used in the training. Therefore, determining the learning rate has become more problematic. Furthermore, the ratio of the error coming from the main feature map in the total error is very small and the effect of this error is limited. In order to overcome these problems, neighborhood errors were limited by utilizing the SoftMax function. In this process, the SoftMax ratios were determined by using the activation differences computed in the forward pass of the training. This process provided a more accurate training for the larger structures and datasets, and it also improved the performance.

Differential CNN with SoftMax utilized backpropagation provided an additional relative performance boost up to 7.30%, 11.41% and 5.92% for precision, recall, and F1-score. Therefore, the total relative performance increase compared to the traditional method becomes up to 57.01%, 57.77% and 65.41% for

precision, recall, and F1-score. The average relative performance increase was 22.54%, 20.53% and 23.16% for precision, recall, and F1-score, respectively. As it can be seen, the SoftMax utilized backpropagation method has improved the performance of the Differential CNN significantly.

Table 5.8. Differential CNN with SoftMax utilized BP and DCNN results comparison

Dataset Name	**Differential CNN with Accumulative BP**			**DCNN**		
	P	**R**	**F1**	**P**	**R**	**F1**
MSRC-v1	**78.75**	**80.468**	**79.392**	61.666	67.606	64.248
Oxford Flowers	**68.088**	**70.936**	**69.242**	58.460	60.250	59.340
MSRC-v2	**52.944**	**59.728**	**54.518**	33.996	38.254	35.204
Ponce Birds	**73.952**	**75.128**	**74.612**	66.666	69.806	67.324
Ponce Butterflies	**82.442**	**85.774**	**83.38**	74.208	78.072	75.554
Leeds Butterflies	**83.808**	**87.852**	**85.062**	77.432	78.886	78.078
KTH Animals	**44.73**	**55.296**	**49.358**	28.488	35.048	29.840
Swedish Leaf	**97.156**	**97.38**	**97.266**	90.668	91.072	90.87
JF-30	**94.596**	**96.596**	**95.584**	86.434	89.394	87.878
Abnormal Object	**43.534**	**55.284**	**46.76**	35.794	49.866	37.368

Table 5.9. Differential CNN with SoftMax utilized BP results for each fold

Dataset Name	Measure	Fold1	Fold2	Fold3	Fold4	Fold5	Average
Ponce Birds	P	73.33	80	70	74.17	72.26	73.952
	R	74	81.81	70.8	76.23	72.8	75.128
	F1	73.66	80.9	70.4	75.18	72.92	74.612
Ponce Butterflies	P	78.94	85.12	83.02	83.69	81.44	82.442
	R	84.15	86.44	88.58	85.6	84.1	85.774
	F1	81.46	85.78	84.83	84.05	80.78	83.38
Leeds Butterflies	P	81.63	82.98	85.04	86.03	83.36	83.808
	R	87.65	87.78	89.09	91.69	83.05	87.852
	F1	83.99	83.42	87	88.24	82.66	85.062
Oxford Flowers	P	67.65	68.75	68.38	68.38	67.28	68.088
	R	71.28	70.55	71.12	72.05	69.68	70.936
	F1	69.42	69.64	69.14	69.67	68.34	69.242
Swedish leaf	P	97.78	94.67	96.89	98.22	98.22	97.156
	R	97.86	95.06	97.32	98.31	98.35	97.38
	F1	97.82	94.86	97.1	98.26	98.29	97.266
KTH-Animals	P	46.42	43.19	44.57	45.31	44.16	44.73
	R	53.57	55.12	54.21	55.41	58.17	55.296
	F1	49.71	48.43	48.92	49.7	50.03	49.358
Abnormal Objects	P	43.12	44.54	41.83	42.89	45.29	43.534
	R	55.32	54.55	52.89	56.21	57.45	55.284
	F1	47.87	46.13	41.6	48.68	49.52	46.76
MSRC-v1	P	81.25	81.25	70.83	85.42	75	78.75
	R	84.93	84.48	71.61	86.07	75.25	80.468
	F1	83.05	82.84	70.2	85.74	75.13	79.392
MSRC-v2	P	51.68	56.95	52.56	53.73	49.8	52.944
	R	52.67	65.3	55.7	65.97	59	59.728
	F1	52.08	59.87	53.3	55.3	52.04	54.518
JF-30	P	95.33	92.22	95.16	96.09	94.18	94.596
	R	97.64	92.99	97.61	98.28	96.46	96.596
	F1	96.47	92.6	96.37	97.17	95.31	95.584

5.2.3. DC-GCNN

Performance of DC-GCNN network was evaluated on ten different datasets and it was compared with DCNN with a fully connected classifier. Initially, all images in all datasets were resized into 231x231 pixels size which is compatible input size of the OverFeat pre-trained neural network structure. The same convolutional structures were used in DC-GCNN and MLP tests. Train rate was set to 0.01. MLP classifier has two hidden layers and an output layer with a number of neurons up to class numbers. The structures used in the experiments can be seen in Table 5.10.

Table 5.10. Network structures of DC-GCNN experiments

Layer No	MLP Structure	DC-GCNN Structure
Input Layer	3x231x231 (RGB)	3x231x231 (RGB)
2	11x11 sized 128 filters with stride 4	11x11 sized 128 filters with stride 4
3	2x2 max-pooling	2x2 max-pooling
4	5x5 sized 512 filters with no stride	5x5 sized 512 filters with no stride
5	2x2 max-pooling	2x2 max-pooling
6	3x3 sized 1024 filters with no stride	3x3 sized 1024 filters with no stride
7	2x2 max-pooling	2x2 max-pooling
8	5x5 sized 4096 filters with no stride	5x5 sized 4096 filters with no stride
9	1024 neuron ReLu	Pattern Later
10	128 neuron ReLu	Summation Layer
11	SoftMax Output	Normalization Layer
12		Output Layer

In previous studies, it was observed that the GCNN classifier achieved a better performance than the MLP structure over numeric data. DC-GCNN also includes the same advantage over the MLP, more successful classification performance values were observed with respect to DCNN. This performance increase was observed in ten different datasets.

According to the experiment results, it is shown that DC-GCNN provided better performance for all datasets in any evaluation metric used. It led a relative performance boost up to 44.45% for precision. Relative performance improvement in the recall was up to 39.69%. It also had a relative performance increase up to 42.57% for F1-score. The overall consideration of all experiment showed that precision, recall, and F1-score was increased by 10.9%, 18.48% and 15.21% on average, relatively. All results can be seen in Table 5.11. These results show the efficiency and importance of the proposed neural network structure.

Table 5.11. DC-GCNN and DCNN experiment results

Dataset Name	**DC-GCNN**			**DCNN**		
	P	**R**	**F1**	**P**	**R**	**F1**
MSRC-v1	**81.252**	**84.374**	**82.376**	56.248	60.4	57.78
Oxford Flowers	**59.514**	**78.854**	**66.803**	57.354	62.8	58.482
ETHZ Shapes	**52.406**	**67.084**	**57.388**	47.678	54.06	49.222
Ponce Birds	**72**	**79.286**	**74.962**	62.332	65.63	62.924
Ponce Butterflies	**71.076**	**80.302**	**75.318**	68.734	75.02	71.398
Leeds Butterflies	**75.756**	**86.252**	**81.646**	66.198	69.55	67.764
Tud Leaf	**88.932**	**90.202**	**89.558**	85.47	86.69	86.067
Swedish Leaf	**86.398**	**90.908**	**88.592**	85.066	86.44	85.416
Glassense Banknotes	**88.82**	**92.428**	**90.562**	86.234	84.83	85.48
Abnormal Object	**41.366**	**58.912**	**46.123**	37.993	46.96	38.467

Table 5.12. DC-GCNN results for each fold

Dataset Name	Measure	Fold1	Fold2	Fold3	Fold4	Fold5	Average
	P	85.42	85.42	79.17	75	81.25	81.252
	R	87.98	89.31	79.91	80.3	84.37	84.374
MSRC-v1	F1	86.68	86.39	78.47	77.56	82.78	82.376
	P	61.24	57.38	58.33	62.34	58.28	59.514
Oxford	R	80.88	76.38	78.65	79.32	79.04	78.854
Flowers	F1	67.45	64.19	64.88	69.38	68.12	66.804
	P	48.49	59.13	44.47	60.67	49.27	52.406
ETHZ	R	62.08	76.91	63.03	71.82	61.58	67.084
Shapes	F1	54.45	66.4	50.53	61.1	54.46	57.388
	P	67.5	75	75	73.33	69.17	72
Ponce	R	78.96	77.32	78.7	77.95	83.5	79.286
Birds	F1	72.78	76.1	76.38	74.91	74.64	74.962
	P	74.07	79.65	67.02	68.72	65.92	71.076
Ponce	R	85.22	85.46	68.12	82.38	80.33	80.302
Butterflies	F1	79.26	82.41	67.57	74.93	72.42	75.318
	P	82.41	72.4	76.72	72.56	74.69	75.756
Leeds	R	88.08	84.97	87.55	85.66	85	86.252
Butterflies	F1	85.15	84.53	81.47	78.57	78.51	81.646
	P	97.22	89.32	81.62	89.32	87.18	88.932
	R	97.44	89.1	83.97	90.3	90.2	90.202
Tud Leaf	F1	97.33	89.21	82.78	89.81	88.66	89.558
	P	88	85.78	88	85.77	84.44	86.398
Swedish	R	91.96	91.53	90.85	89.99	90.21	90.908
Leaf	F1	89.94	88.56	89.4	87.83	87.23	88.592
	P	87.17	93.51	92.04	74.88	96.5	88.82
Glassense	R	91.87	96.07	95.34	82.7	96.16	92.428
Banknotes	F1	89.45	94.77	93.66	78.6	96.33	90.562
	P	41.12	38.1	42.28	41.19	44.14	41.366
Abnormal	R	64.24	65.88	46.78	60.12	57.54	58.912
Object	F1	45.43	45.59	44.34	47.24	48.02	46.124

Table 5.13. DCNN results for each fold

Dataset Name	Measure	Fold1	Fold2	Fold3	Fold4	Fold5	Average
MSRC-v1	P	58.33	52.08	54.17	64.58	52.08	56.248
	R	62.36	58.07	58.82	65.13	57.63	60.402
	F1	60.28	54.91	56.4	64.86	52.45	57.78
Oxford Flowers	P	54.38	59.86	56.38	60.12	56.03	57.354
	R	64.56	60.56	61.24	64.98	62.66	62.8
	F1	56.34	59.32	57.87	60.83	58.05	58.482
ETHZ Shapes	P	39.01	54.45	53.13	43.74	48.06	47.678
	R	50.56	59.26	60.67	53.5	46.3	54.058
	F1	44.04	54.79	55.07	48.13	44.08	49.222
Ponce Birds	P	60.83	58.33	62.5	56.67	73.33	62.332
	R	61.99	68.13	63.71	59.66	74.66	65.63
	F1	61.41	60.63	62.76	56.23	73.59	62.924
Ponce Butterflies	P	65.75	65.41	73.53	66.78	72.2	68.734
	R	70.41	75.21	75.83	75.32	78.32	75.018
	F1	67.95	69.97	74.66	70.79	73.62	71.398
Leeds Butterflies	P	67.33	69.48	71.21	54.91	68.06	66.198
	R	70.96	71.36	74.91	58.38	72.16	69.554
	F1	69.1	70.06	73.02	56.59	70.05	67.764
Tud Leaf	P	91.67	78.42	75.85	92.09	89.32	85.47
	R	91.97	79.72	76.77	93.33	91.67	86.692
	F1	91.82	79.06	76.31	92.71	90.48	86.076
Swedish Leaf	P	84.44	82.67	84	86.22	88	85.066
	R	84.99	86.14	85.29	87.02	88.74	86.436
	F1	84.71	82.94	84.44	86.62	88.37	85.416
Glassense Banknotes	P	92.04	80.89	82.67	87.6	87.97	86.234
	R	91.91	79.84	82.85	84.11	85.44	84.83
	F1	91.97	80.36	82.56	85.82	86.69	85.48
Abnormal Object	P	45.37	36.69	31.93	40.14	35.84	37.994
	R	51.53	46.12	44.05	47.12	45.98	46.96
	F1	45.9	36.45	33.05	40.12	36.81	38.466

Another detail that should be considered in this study is that the properties produced with a larger convolutional structure have been taught to a smaller convolutional structure. The OverFeat structure is considerably larger than the structures used in the experiment. However, by adding the GCNN structure to the deep learning structure, the features produced with this large structure were similarly taught to a smaller structure, which significantly increased the performance.

6. CONCLUSSION

The popularity of deep learning structures increased rapidly with up-to-date large application areas. The most important factor behind this popularity is the high representation ability, high accuracy and robust performance of deep learning structures. These are supplied by the convolution operation that gives these features to a large extent. Convolution process is based on extracting feature maps by sliding the trainable filters on the image and calculating the activation values. These feature maps allow the network to recognize various patterns on the input image. The popularity of the deep learning structures led to the development of many new convolutional structures. On the other hand, many new classifier topologies are also proposed for performance improvement on deep learning structures.

In this thesis, two novel deep convolutional neural network structures are proposed. These two structures are focused on improving the performance of CNN through different ideas. Differential CNN improves the convolution method while DC-GCNN proposes an adaptation of a strong kernel-based classifier to CNN. Both structures required developments in the standard back-propagation algorithm. Specific training algorithms are developed and evaluated for each network.

Differential CNN is a novel deep learning structure applying a novel convolution technique named as differential convolution in its convolutional layers. Differential convolution calculates directional change amounts among a pixel and its neighbors to extract pattern orientation of them. As in mathematical differentiation, change amount calculations consider sequential changes by computing the difference among the activations of the pixels. Since images can be defined as a superposition of waves with different frequencies, differential convolution generates new feature maps representing signal changes. These maps lead detection of the big differences in the patterns which are valuable information for classification problems. Differential convolution provides these maps by

performing four additional convolution operations with predefined constant filters over the feature map generated by traditional convolution. Simple implementation of the differential convolution makes the technique adaptable for all the convolutional structures. Structures adapting differential convolution require an update on the error back-propagation algorithm. Therefore, an improved backpropagation algorithm named as accumulative back-propagation algorithm providing accumulative error over the neighborhood activations is proposed. In this method, neurons additionally acquire error information via neighborhood activations. Briefly, the proposed technique utilizes neighborhood activations in both forward and backward passes. While this operation represents signal changes in the forward pass, it also transmits error information in the backward pass.

Differential convolution technique was evaluated on four different experiment sets for performance evaluation. In the first experiment set, a traditional CNN structure was equipped with differential convolution and the performance increase was measured on 10 different image datasets. In the second experiment set performance increase of differential convolution adapted AlexNet model was measured. In the third experiment, Differential CNN was compared with different traditional CNN structures and the other convolution techniques named as Tiled CNN, Dilated CNN. In the last experiment set differential convolution was adapted to NIN and VGGNet models and performance increase was measured.

In the first experiment set, accuracy, precision, recall, and F1-measure were utilized as performance measures. Compared to the traditional method, Differential CNN had a relative performance increase on accuracy, precision, recall and F1 score up to 55.29%, 58.43%, 41.75%, and 56.43%, respectively. In the second experiment set differential convolution improved the top1 and top5 accuracies of AlexNet up to 5.30% and 4.75%, respectively. In the third experiment set, Differential CNN outperformed all compared convolutional structures. Differential CNN provided 4.33% and 7.72% higher relative top1 performance values than the closest performing model for CIFAR-10 and CIFAR-

100, respectively. In the last experiment set, differential convolution adapted VGGNet model obtained 93.58% and 75.06% top-1 accuracy values for CIFAR10 and CIFAR100 datasets, respectively. Top-1 accuracy values of differential convolution adapted NIN model for the same datasets were 92.44% and 72.65%. These results demonstrate the effectiveness and adaptability of the differential convolution technique. Another learning algorithm with SoftMax is also hybridized with differential CNN and it has been shown that the higher performance values can be achieved. SoftMax utilized backpropagation algorithm provided an additional relative performance boost up to 7.30%, 11.41% and 5.92% for precision, recall, and F1-score, respectively. The total relative performance increase compared to the traditional method is up to 57.01%, 57.77% and 65.41% for precision, recall, and F1-score, respectively.

Novelties and efficiencies of the proposed convolution technique can be summarized as follow. It preserves directional changes amount among pixel activations which are valuable information to extract pattern orientation, increases the number of feature maps without increasing the number of filters, and provides error feeds through neighborhood activations. Moreover, it is easily adaptable to all CNN structures for providing an increase in performance, and faster error minimization.

The second proposed structure, DC-GCNN, aims to improve the classification performance with GCNN, an efficient kernel-based classifier, adaptation. DC-GCNN is composed of convolutional layers of the deep learning structure and a kernel-based classifier. The idea behind this integration is to solve decision of efficient structure in fully connected part and provide high classification performance. After integration of GCNN in fully connected part of the DCNN, a new training method is also required. Development of this training algorithm and weight update approach with error backpropagation resulted to a new deep convolutional neural network. Proposed learning algorithm propagates error from the output neuron of the GCNN to output of the last convolutional layer

without updating any parameter in the GCNN and applies a gradient descent approach for the convolutional part. Hence, the new neural network decreases the number of parameters to be tuned in deep architecture. Moreover, the obtained DC-GCNN has non-complex structure determination policy advantages. While the convolutional part is determined intuitively, classifier part of DC-GCNN has a fixed structure with five layers which are input, pattern, summation, normalization, and output layers. The number of neurons in the input layer is determined by the number of features extracted with the convolutional part. Each training data is represented with a neuron in the pattern layer. The number of neurons in the summation and normalization layers are determined by the number of classes. Finally, the output layer has a single neuron that determines the winner.

Although there are a number of parameters that should be specified such as the number of layers, the number of neurons for a fully connected classifier of DCNN, the only parameter to be set is the training rate for DC-GCNN. On the other hand, the main disadvantage of the DC-GCNN is the computation complexity in the pattern layer for relatively big datasets. Adapting pattern layer size reduction methods used for GRNN into this structure may be a solution to this problem. The method used to train DC-GCNN may also be utilized for training other kernel-based networks such as GRNN and PNN in deep learning structures. While different convolutional structures may lead to different performance results, it has been explicitly shown that DC-GCNN performs better than MLP over the same convolutional structure on experimental datasets.

DC-GCNN was tested on 10 different datasets. All of them are image datasets. They are Ponce bird dataset, MSRC dataset, Oxford Flowers dataset, ETHZ dataset, Ponce butterfly dataset, Leeds butterfly dataset, Glassense banknotes dataset, Swedish leaf dataset, Caltech leaves dataset and abnormal object dataset. The same images and convolutional structures were used in DC-GCNN and DCNN tests. 5-fold cross validation is utilized in all tests. It is observed that DC-GCNN has better performance results than DCNN with traditional fully

connected methodology has the same convolutional structure. DC-GCNN provides a relative performance boost up to 44.45% for precision, 39.69% for recall and 42.57% for F1-score on application datasets. As a result of this, it can be said that the DC-GCNN has the ability to improve the performance of the convolutional neural network with a fixed classifier structure. Test results proved the improvement and efficiency of the DC-GCNN.

Finally, proposed two new and novel solutions improved the overall performance of deep convolutional neural networks. The differential convolution operation with an improved back-propagation learning algorithm and a new neural network structure named as DC-GCNN are introduced. These contributions are proposed to all artificial intelligence and machine learning science societies. Differential convolution technique is a hybridizable approach. It is adaptable to any convolutional neural network.

The proposed contributions are tested on different datasets with different characteristics. Test results proved the efficiency and importance of these dissertation contributions.

REFERENCES

Afkham, H. M., Targhi, A. T., Eklundh, J. O., Pronobis, A., 2008. Joint visual vocabulary for animal classification. In Pattern Recognition, 2008. ICPR 2008. 19th International Conference on (pp. 1-4). IEEE.

Beck, A., Teboulle, M., 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM journal on imaging sciences, 2(1), 183-202.

Chandrakumar, T., Kathirvel, R., 2016. Classifying diabetic retinopathy using deep learning architecture. Int J Eng Res Technol, 5(6), 19-24.

Chua, L. O., Yang, L., 1988. Cellular neural networks: Applications. IEEE Transactions on circuits and systems, 35(10), 1273-1290.

Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L., Schmidhuber, J., 2011. Flexible, high performance convolutional neural networks for image classification. In IJCAI Proceedings-International Joint Conference on Artificial Intelligence (Vol. 22, No. 1, p. 1237).

Ciresan, D., Meier, U., Schmidhuber, J., 2012. Multi-column deep neural networks for image classification. arXiv preprint arXiv:1202.2745.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248-255). Ieee.

Dozat, T., 2016. Incorporating nesterov momentum into adam.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., Zisserman, A., 2010. The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2), 303-338.

Ferrari, V., Jurie, F., Schmid, C., 2010. From images to shape models for object detection. International journal of computer vision, 87(3), 284-303.

Karpathy, A., 2015. Transfer learning and fine-tuning convolutional neural networks. CS321n Lecture Notes.

Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Krizhevsky, A., Hinton, G., 2009. Learning multiple layers of features from tiny images (Vol. 1, No. 4, p. 7). Technical report, University of Toronto.

Krizhevsky, A., Sutskever, I., Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Lazebnik, S., Schmid, C., Ponce, J., 2004. Semi-local affine parts for object recognition. In British Machine Vision Conference (BMVC'04) (pp. 779-788). The British Machine Vision Association (BMVA).

Lazebnik, S., Schmid, C., Ponce, J., 2005. A maximum entropy framework for part-based texture and object recognition. In Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on (Vol. 1, pp. 832-838). IEEE.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Vapnik, V., 1995. Learning algorithms for classification: A comparison on handwritten digit recognition. Neural networks: the statistical mechanics perspective, 261, 276.

Lin, M., Chen, Q., Yan, S., 2013. Network in network. arXiv preprint arXiv:1312.4400.

McCulloch, W. S., Pitts, W, 1943. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133.

Nesterov, Y., 1983. A method for unconstrained convex minimization problem with the rate of convergence O (1/k^ 2). In Doklady AN USSR (Vol. 269, pp. 543-547).

Ngiam, J., Chen, Z., Chia, D., Koh, P. W., Le, Q. V., Ng, A. Y., 2010. Tiled convolutional neural networks. In Advances in neural information processing systems (pp. 1279-1287).

Nielsen, M. A., 2015. Neural networks and deep learning (Vol. 25). USA: Determination press.

Nilsback, M. E., Zisserman, A., 2006. A visual vocabulary for flower classification. In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on (Vol. 2, pp. 1447-1454). IEEE.

Noh, H., Hong, S., Han, B., 2015.. Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE international conference on computer vision (pp. 1520-1528).

Ozyildirim, B. M., Avci, M., 2013. Generalized classifier neural network. Neural Networks, 39, 18-26.

Ozyildirim, B. M., Avci, M, 2014. Logarithmic learning for generalized classifier neural network. Neural Networks, 60, 133-140.

Ozyildirim, B. M., Avci, M., 2016. One pass learning for generalized classifier neural network. Neural Networks, 73, 70-76.

Pukelsheim, F., 1994. The three-sigma rule. The American Statistician, 48(2), 88-91.

Ravı, D., Wong, C., Deligianni, F., Berthelot, M., Andreu-Perez, J., Lo, B., & Yang, G. Z. (2017). Deep learning for health informatics. IEEE journal of biomedical and health informatics, 21(1), 4-21.

Ren, W., Yu, Y., Zhang, J., Huang, K., 2014. Learning convolutional nonlinear features for k nearest neighbour image classification. In Pattern Recognition (ICPR), 2014 22nd International Conference on (pp. 4358-4363). IEEE.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.

Rumelhart, D. E., Hinton, G. E., Williams, R. J, 1986. Learning representations by back-propagating errors. Nature, 323(6088), 533.

Saleh, B., Farhadi, A., Elgammal, A., 2013. Object-centric anomaly detection by attribute-based reasoning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 787-794).

Sankar, M., Batri, K., Parvathi, R., 2016. Earliest diabetic retinopathy classification using deep convolution neural networks. pdf. Int. J. Adv. Eng. Technol.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural networks, 61, 85-117.

Seeland, M., Rzanny, M., Alaqraa, N., Wäldchen, J., Mäder, P., 2017. Plant species classification using flower images—A comparative study of local feature representations. PloS one, 12(2), e0170629.

Sercu, T., Goel, V., 2016. Dense prediction on sequences with time-dilated convolutions for speech recognition. arXiv preprint arXiv:1611.09288.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y., 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.

Söderkvist, O., 2001. Computer vision classification of leaves from swedish trees.

Steinkraus, D., Buck, I., Simard, P. Y., 2005. Using GPUs for machine learning algorithms. In Eighth International Conference on Document Analysis and Recognition (ICDAR'05) (pp. 1115-1120). IEEE.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhouche, V., Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

Tang, Y., 2013. Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239.

Tieleman, T., Hinton, G., 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning, 4(2), 26-31.

Yu, F., Koltun, V., 2015. Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R., 2013. Regularization of neural networks using

Wang, J., Markert, K., Everingham, M., 2009. Learning models for object recognition from natural language descriptions.

Weber, M., 1999. Caltech leaves 1999. http://www.vision.caltech.edu/archive.html.

Winn, J., Criminisi, A., Minka, T., 2005. Object categorization by learned universal visual dictionary. In Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on (Vol. 2, pp. 1800-1807). IEEE.

Zeiler, M. D., 2012. ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.

CURRICULUM VITAE

Mehmet SARIGÜL was born in Hatay, in 1989. He completed his elementary education at General Refet Bele Primary Education School. He graduated from Harbiye High School, in 2005. He received the B.E and MSc degrees from Department of Computer Engineering, Çukurova University in 2010 and 2015, respectively. Since 2018, he has been working as a research assistant at Computer Engineering Department of Iskenderun Technical University in Hatay.