
GAZİ ÜNİVERSİTESİ
Gazi
1926

CLASSIFICATION OF LUNG CT IMAGES USING DEEP CONVOLUTIONAL NEURAL NETWORK

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
GAZİ UNIVERSITY

BY

Homay DANAEI MEHR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

DECEMBER 2017

The thesis study titled “CLASSIFICATION OF LUNG CT IMAGES USING DEEP CONVOLUTIONAL NEURAL NETWORK” is submitted by Homay DANAEI MEHR in partial fulfillment of the requirements for the degree of Master of Science in the Department of Computer Engineering, Gazi University by the following committee.

Supervisor: Asst. Prof. Dr. Hüseyin POLAT

Department of Computer Engineering, Gazi University

I certify that this thesis is a graduate thesis in terms of quality and content …..……..……..

Chairman: Asst. Prof. Dr. Javad RAHEBI

Department of Electric-Electronic Engineering, University of Turkish Aeronautical Association

I certify that this thesis is a graduate thesis in terms of quality and content …..……..……..

Member: Asst. Prof. Dr. Cemal KOÇAK

Department of Computer Engineering, Gazi University

I certify that this thesis is a graduate thesis in terms of quality and content …..……..……..

Date: 20/12/2017

I certify that this thesis, accepted by the committee, meets the requirements for being a Master of Science Thesis.

……..…….

Prof. Dr. Hadi GÖKÇEN

Dean of Graduate School of Natural and Applied Sciences

ETHICAL STATEMENT

I hereby declare that in this thesis study I prepared in accordance with thesis writing rules of Gazi University Graduate School of Natural and Applied Sciences;

- All data, information and documents presented in this thesis have been obtained within the scope of academic rules and ethical conduct,
- All information, documents, assessments and results have been presented in accordance with scientific ethical conduct and moral rules,
- All material used in this thesis that are not original to this work have been fully cited and referenced,
- No change has been made in the data used,
- The work presented in this thesis is original,

or else, I admit all loss of rights to be incurred against me.

Homay DANAEI MEHR

20/12/2017

AKCİĞER TOMOGRAFİ GÖRÜNTÜLERİNİN DERİN EVRİŞİMSEL SİNİR AĞLARI İLE SINIFLANDIRILMASI

(Yüksek Lisans Tezi)

Homay DANAEI MEHR

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Aralık 2017

ÖZET

Akciğer kanseri, kadınlarda ve erkeklerde dünyada en yaygın görülen kanser türlerinden biridir. Akciğer kanserinden ölüm oranı, diğer kanser türlerine oranla %70' in üzerindedir, bundan dolayı Amerikan Kanser Derneği tarafından 2016 yılında en agresif kanser türü olarak tanımlanmıştır. Akciğer kanserinin erken teşhisi hastaların hayatta kalma oranını artırabilir. Bunun için makine öğrenme teknikleri kullanılarak medikal görüntülerin sınıflandırılması, akciğer kanserinin erken teşhisinde işlem hızını artırarak doktorlara yardımcı olabilir. Geleneksel makine öğrenme teknikleri ile karşılaştırıldığında, derin öğrenme metotları, otomatik öznitelik çıkarma kabiliyetine sahip oldukları için daha etkin metotlardırlar. Bu tezde, Data Science Bowl ve Kaggle veri setindeki akciğer tomografi görüntüleri üzerinden akciğer kanserinin teşhisi için bir derin öğrenme metodu olarak evrişimsel sinir ağları kullanılmıştır. Akciğer tomografi görüntülerinin sağlıklı ve hastalıklı olarak sınıflandırılması için evrişimsel sinir ağlarının AlexNet ve GoogleNet mimarileri kullanılmıştır. AlexNet ve GoogleNet mimarileri ile sınıflandırmada sırasıyla %95.919 ve %96.360 doğruluk oranları elde edilmiştir. Evrişimsel sinir ağlarının bu iki mimarisi karşılaştırıldığında, GoogleNet mimarisinin, AlexNet mimarisine göre akciğer tomografi görüntülerinin sınıflandırılmasında daha yüksek doğruluk oranına ulaştığı görülmüştür. Sonuç olarak, derin öğrenme yöntemleri kullanılarak akciğer tomografi görüntülerinin sınıflandırılması ile, zorlamasız bir yöntemle akciğer kanserinin erken tanısına ilişkin daha fazla bilgi elde edilebileceği gösterilmiştir.

Bilim Kodu : 92431
Anahtar Kelimeler : Akciğer Kanserinin Teşhisi, Derin Öğrenme, Evrişimsel Sinir Ağları, Bilgisayarlı Tomografi
Sayfa Adedi : 94
Danışman : Yrd. Doç. Dr. Hüseyin POLAT

CLASSIFICATION OF LUNG CT IMAGES USING DEEP CONVOLUTIONAL NEURAL NETWORKS

(M. Sc. Thesis)

Homay DANAEI MEHR

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

December 2017

ABSTRACT

Lung cancer is one of the mostly observed cancer types in both men and women worldwide. Mortality rate of over 70% put the lung cancer among the most aggressive cancers list in 2016 by American Cancer Society. However, early diagnosis of lung cancer would increase survival rate of patients. To this, Machine learning techniques for classification of medical images is used to assist physicians in order to accelerate diagnosis process. In comparison with shallow machine learning techniques, deep learning methods are more effective as they are capable of extracting features automatically. In this thesis, Convolutional Neural Network is used as one of the deep learning methods to diagnose lung cancer over the lung CT images of Data Science Bowl and Kaggle dataset. AlexNet and GoogleNet are two architectures of Convolutional Neural Network which are used to classify lung CT images as benign and malignant. AlexNet and GoogleNet architectures achieved 95.919% and 96.360% accuracy rates respectively in classification of lung CT images. By comparison two architectures of Convolutional Neural Networks, it is demonstrated that GoogleNet architecture achieved higher accuracy rate than AlexNet architecture in classification of lung CT scan images. In conclusion, it has been proved that with the classification of the lung CT scan images using deep learning methods, more information concerning early diagnosis of lung cancer may be obtained with a noninvasive method.

Science Code : 92431
Key Words : Lung Cancer Diagnosis, Deep Learning, Convolutional Neural Networks, Computed Tomography
Page Number : 94
Supervisor : Asst. Prof. Dr. Hüseyin POLAT

ACKNOWLEDGEMENTS

Firstly, I am ever grateful to God, the Creator, and the Guardian, and to whom I owe my very existence.

Foremost, I would like to express my sincere gratitude to my advisor Assist. Prof. Dr. Hüseyin POLAT for the continuous support of my thesis, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my study.

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

I would like to thank my thesis committee for providing crucial practical guidance.

My special thanks go to my beloved mother and father, the healer of my mental twinges and a cure-all for all tortures of mind.

Finally, I am indebted to my special person in my life, Farid for his valuable help and moral support.

CONTENTS

Page

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

Figure **Page**

Figure **Page**

SYMBOLS AND ABBREVIATIONS

The symbols and abbreviations used in this thesis are presented below along with explanations.

Abbreviations	Description
ANN	Artificial Neural Network
BVLC	Berkeley Vision and Learning Center
CADe	Computer-Aided Detection
CADx	Computer-Aided Diagnosis
CNN	Convolutional Neural Network
CT	Computed Tomography
Cuda	Computer unified device architecture
DBN	Deep Belief Networks
DICOM	Digital Imaging and Communications in Medicine
Digits	Deep Learning GPU Training System
DLCS	Danish Lung Cancer Screening
DNN	Deep Neural Network
DSN	Deep Stacking Networks
FDA	Food and Drug Administration
FN	False Negative
FP	False Positives
IDRI	Image Database Resource Initiative
ILD	Interstitial Lung Disease
ILSVRC	ImageNet Large Scale Visual Recognition challenge
JSRT	Japanese Society of Radiological Technology
KNN	K Nearest Neighborhood
LBP	Local Binary Pattern
LIDC	Lung Image Database Consortium
LSTM	Long Short Term Memory
LUNA	Lung Nodule Analysis
MID	Multicentric Italian Lung Detection
MLP	Multilayer Perceptron

Abbreviations	Description
MSE	Mean Squared Error
MTANN	Massive Training Artificial Neural Network
NSCLC	Non-Small Cell Lung Cancer
PNG	Portable Network Graphics
RBM	Restricted Boltzmann Machine
Relu	Rectified linear unit
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristic
SAE	Stacked Auto Encoder
SCLC	Small Cell Lung Cancer
SDAE	Stacked Denoising Auto Encoder
SGD	Stochastic gradient descent
SIFT	Scale Invariant Feature Transform
SVM 3D matrix	SVM based on three dimensional matrixes
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

1. INTRODUCTION

Cancer, an emotive subject of our age which millions of people worldwide struggling with and there is still no final cure for it. However, taking it under control by early detection can be a way to at least increase the survival rate. There are hundreds of different types of cancer which were observed up to date, and most of which are deadly. Based on the location and the type of tumor whether it is benign or malignant a physician can make decision for treatment. However, diagnosis of tumor type is a laborious procedure and in some cases, it is needed to get the patient under surgical operation and remove part of the tumor and find out in the laboratory whether it is malignant or benign.

Coming after prostate and breast cancer, lung cancer is the second mostly observed cancer type in both men and women [1]. Annually, over than 1.2 million people are struggling with this disease and most of which are losing their lives and this makes lung cancer the deadliest cancer among other types [2].

Basically, the body keeps the control of cell growth mechanism under control, in this case when new cells are required this system divides cells to produce new one but as much as it's required. Any disturbance in this system can cause dramatic effects such as uncontrolled multiplication of cells that can eventually cause the formation of a mass known as a tumor. Spreading out the cancer is called metastasis [3].

The death toll of over 70%, American Cancer Society put the lung cancer among the most aggressive cancers list in 2016 [4]. Lung cancer was observed in roughly 42 000 people in 2010 which means 115 people every day. Occurring cancer has been linked to use of tobacco products and smoking is known as the main factor leading to lung cancer, so far. Beside this, roughly 10% of which these cancers are diagnosed are non-smokers [5]. In comparison with a lifetime nonsmoker, a lifetime smoker, 20 to 30 times more, runs the risk of developing lung cancer. The tendency to smoke is falling down in developed countries like the United States and China, whereas smoking takes tens of millions of new victims annually around the world [6]. Global industrialization, hence releasing harmful substances as well as gases to the environment - most of which have carcinogenic effects - and exposure to these elements develops the risk of lung cancer, as well [7].

Five-year life expectancy is common between 65% of patients of Non-Small Cell Lung Cancer (NSCLC) but if the disease is detected in early stages, whereas long run life expectancy can be dramatically decreased to 1% for those who have metastasis [8]. Though, the probability of survival will be increased to 49% if the cancer is detected in the early stage when it is limited to the lung and has not spread out to the lymph [9]. Tumors are divided into two main categories and those are: benign and malignant. Benign refers to the tumors which are not dangerous as cancerous tumors and/or without the feature of spreading out. Hence, these tumors have less detrimental effects as they can be get under control and/or sometimes can be removed with less chance of getting back. Beside this, malignant refers to the types which are growing intensively and/or have the potential of seizure and catastrophically damaging tissues as well as the potential of passing through the bloodstream or lymphatic system and spreading out of the body in a very short period of time [10]. Small Cell Lung Cancer (SCLC) and NSCLC are two main lung cancer types. SCLC and NSCLC lung cancer types are given by Figure 1.1.

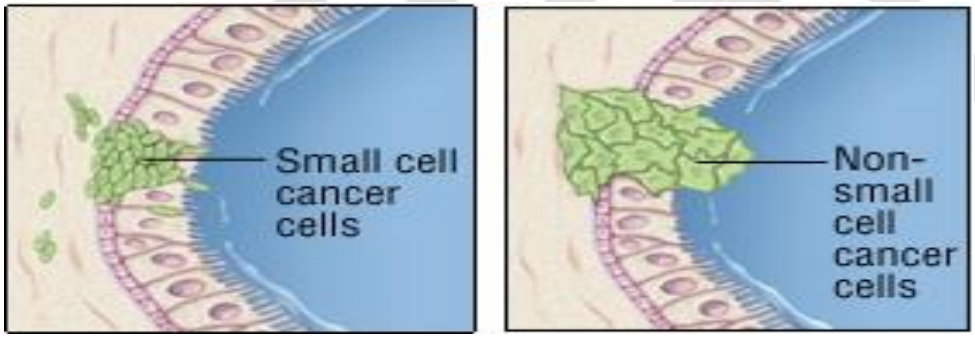

Figure 1.1. Small cell cancer cells (SCLC) and non-small cell cancer cells (NSCLC)

Deriving from epithelial and neuroendocrine cells, SCLC is extremely aggressive as well as a hard-prognosis neuroendocrine tumor, involving small tumor cells and intensively linked with smoking. This type of lung cancer is hard to get under control due to its fast spreading out characteristic. Roughly, 25% of diagnosed lung cancers are SCLC [11]. 75% of all diagnosed lung cancers belong to NSCLC which itself divides into three main categories, involving; Adeno Carcinoma, Squamous Cell Carcinoma, and Large cell carcinoma [12] When it comes to the stage, scientists divided lung cancer into four stages (I to IV), depending on tumor size [13].

Depend on physician's decision there are some diagnosis methods of lung cancer:

1) Imaging methods:

- Since many years ago to diagnose diseases X-ray is the efficient method it is useful in lung cancer diagnosis. Actually, X-rays include powerful radiation and waves which are very short in length than normal light [14].

- Computed Tomography (CT) is a computer aid method in which assembled image data by a special X-ray apparatus which are taken from different sides of the body is processed by computer to show a cross-section of body organs through which computer make an interception of CT scans of the body and make radiologists able to diagnose cancer more easily. Resultantly, the physician can make a precise decision of presence of a tumor as well as its size and precise location and the possibility of extension to the adjacent tissues [15].

- Magnetic resonance imaging or so-called MRI is an advanced imaging method in which magnetic field and radio waves are applied together to make clear and accurate images of the internal body parts [16].

- Positron Emission Tomography (PET) is another computerized imaging method in which computer make an image of chemical changes occur in tissues. In this method, an injection of a radioactive sugar takes place that makes the radiologist able to find the location of the cancerous tissue since these tissues have more tendency to take sugar than the other substances [17].

2) A cough along with sputum sometimes is a perilous sign. Going through the sputum under microscope sometimes can unearth the presence of lung cancer cells [18].

3) Sometimes cancer suspicious tissues are removed for sampling to examine carefully in the laboratory through a procedure so-called biopsy [19].

Computed tomography scan cancer imaging

Lung cancer detection has become easier after emerging CT scanners as for decades X-ray images were the most effective way of detecting lung cancer. At first, any nodule found on CT scan was perceived as malignant unless no growth was recorded after two years of monitoring. Since a large portion of the detected nodules under CT scan was malignant, this method was applied to reduce the risk of tardy intervention. However, there was something similar in all nodules and that was the diameter of those which were larger than 5 mm - most of which had the diameter between 1 to 3 cm [20].

The disadvantage of CT scan is the exposure of the patients to the high dose of radiation which is increased the cancer rate and consequently increases the demanding to retreatment which is risky. However, being painless, quick and accessible in many treatment centers as well as the accuracy of this method, makes CT scan method more preferable to both patients and physicians [15]. Another advantage of CT scan in comparison with chest radiographs is CT scan can make a clear image of those lung nodules which are slow growing and as small as 1-2 mm in diameter which cannot visualize on chest radiographs [20].

Providing better lucidity by lowering the noise through imaging, make CT images the advantage of clearness and therefore the precise diagnosis of lung cancer in comparison with X-ray and MRI images [21]. Reportedly, early stage detect of lung cancer is possible in 85% of the cases through CT screening. Hence, the survival rate can be increased up to 10 years in 88% of lung cancer detected patients in stage I [22]. Compared to chest X-ray, low dose helical CT screening of lung cancer in patients can decrease the death rate by 20% [23]. Image of the Lung cancer in the right lobe of patient’s lung achieved by CT scan is shown in Figure 1.2.

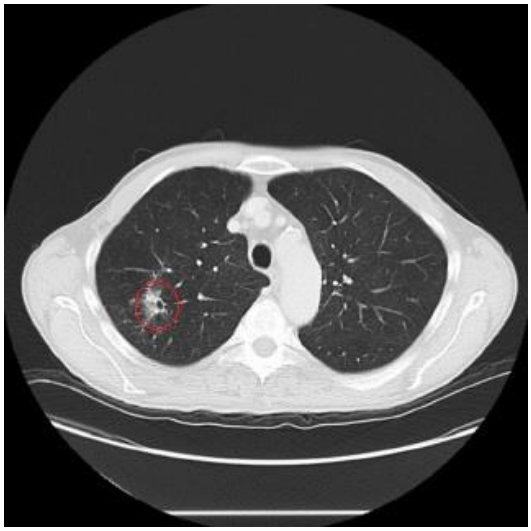

Figure 1.2. Lung cancer in right lobe of patient's lung achieved by CT scan [24]

Computer aided diagnosis

In order to early detection of different diseases, especially various type of cancers medical imaging assists physicians to diagnose diseases before it is too late [25]. Nowadays, atomization has made computers to hand humankind in every dimension of life. Hence, using computers assistance in the medical workflow as well as an inevitable subject of our age which helps physicians to make precise decisions as well as rising up the accuracy of the diagnose. Computer assistance itself can be divided into two main categories which are: Computer-Aided Detection (CADe) and Computer-Aided Diagnosis (CADx) which both are known as CAD. In oncological subjects, the aim of CADe is only tumor detection, whereas the main goal of the CADx is to differentiate between malignant and benign tumors.

Generally, in CAD diagnosis systems, in order to classify tumors different image processing techniques are applied on images and features are extracted [26]. Machine learning methods make a model of training for medical images and they are able to handle all objects of data in computer assistance structure. In recent years, deep learning methods are more successful than shallow machine learning methods in CAD. Deep learning methods are independent of handcraft or any other feature extraction methods. Furthermore, deep learning methods are able to extract and select the features and then classify dataset in its architecture. Therefore, deep learning methods in CAD systems help physicians in diagnosis of cancer by improving the accuracy of diagnosis and cost efficiency in a short time [27].

Objective of thesis

The major purpose of this thesis is diagnosis of lung cancer in early stages over the CT scan images. In order to diagnose of lung cancer, deep learning is used as one of the machine learning techniques. In this regard in order to classify the cancerous and non-cancerous CT scan images, one of the deep learning architectures named Convolutional Neural Network (CNN) has been used. AlexNet and GoogleNet which are two architectures of CNN have been evaluated in lung cancer diagnosis.

Related works

In shallow machine learning methods and CAD techniques for classification of images before applying image processing methods, preprocessing of images are the most important issue [28]. Some of the traditional detection and classification methods are reported below.

N. Niki et al., have used K-means clustering algorithm to detect and clustering the lung cancer nodules on CT Scan images. To classification of malignant and benign nodules, they used the linear discriminant algorithm. The proposed CAD system was showed high performance of Receiver Operating Characteristic (ROC) than the physicians' test on the same CT Scan images [29].

Y. Matsuki et al., have used Artificial Neural Network (ANN) to diagnose normal and abnormal lung tumors on CT scan images. A team of radiologists diagnosed benign and malignant cancer cells without using CADx system. By comparison with the results which radiologists diagnosed, ANN algorithm showed considerable performance. The area under the ROC curve (A_z) of ANN algorithm and the radiologists diagnosis were 0.951 and 0.831 respectively. Applying ANN algorithm by radiologists caused the A_z value raised up to 0.959 [30].

M.G.Penedo et al., have applied two ANN for detection and classification of lung nodules. The used dataset was CT scan images of the chest that collected by the hospital of Santiago de Compostela. After preprocessing and extraction of suspected areas of chest images they used two Multilayer Perceptron (MLP) neural network. In first step, MLP was used for detection of cancerous nodules and in the second step, another MLP was utilized for

classification. Results showed that sensitivity of two ANN was in the range of 89% to 96% and False Positive (FP) of two networks were between 5 and 7 [31].

A. Teramoto and H.Fujita, have proposed cylindrical nodule-enhancement filter method to detect the lung cancer nodules of CT scan images. The used dataset was Lung Image Database Consortium (LIDC). Their aim of using proposed segmentation and detection method was to increase the speed of nodule detection. Support Vector Machine (SVM) algorithm was used to classify the detected nodules. Results showed that 80% of nodules were detected by the proposed method. Detection speed was compared to other similar works in the literature and its speed was higher than the other methods [32].

M. Kakar and D.R.Olsen, have used SVM classification algorithm to recognize the lung cancer lesions on CT scan images which collected by Radium Hospitalet Medical Center of Oslo, Norway. First Gabor filter method and Fuzzy C-Means clustering method were used for feature extraction and segmentation. Cluster centers optimized by using the Genetic algorithm and eventually SVM classifier were used for classification of the region of lungs and lesions. Results showed that SVM classification algorithm achieved 89.48% sensitivity in differentiating the regions. Moreover, their used method for detection of left lung, right lung and lesions achieved the accuracy rates of 94.06%, 94.32% and 89.04% [33].

H. Chen et al., have applied ANN and logistic regression to discriminate lung cancer nodules on CT scan images. By comparison, two algorithms results showed that ANN algorithm obtained better performance than logistic regression analyses. By considering the mean value and standard error, the accuracy rate of ANN and logistic regression were $90.0 \pm 2.0\%$ and $86.9 \pm 1.6\%$ respectively. Likewise, the value of the area under the ROC curve for ANN was higher than logistic regression. 0.955 ± 0.015 and 0.929 ± 0.017 were the value of the area under the ROC curve for ANN and logistic regression for differentiating benign and malignant nodules [34].

Q. Wang et al., Have applied five CAD methods based on SVM and ANN to discriminate lung cancer nodules. CT scan images of Jilin Tumor Hospital were used. The five proposed methods were SVM based on three-dimensional matrixes (SVM 3D matrix), SVM with unfolding three-dimensional matrix, SVM by region of interest of nodules, ANN based on the region of interest of nodules and SVM classifier. Results showed that SVM 3D matrix

algorithm achieved the highest performance in classification of nodules. The value of True Positive (TP) and the area under the Roc curve of SVM 3D matrix were 0.995 and 98.2% respectively [35].

A. Kulkarni and A.Panditrao, have proposed a method based on images processing techniques and SVM classifier to discriminate the stages of lung cancer. LIDC, CT scan of chest dataset was used. For preprocessing CT scan images median filtering method was used to eliminate noises and Gabor filter method was used for image enhancement. Watershed method used for segmentation of CT scan images. Area, perimeter, and eccentricity were extracted as features for identification of cancer stages. After preprocessing images SVM classification algorithm was used to differentiate the nodules and detection of cancer stages. Results showed that the proposed algorithm could detect stages of lung cancer by the size of extracted features [36].

H. Arimura et al., have used a CAD method to detect the benign and malignant lung cancer nodules of CT scan images. Low-dos CT scan images of lung nodules, which collected in Nagano, Japan named LDCT dataset was used. To preprocess the images first, linear discriminant analysis was used for segmentation and filtering methods were applied as well. After determining the region of nodules Massive Training Artificial Neural Network (MTANN) and linear discriminant analysis algorithms were used for classification of nodules. Results showed that MTANN outperformed linear discriminant analysis algorithm in reduction of FP. The sensitivity of detecting benign and malignant nodules was 83% and 84% respectively [37].

K. Suzuki et al., have applied Multi MTANN algorithm to distinguish the benign and malignant nodules of Lung CT scan images. Their purpose of using Multi MTANN was decreasing the FP. Low-dos CT scan images of Nagano, Japan LDCT dataset was used. Results show that Multi MTANN algorithm reduced the FP significantly (27.4 to 4.8) and it achieved 80.3% sensitivity [38].

C. Jacobs et al., have applied K Nearest Neighborhood (KNN) algorithm to classify the solid, non-solid and part solid of lung cancer nodules. Dutch Belgian Nelson CT scan images were used. To preprocess the images, segmentation methods used for features extraction. They reported that performance of their used method and diagnosis of experts were almost similar.

Cohen's kappa coefficient was between 0.54 and 0.72 and the Cohen's kappa coefficient value of experts' diagnosis was between 0.56 and 0.81 [39].

T.W. Way et al., have compared SVM with Linear discriminant analysis for classification of CT scan of lung nodules. After using segmentation methods and K-means algorithm for clustering malignant and benign nodules classification algorithms were applied. The linear discriminant analysis algorithm by using stepwise feature selection method improved the value of the area under the ROC curve from 0.821 ± 0.026 to 0.857 ± 0.023. Furthermore, results showed that the value of the area under the ROC curve of SVM classifier was higher than the value of the area under the ROC curve of linear discriminant analysis algorithm [40].

Since large image datasets of lung cancers are rare and deep learning methods are novel in diagnosis of diseases, there are few researches in diagnosis of lung cancer [41]. Subsequently, so far the methods which are based on deep learning methods are described below.

B.V. Ginneken et al., have compared Overfeat CNN and Food and Drug Administration (FDA) as a commercial method of CAD for detection of lung cancer nodules. CT scan images of LIDC was used for detection of nodules. Features of lung nodules were extracted by Overfeat CNN and SVM algorithm was used for classification of nodules. Furthermore, nodules were detected by commercial CAD system. Results showed that each method could detect nodules by over 70% sensitivity [42].

M. Anthimopoulos et al., have proposed CNN to classify and characterize different lung tissues of lung diseases. CT scan images of University Hospital of Geneva and Bern University Hospital were used as Interstitial Lung Disease (ILD) datasets. Their proposed CNN contained five convolutional layers, one pooling layer, and three fully connected layers. The proposed algorithm was compared by other CNN architecture e.g. LeNet, AlexNet and VGG Net. Results showed that the proposed CNN for classification and detection of tissues was superior compared to the other algorithms. The proposed CNN achieved 85.61% accuracy rate [43].

R. Gruetzemacher and A. Gupta, have used Deep Neural Network (DNN) for classification of lung cancer nodules. CT scan images of LIDC and Image Database Resource Initiative (IDRI) were used as a dataset. Four different topologies with different numbers of convolutional layers were compared. Results demonstrated that accuracy rate of all used methods by different convolutional layers were close to each other and network by five convolutional layers achieved the highest accuracy rate (82.10%) [44].

W. Sun et al., have compared three algorithms of deep learning and traditional CAD system to diagnose lung cancer nodules on CT scan images. They used LIDC and IDRI datasets for diagnosis of lung cancer. DBN, CNN and Stacked Denoising Auto Encoder (SDAE) were used as three algorithms of deep learning. Results of accuracy rates demonstrated that CNN and DBN were superior compared to the SDAE and traditional CAD methods Furthermore, DBN achieved the highest accuracy rate of nodules classification. (81.19%) [41].

F. Ciompi et al., have applied Multi-scale CNN with multi-stream architecture as a deep learning method for classification of lung cancer nodules on CT scan images. In order to characterize lung cancer nodules, Multicentric Italian Lung Detection (MID) and Danish Lung Cancer Screening (DLCS) datasets were used. Automatic nodules classification in six types was done without using any segmentation methods. All scales of CNN were combined in a fully connected layer of CNN. Results of proposed multi-scale CNN were compared with radiologists' diagnosis and the average accuracy rate of CNN (69.6%) is close to average accuracy rate of radiologists (72.9%). Moreover, accuracy of CNN with three scale was compared with SVM based pixel intensity of patches and SVM based unsupervised learning of features. Results show that CNN with three scale achieved higher accuracy rate (79.5%) than other two SVM based methods [45].

K.L. Hua et al., have proposed two deep learning algorithms named CNN and Deep Belief Networks (DBN) to classify lung cancer nodules. CT scan images of LIDC dataset were used. Two proposed deep learning algorithms were compared with two algorithms of feature descriptors. The first method was the combination of Scale Invariant Feature Transform (SIFT) and Local Binary Pattern (LBP) and the second one was fractal analysis. For two methods of SIFT+LBP and fractal analysis SVM and KNN classifiers were utilized. Experimental results demonstrated that performance of two proposed deep learning methods

were higher than SIFT+LBP and fractal analysis in classification of nodules (sensitivity value of 73.4% and 73.3% for DBN and CNN) [28].

Q. Li et al., have proposed a CNN algorithm with a single convolutional layer for classification of patches on high resolution computed tomography (HRCT) images. ILD lung dataset was used for this purpose. Furthermore, combination of SVM classifier with three feature extraction methods (i.e. SIFT, LBP, Restricted Boltzmann Machine (RBM)) were used to extract features and classify images. Their proposed CNN was compared with the combination of three feature extraction methods and SVM classifier. Results showed that their proposed CNN achieved higher Sensitivity or Recall (about 0.88) and Precision values (about 0.93) than the other methods [46].

W. Shen et al., have proposed multi scale CNN for classification of malignant and benign nodules of lung. CT scan images of LIDC and IDRI datasets were used. SVM and Random Forest were used as classification algorithms of CNN. Their proposed CNN algorithm with Random Forest classifier achieved 86.84% accuracy rate in classification of lung nodules without using any segmentation methods [47].

P. Rao et al., have proposed CanNet as CNN model to classify lung CT scan images of LIDC dataset. Their proposed CanNet contained two convolutional layer, one max pooling and one fully connected layer. In comparison with traditional ANN and LeNet their proposed CanNet model achieved the highest accuracy rate in classification of lung CT scan images. Accuracy rate of each LeNet, ANN and CanNet was 56%, 72.5% and 76% respectively [48].

W. Alakwaa et al., have used 3D CNN to classify lung CT scan images of Data Science Bowl and Kaggle. For nodules detection they have used U-Net as an architecture of CNN in biomedical field on Lung Nodule Analysis (LUNA) dataset. LUNA was the assistant dataset to train network for nodule detection in Kaggle dataset. Results showed that CNN by using U-Net architecture in classification of lung CT scan images achieved 86.6% accuracy rate. FP rate and False Negetive (FN) rate of CNN were 11.9% and 14.7% respectively [49].

Q. Song et al., have compared performance of DNN, CNN and Stacked Auto Encoder (SAE) algorithms in classification of CT scan images of LIDC-IDRI datasets. Results showed that CNN algorithm surpassed other two algorithms in classification of lung CT scan images.

CNN, DNN and SAE achieved 84.15%, 82.37% and 82.59% accuracy rate. CNN and SAE achieved the same sensitivity (83.96%) and DNN achieved 80.66% sensitivity [50].

N. Bondfale and S. Banait, have used CNN for classification of ILD dataset of lung CT scan images. They reported results of CNN for classification of healthy, ground-glass opacity, micro nodules, reticulation, honeycombing, consolidation and ground-glass opacity with reticulation seven classes of ILDs were favorable [51].

2. MATERIAL AND METHODS

In this thesis, CNN which is one of the most popular algorithms of deep learning is used for classification of lung CT scan images. This section covers ANN algorithm which is the foundation of CNN algorithm. After the description of ANN, the background of ANN, perceptron concept, activation functions, gradient descent algorithm and backpropagation training algorithm are described. Other sections are assigned to describe the deep learning.

2.1. Artificial Neural Networks

ANN inspired the system of biological nervous and process information by interconnected neurons [52]. Actually billions of variant neurons by different lengths in all part of the human body forming the nervous system [53]. Biological neuron's system is shown by Figure 2.1.

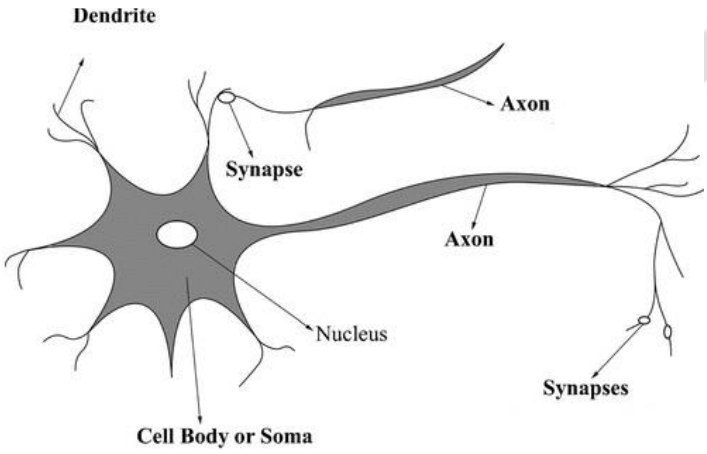

Figure 2.1. The structure of a biological neuron [54]

In the biological model of a neuron, the nucleus is in the middle of the cell body (soma). The receiver of signals called dendrites which are connected to the cell body. The longest part of the cell body with various branches is axon. Axon is the connection point of one neuron to other neurons by the connection which is called synaptic junctions and it passes signals to the dendrites and cell body of the other neuron. Through the chemical signal transferring between two cells of a synapse, the sender part releases special kind of matters. Consequently, the electrical potential is increased or decreased on the receiver side. In order to fire the cell, electrical potential should achieve its threshold amount and axon receives signals by constant and periodic power. In this regard, neurons send signals to dendrites and

neural activities are sent to the cells of nerve or muscles fibers. Muscles and organs with sensors e.g. eyes or ears send information to the other types of neurons which are named receiver neurons [54]. When a person starts learning actually in his brain, changing operation of synaptic connections are take place. As a result, electrical activities take place in internal of a neuron and chemical processing takes place only in synapses [55].

The first model of the simulated biological neurons which was called threshold logic algorithm was developed by McCulloch and Pitts in 1943. Their model presented a various hypothesis of neurons' estimations [56]. Afterward, Hebbien unsupervised learning method was suggested by Hebb and his proposed assumption model was inspired by neural plasticity. A simulator of Hebbian learning network is developed by Farley and Clark and Rochester et al. in 1950 [57]. Rosenblatt developed a pattern recognition model which was called perceptron model. His created model of the network has two layers and its computation system contained addition and subtraction operations [58].

Minsky and Papert created single layer neural network in 1960. Their created model was not able to simulate the XOR operation. XOR function is shown by figure XOR. Lack of powerful computers to overcome the problems of high time consumption lead to suspend the neural networks studies.

After a while, a biological principle learning algorithm of neural networks was proposed by Klopf in 1972 [59]. Backpropagation learning algorithm which used multiple layers and various threshold functions was created by Werbos in 1975 and it could overcome XOR problem by using only one hidden layer. Since the progress of SVM and linear classifiers were considerably high, ANN has become a less interested algorithm. In late 2000, interests of deep learning methods captured attentions to ANN again [60].

2.1.1. Perceptron concept

Perceptron learning algorithm of the neural network was created by Rosenblatt in 1958. The perceptron algorithm was the first algorithm for simulation of human learning system [58].

A perceptron is a single neuron learning algorithm and creates an output of a single neuron by calculating the weights of inputs and applying threshold activation function by

considering the threshold of the activation function as bias (b). A single layer perceptron is shown by figure Figure 2.2.

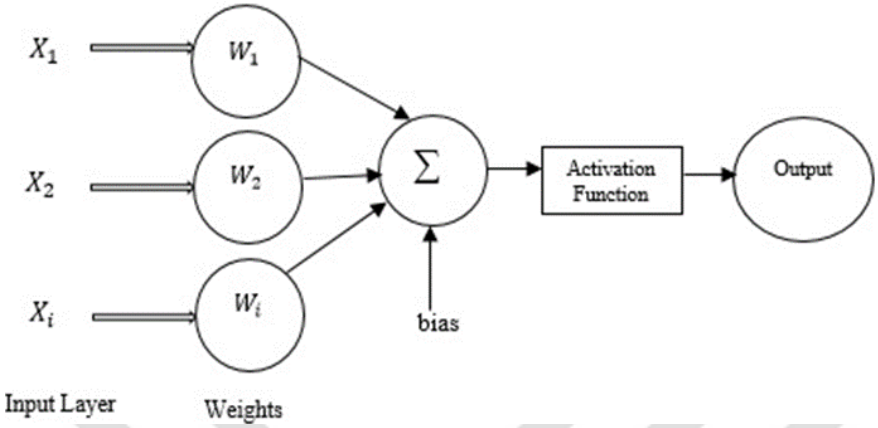

Figure 2.2. Single layer perceptron

By comparison of the bias value and sum of weights, the output of the perceptron will be 1 if sum of weights is larger than bias otherwise output is 0. (Eq. 2.1)

$$Output = \{\ 1 \quad if \quad (\sum_i W_i X_i) > b$$

$$0 \quad \text{otherwise} \qquad \} \tag{2.1}$$

Since a single layer perceptron is a linear learning model and makes decision among two classes it is not able to solve nonlinear problems, e.g., XOR function. A linear and nonlinear separating models are shown by Figure 2.3. A linear separable can separate objects in two sides whereas nonlinear separable is not able to separate objects in two sides by a one-dimension hyperplane. As mentioned before XOR function is a simple nonlinear separable function [61, 53].

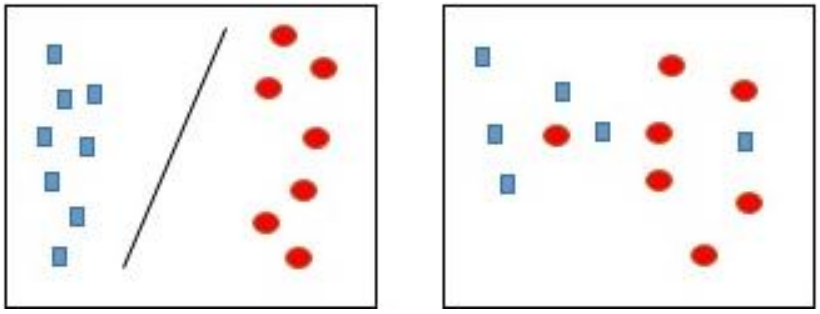

Figure 2.3. Linear separable (left), Nonlinear separable (right)

2.1.2. Activation functions

To make decision about domain of the output of the ANNs, activation function is used. To achieve nonlinearity of output, activation function applies mathematical operation on the real values of inputs [62]. Some of the most used activation functions are described below.

Logistic sigmoid function

One of the most used activation function is logistic sigmoid function. Logistic sigmoid function could reduce the computational of training and this advantage of the function leads to be more acceptable than the other functions [63, 64]. Function of logistic sigmoid and its derivative are given by Eq. 2.2 and Eq. 2.3 respectively. Diagram of logistic sigmoid function is shown by Figure 2.4.

$$f(z) = \frac{1}{1+\exp(-z)} \tag{2.2}$$

$$f'(z) = f(z)(1 - f(z)) \tag{2.3}$$

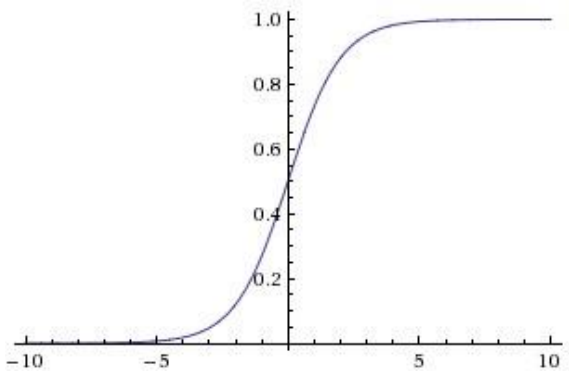

Figure 2.4. Logistic sigmoid function diagram

Hyperbolic tangent function

Hyperbolic tangent function is the superior function. Formula of the Hyperbolic tangent and its derivative are given by Eq. 2.4 and Eq. 2.5 equations. By considering Eq. 2.4, weighted inputs are determined as z in the range of 0 and1 and calculated outputs are in the range of -1,1 [65]. Diagram of Hyperbolic tangent is shown by Figure 2.5.

$$f(z) = \tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^{x}-e^{-z}}{e^{z}+e^{-z}} \quad (2.4)$$

$$f'(z) = 1 - (f(z))^{2} \quad (2.5)$$

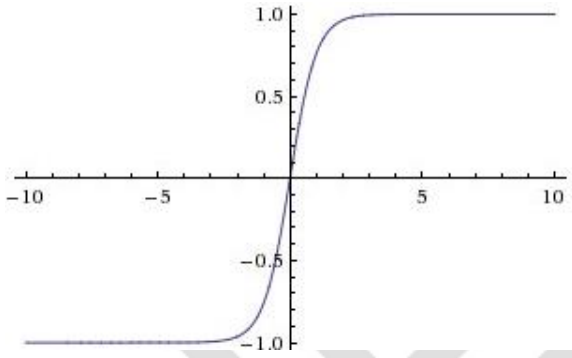

Figure 2.5. Hyperbolic tangent diagram

Gaussian function

Gaussian function is a continuous function and its output value is in the range of 0 and 1. Formula of the Gaussian function is given by Eq. 2.6 equation. In this formula σ represented standard deviation [66]. Diagram of the bell shaped of the Gaussian function is shown by Figure 2.6.

$$f(z) = e^{-\frac{z^{2}}{2\sigma^{2}}} \quad (2.6)$$

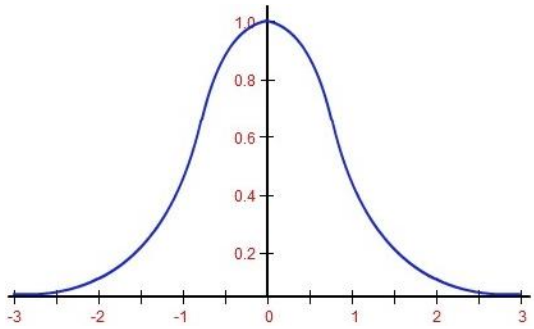

Figure 2.6. Gaussian function

Relu activation function

Generally, in CNNs for increasing nonlinearity Rectified linear unit activation function (Relu) is used. Using Relu activation function takes advantages of high performance, fast learning and simple structure therefore Relu activation function is more preferred than Logistic sigmoid and Hyperbolic tangent functions. Formula of Relu function and its derivative are shown by Eq. 2.7 and Eq. 2.8 equations. For $z \leq 0$ the gradient of Relu function is 0 in other respects the gradient of Relu is 1. Although gradient for $z = 0$ is not defined, calculating average of gradient through training could achieve the result [67]. Diagram of the Relu activation function is shown by Figure 2.7.

$$f(z) = \max(0, x) \tag{2.7}$$

$$f'(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases} \tag{2.8}$$

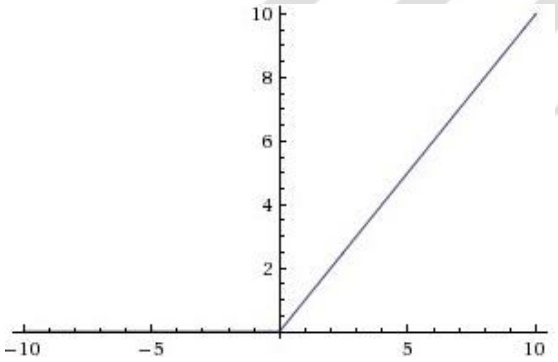

Figure 2.7. Rectified linear unit's function diagram

2.1.3. Multilayer Perceptron

Since single layer perceptron could not solve the nonlinear problems, MLP architecture was proposed to overcome the nonlinear problem. In MLP architecture, hidden layers are proposed between the input layer and the output layer. Information is sent to hidden layers from input layer and after applying operations on information in the hidden layer, they were sent to the output layer. MLP leads the perceptron to solve the nonlinear problems [53].

In ANN, all layers are connected to each other and the output unit of one layer could be the input unit of the next layer. MLP with one hidden layer and one output unit is shown by Figure 2.8. In this architecture, the nodes without any connections by +1 values are called bias and the other parts of the architecture are input layer on the left side of the network and the output layer with one node on the right side of the network. One hidden layer is located between input and output layers and through training process the values of hidden nodes are not observable. The number of layers is denoted by n_1 and in this architecture number of layers are one. Input and output layers are denoted by L_1 and L_{n_1} respectively. The parameres of this network are formulated as Eq. 2.9 equation. By extending equation Eq. 2.9 it can be interpreted that $W_{ij}^{(l)}$ is the parameters (weights) among unit *j* which is located in layer l and unit i which is located in layer $l+1$. Moreover bias of layer $i+1$ and related to unit i is $b_i^{(l)}$. In figure 2.8, biases are denoted as $W^{(1)} \in R^{3\times3}$ and $W^{(2)} \in R^{1\times3}$. By considering layer as , number of nodes are denoted s_l.

$$(\mathrm{W},\mathrm{b}) = (\mathrm{W}^1,\mathrm{b}^1,\mathrm{W}^2,\mathrm{b}^2), \tag{2.9}$$

For unit i of layer , $a_i^{(l)}$ is the activation of layer l , therefore $a_i^{(l)} = x_i$ is the *i*-th input in layer $l=1$. For W, b parameters $h_{W,b}(x)$ is defined as hypothesis of Figure 2.8 of neural network and it is given by Eq. 2.10, Eq. 2.11, Eq. 2.12 and Eq. 2.13 equations.

$$\mathrm{a}_1^2 = \mathrm{f}(\mathrm{W}_{11}^{(1)}\mathrm{x}_1 + \mathrm{W}_{12}^{(1)}\mathrm{x}_2 + \mathrm{W}_{13}^{(1)}\mathrm{x}_3 + \mathrm{b}_1^{(1)}) \tag{2.10}$$

$$\mathrm{a}_2^2 = \mathrm{f}(\mathrm{W}_{21}^{(1)}\mathrm{x}_1 + \mathrm{W}_{22}^{(1)}\mathrm{x}_2 + \mathrm{W}_{23}^{(1)}\mathrm{x}_3 + \mathrm{b}_2^{(1)}) \tag{2.11}$$

$$\mathrm{a}_3^2 = \mathrm{f}(\mathrm{W}_{31}^{(1)}\mathrm{x}_1 + \mathrm{W}_{32}^{(1)}\mathrm{x}_2 + \mathrm{W}_{33}^{(1)}\mathrm{x}_3 + \mathrm{b}_3^{(1)}) \tag{2.12}$$

$$\mathrm{h}_{\mathrm{W}},\mathrm{b}^{(\mathrm{x})} = \mathrm{a}_1^{(3)} = \mathrm{f}\left(\mathrm{W}_{11}^{(2)}\mathrm{a}_1^{(2)} + \mathrm{W}_{12}^{(2)}\mathrm{a}_2^{(2)} + \mathrm{W}_{13}^{(2)}\mathrm{a}_3^{(2)} + \mathrm{b}_1^{(2)}\right) \tag{2.13}$$

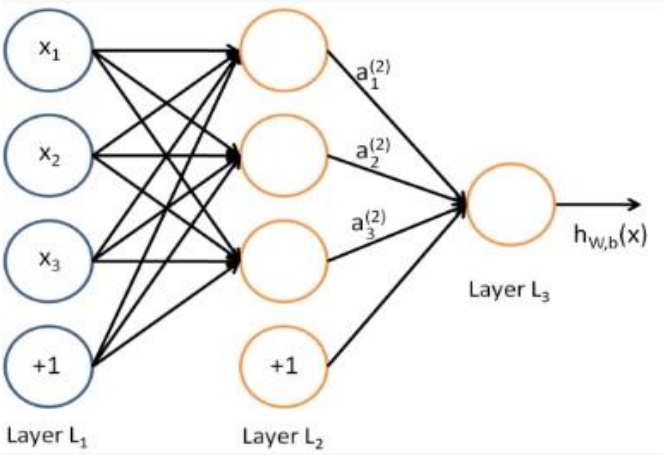

Figure 2.8. Multilayer Perceptron ANN with one hidden layer and one output

Eventually for unit i of layer l, z_i^l is sum of all weighted inputs by bias and it can be formulated as Eq. 2.14. Therefore $a_i^{(l)}$ is formulated as a function of $z_i^{(l)}$ by Eq. 2.15. By extending the activation function of $f(.)$ as Eq. 2.16, z^2, $a^{(2)}$, z^3, $h_{w,b}(x)$ functions are given as Eq. 2.17, Eq. 2.18, Eq. 2.19 and Eq. 2.20 respectively.

$$z_i^{(2)} = \sum_{j=1}^{n} W_{ij}^{(1)} x_{j+b_i}) \tag{2.14}$$

$$a_i^{(l)} = f\left(z_i^{(l)}\right) \tag{2.15}$$

$$f([z_1, z_2, z_3) = [f(z_1), f(z_2), f(z_3)] \tag{2.16}$$

$$z^2 = W^{(1)}x + b^{(1)} \tag{2.17}$$

$$a^{(2)} = f\left(z^{(2)}\right) \tag{2.18}$$

$$z^3 = W^{(2)}a^{(2)} + b^{(2)} \tag{2.19}$$

$$h_{w,b}(x) = a^3 = f\left(z^{(3)}\right) \tag{2.20}$$

This stage is called forward propagation. The total weighted sum of inputs in layer $l + 1$ and activation layer of $l + 1$ are formulated by Eq. 2.21 and Eq. 2.22.

$$z^{(l+1)} = W^{(1)}a^{(1)} + b^{(1)} \tag{2.21}$$

$$a^{(l+1)} = f\left(z^{(l+1)}\right) \tag{2.22}$$

By applying linear algebra on data the structure of matrix and matrix-vector network computations will be fast and efficient. For network with more hidden layers' activations are calculated as the last mentioned equations by forward propagation step.

ANNs not only have one output unit but also have more than one output unit. MLP neural network by two hidden layers and two units in the output layer are shown by Figure 2.9. For training of multiple output units examples of $(x^{(i)}, y^{(i)})$ are required. In some fields such as solving medical problems for diagnosis of a disease two, vectors are required. x for inputs and y for classes of outputs. (healthy or not healthy) [68].

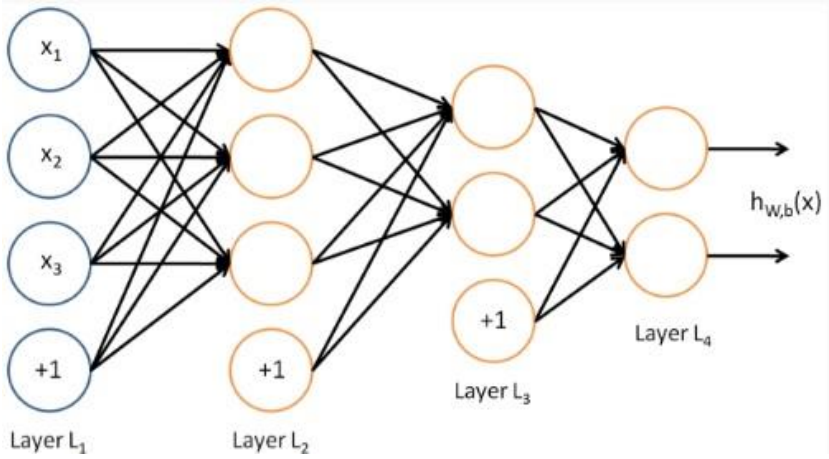

Figure 2.9. Multilayer Perceptron ANN with 2 hidden layers and 2 output unit

Loss function (cost function)

Loss function (cost function) is determined for performance evaluation of neural networks. Loss (cost) function calculates differentiate between the prediction of labels which are achieved by the algorithm and ground truth labels. There are several loss (cost) functions for measuring how ANN did well [69]. Mean Squared Error (MSE) and Cross entropy error are two common loss functions which are described below.

Mean squared error

The most popular and the simplest loss (cost) function is MSE and its equation is given by Eq. 2.23. In Eq. 2.23 equation number of training samples are denoted by m and the i^{th}

example of training is denoted by x^i as well. Furthermore for class labeling of i^{th} example of training y^i is defind in Eq. 2.23 equation and eventually $h(x^i)$ is denoted to predict the i^{th} training example of algorithm [70].

$$\mathrm{L(W,b)} = \frac{1}{m(\sum_{i=1}^{m} ||h(x^i) - y^i||^2)} \tag{2.23}$$

Cross entropy error

In classification and probabilistic problems, the most popular loss (cost) function is cross entropy error function. The formula of cross entropy is given by Eq. 2.24. In this equation, x values are denoted as inputs and a_1^L , a_2^L ,… and a_j^L are denoted as real values of all output neurons (j) in output layer and y indicates desired output values and n is number of all training samples [71].

$$\mathrm{C} = -\frac{1}{\mathrm{n}}\left(\sum_{\mathrm{x}} \sum_{\mathrm{j}} [\mathrm{y_j Ln\ a_j^L} + \left(1 + \mathrm{y_j}\right)\mathrm{Ln}(1 - \mathrm{a_j^L}\right) \tag{2.24}$$

2.1.4. Gradient descent algorithm

Gradient descent optimization algorithm tries to find local minimum to minimize loss (cost) function $(L(\theta))$ by weights (parameters) which are denoted as θ. In Gradient descent algorithm weights are upadated contrary in direction of loss function's slope and it denoted as $\nabla_\theta\, L(\theta)$.The step size to reach the bottom of the slope direction in order to minimize loss function is denoted by η which is called learning rate. Depending on the amount of data three types of Gradient descent algorithms are used which are described and formulated below. It is considerable that amount of data affects performance and time of updating [72, 73].

Batch gradient descent

To minimize loss function, weights (parameters) (θ) are updated by gradient algorithm which it is applied on the whole training set in each iteration and it reaches to the one set of updated weights.

Because of updating all training set in each iteration in the large dataset the computation of batch gradient descent takes too much time. More computation in large dataset leads to

having a redundant operation to update all samples of training set in order to reach one updated set of parameters. To achieve the convergence batch gradient descent computes the global minimum and the local minimum in convex and non-convex areas respectively. This algorithm updates weights with low speed. Batch gradient descent formula is shown by Eq. 2.25 equation. In this equation, η and θ are learning rate and weights (parameters) respectively [73].

$$\theta = \theta - \eta . \nabla_{\theta} L(\theta) \tag{2.25}$$

Stochastic gradient descent

Stochastic Gradient Descent (SGD) algorithm apply updating operation on each sample of training which is denoted by x^i and its label denoted by y^i in each iteration. Each training sample is updating without depending on the other samples therefore, redundant computation does not take place. Applying SGD without any recalculation of training samples leads to be faster and more popular than batch gradient descent algorithm. SGD formula is given by Eq. 2.26 equation [73].

$$\theta = \theta - \eta . \nabla_{\theta} L\left(\theta; x^i; y^i\right) \tag{2.26}$$

Mini batch gradient descent

In mini batch gradient descent algorithm for each iteration n samples of training set are updated rather than one sample in each iteration. Updating the subset of parameters in each iteration to minimize loss function by mini batch gradient descent algorithm leads to be fast and more convergence. Mini batch gradient descent formula is given by Eq. 2.27 equation where subset of parameters are started from x^i and its label by y^i to x^{i+n} by their lables of y^{i+n} [73].

$$\theta = \theta - \eta . \nabla_{\theta} L\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right) \tag{2.27}$$

2.1.5. Backpropagation algorithm

Backpropagation algorithm is one of the best learning algorithms of ANNs, due to its ease of use in computation, conception, and function [74]. Bryson and Bo in1969 have presented Backpropagation algorithm for the first time [75]. In 1974, Werbos and Rumelhart have attempted to rediscover the backpropagation algorithm. In 1986, PDP group (David Rumelhart and McClelland) has utilized and developed the backpropagation algorithm to calculate the gradients [76, 77]. Backpropagation algorithm formulated as below:

By considering batch gradient descent to train the neural network, and MSE equation which is given by Eq. 2.23 equation, computation of the loss (cost) function for a single training set of (x, y), is given by Eq. 2.28.

$$\mathrm{J(W,b;x,y)} = \frac{1}{2}\left|\left|\mathrm{h_{W,b}(x)} - \mathrm{y}\right|\right|^2 \tag{2.28}$$

For m numbers of training sets which are shown in Eq. 2.29, the loss function is formulated by Eq. 2.30. An average sum-of-squares error is the first part of the loss function. To recline value of weights and avoid of overfitting, weight decay was considered in the second term of the cost function.

$$\left\{\left(\mathrm{x}^{(1)},\mathrm{y}^{(1)}\right),\ldots,\left(\mathrm{x}^{(\mathrm{m})},\mathrm{y}^{(\mathrm{m})}\right)\right\} \tag{2.29}$$

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m} J\left(W,b;x^{(i)},y^{(i)}\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_1-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{l}\right)^2 =$$
$$\left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left|\left|h_{W,b}\left(x^{(i)}\right) - y^{(i)}\right|\right|^2\right) + \frac{\lambda}{2}\sum_{l=1}^{n_1-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{l}\right)^2 \right. \tag{2.30}$$

In classification problems, by using sigmoid activation function the y term of cost function would be 0 or 1, and by using tanh activation function it would be -1 or +1 to define the labels of classification. By initializing W_{ij}^{l} and $b_i^{(l)}$ to a value near zero and using the batch gradient descent algorithm neural network would be trained. In this regard to obtain minimum value of the cost function ($J(W,b)$) is the main purpose of backpropagation algorithm. Gradient descent is sensitive to local optima because cost function type is a non-convex. In contrast, practically gradient descent works great enough. It is essential to symmetry breaking and prevent the same value of outputs all parameters would be initialized

by random value. In Eq. 2.31 and Eq. 2.32 bias (b) and weigh (w) values were updated by gradient descent. By considering α as the learning rate.

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) \tag{2.31}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b) \tag{2.32}$$

Partial derivatives would be calculated by the method which backpropagation algorithm proposed. A derivative of the cost function ($J(W, b)$) would be calculated by Eq. 2.33 and Eq. 2.34.

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) = \left[\frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial W_{ij}^{(l)}} J\left(W, b; x^{(i)}, y^{(i)}\right)\right] + \lambda W_{ij}^{(l)} \tag{2.33}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial b_i^{(l)}} J\left(W, b; x^{(i)}, y^{(i)}\right) \tag{2.34}$$

In this regard firstly in backpropagation algorithm for a (x, y), as an example of training set (forward pass) would be applied. Activations and the output value of $h_{W,b}(x)$ would be calculated in forward pass step. The error measure ($\delta_i^{(l)}$) of each node (i) in each layer (l) for determine errors occurs in output would be computed. By considering n_l as an output layer of the network, the output node ($\delta_i^{(n_l)}$) would be the result of the difference among activation of the network and the real value of the network's target. The steps of backpropagation algorithm are given below:

1) To calculate the activations of layers ($L_2, L_3, \dots, L_{n_l}$), feedforward pass were applied.
2) The output in layer n_l formulated by Eq. 2.35 where i is the unit number of each output.

$$\delta_i^{(n1)} = \frac{\partial}{\partial z_i^{(n1)}} \frac{1}{2} ||y - h_{W,b}(x)||^2 = -\left(y_i - a_i^{n_l}\right).f'\left(z_i^{n_l}\right) \tag{2.35}$$

3) $\delta_i^{(l)}$ was computed by Eq. 2.36 for every node (i) in layers $l = n_l - 1, n_l - 2, n_l - 1 n_l - 3, \dots, 2$. By considering $f(z)$ as a sigmoid activation function $f'(z)$ computed as Eq. 2.37.

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_l+1} W_{ij}^{(l)}\ \delta_j^{(l+1)}\right) f'(z_i^l) \tag{2.36}$$

$$f'(z_i^l) = a_i^l(1 - a_i^l) \tag{2.37}$$

4) The partial derivatives were computed by Eq. 2.38 and Eq. 2.39.

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)} \tag{2.38}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)} \tag{2.39}$$

To decrease the cost function $J(W, b)$ in training of ANN, batch gradient descent would be repeated in several iterations. One iteration of batch gradient descent by considering to $\Delta W^{(l)}$ as a matrix with the same dimension of $W^{(l)}$ and $\Delta b^{(l)}$ as a vector with the same dimention of bias ($b^{(l)}$) was given in the pseudo-code [71].

1) For all layers (l), $\Delta W^{(l)}$ and $\Delta b^{(l)}$ were set by zero as Eq. 2.40 and Eq. 2.41.

$$\Delta W^{(l)} := 0 \tag{2.40}$$

$$\Delta b^{(l)} := 0 \tag{2.41}$$

2) For i=1 to m, $\Delta W^{(l)}$ and $\Delta b^{(l)}$ computed by backpropagation algorithm, given in Eq. 2.42 and Eq. 2.43.

$$\Delta W^{(l)} := \Delta W^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y) \tag{2.42}$$

$$\Delta b^{(l)} := \Delta b^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y) \tag{2.43}$$

3) $W^{(l)}$ and $b^{(l)}$ are updated as Eq. 2.44 and Eq. 2.45.

$$W^{(l)} = W^{(l)} - \alpha\left[\left(\frac{1}{m}\Delta W^{(l)}\right) + \lambda W^{(l)}\right] \quad (2.44)$$

$$b^{(l)} = b^{(l)} - \alpha\left[\frac{1}{m}\Delta b^{(l)}\right] \quad (2.45)$$

2.2. Deep Learning and Convolutional Neural Networks

In shallow machine learning methods to discover features form dataset, manual feature selection is applied on features and selected features feed to a particular machine learning algorithm, Whereas deep learning methods are able to extract features from raw dataset automatically then detect or classify dataset by extracted low, middle and high features [78-80]. The main characteristic of deep learning methods is applying nonlinear functions on raw data as inputs to produce abstracted outputs [78]. Nowadays, it is easy to access to big datasets and computers by powerful processing systems which they are the main requirements of deep learning methods. Therefore, availability of the essential necessities of deep learning methods make them to be appropriate and popular to solve problems [81]. CNN, DBN, Recurrent Neural Networks (RNN), Long Short Term Memory (LSTM) and Deep Stacking Networks (DSN) are deep learning architectures which are used in computer vision, automatic audio classification and natural language processing fields to solve problems of large datasets [82-84].

Traditional ANN which is one of the first projects about visual cortex of cats by Hubel and Wiesel and CNNs are alike. Both algorithms include neurons which contain weights and biases [85]. Although the structure of both CNNs and ANNs include layers, there are main differents between the structure of both networks. Structure of layers in ordinary ANN algorithm is one dimensional and connections of all layers are fully connected. CNNs have three-dimensional neurons in a layer which include width, height, and depth. Furthermore, in CNNs each neuron of one layer connected to the only one region of the previous layer without any fully connected between layers. Inputs of outputs of CNN have a 3D volume of width, height, and depth. Three dimensional of a neuron is shown by Figure 2.10, which is shown in CNN each neuron has 3D volume. From 2000, to solve different problems of nervous systems, biological systems and natural systems, CNNs have achieved acceptable results by using detection, segmentation and recognition methods [78, 86-88].

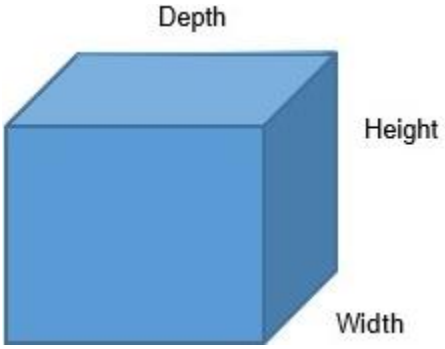

Figure 2.10. 3D volume of neuron in CNN structure

In 1989 the first CNN was developed for classification of handwritten digits by Le Cun et al. They used backpropagation and gradient descent algorithms in their developed CNN. Until 1990, CNNs are utilized in Commerce fields e.g. reading of cheques [89]. Because of the main requirements of CNNs are large datasets and powerful computers, in last two decades using of CNNs were to be stopped and the other machine learning methods became to be popular. In 2006 for the training of a special DNN greedy layer-wise pre-training algorithm is proposed by Hinton et al. Their proposed algorithm caused to drop attention of scientists to deep learning algorithms [90].The first use of GPU for the training of CNNs by Ciresan et al. took place in 2011 and their works were about handwritten digits of MNIST dataset. Nowadays achievements of using CNNs in different fields especially in solving supervised problems make it to be popular. In last decades different work groups achieved successful results by using CNNs in the competition of ImageNet Large Scale Visual Recognition challenge (ILSVRC). Krizhevsky et al achieved 15.3% classification error in ILSVRC-2012 [91] and Clarifia group achieved 11.7% error rate in ILSVRC-2013 [92]. In ILSVRC-2014 GoogleNet group gained 6.66% classification error rate. MSRA group and Trimps- Soushen group achieved 3.57% error rate in ILSVRC-2015 and 2.99% error rate in ILSVRC-2016 as winners of challenge respectively [63, 93]. LeNet, AlexNet, GoogleNet, VGG-Net, Res-Net, ZF-Net are the most used architectures of CNN algorithm in classification and pattern recognition fields [63, 91-94]. Architectures of CNN take the advantage of powerful GPUs to minimize training time and improve the accuracy of classifications [95].

One of the advantages of using deep learning algorithms was its efficiency on a huge amount of datasets. Since deep learning achieved high performance in large datasets of image and speech, the need for powerful hardware and appropriate software become to the most

important challenges. The performance of the powerful GPUs was more than CPUs. The first use of GPU in ImageNet proposed and it was achieved high performance in classification of ImageNet [96]. The results of GPU based method was 10% higher than the CPU based method. In this thesis, the large dataset of lung CT scan images was used and consequently, the CPU based method could not handle all samples. To profit the high performance of GPU for faster classification, Nvidia Titan (12 GB) GPU was used in this work. Nvidia with the support of Computer unified device architecture (Cuda) and CuDNN library was considered. In this thesis deep learning GPU Training System (Digits) as a framework for training the different architecture of CNN by using Caffe framework and CuDNN library is used. Nvidia Caffe Digits is our used platform for CNN modeling of lung CT scan dataset. Intel Corei5 is used as the cpu of the PC and Ubuntu version 16.4 is used as operation system.The description of requirements of this work is given below.

Cuda for GPU computing: Nvidia created Cuda as a parallel computer platform on GPU. In order to use the Cuda by developers, Nvidia created the toolkit which contains a compiler, libraries, debugger and etc.

CuDNN: CuDNN is a library of Nvidia's GPU for using DNNs and it uses deep learning frameworks such as Caffe, TensorFlow and etc [97].

Caffe Framework: The Caffe open source framework is developed by Berkeley Vision and Learning Center (BVLC) to implement the deep learning networks. Different architectures of deep learning are supported by Caffe. Moreover, it is a C++ library and designed to has bindings to Phyton and Matlab as well. Because of computational complexity, Caffe uses CPU and GPU in parallel in order to accelerate the processing. It leads to decrease the time of training model from days to hours.

Digits: Nvidia provides Digits as a framework that supports Caffe and Torch to train and design deep train networks. Furthermore, it accelerates classification and segmentation tasks [97].

2.2.1. Architecture of convolutional neural networks

In fully connected ANNs architecture input layer is a one dimensional which are transferred to the hidden layers and then they are sent to the output layer. Although neurons in hidden layers are not connected to each other they are connected to all neurons in the previous layer. In classification problems output layer determines the score of each class of dataset. In contrast to fully connected ANNs, in CNNs there is not any fully connected in middle layers and each neuron connected to a local region which it contains a part of neurons in the previous layer. In CNNs, filter banks are used as a unit to connect to the part of the previous layer and it is called weight connection. In CNN layers by using local connections local features are detected and by pooling operations identical features are merged to be one feature. The architecture of CNN contains three main layers as a convolutional layer, pooling layer and a fully connected output layer. Moreover, some other layers e.g. normalization layer, are used beside main layers [98]. The main three layers of CNN are described below, afterward normalization layer is described as well.

Convolutional layer

The convolutional layer which is the main part of the architecture of CNN includes feature maps (depth slices) and each feature map includes sets of neurons. Similar to ANN neurons in CNN imitate biological neurons. The main difference of neurons in feature maps of convolutional layers and neurons in ordinary neural network layers is their connection types. The connections of neurons in ANNs are fully connections while the connections between neurons of the convolutional layer are local connections. In local connection each neuron in feature map connected to the part of neurons of the previous layer. Although the connection between neurons in CNN are local connection type, similar to ANN the output of a neuron in CNN is calculated by a nonlinear activation function. Relu is one of the most common activation functions which is used in training of data in CNN.

In each convolutional layer filters which are the connections between neurons of current layer and neurons of the previous layer, are used. Local connections and parameters sharing are three characteristics and advantages of the convolutional layer which are described below [99, 100]. Before describing advantages of convolutional layer some terms of the convolutional layer are explained afterward implementation of CNN is described below.

Spatial arrangement

Hyperparameters are concepts for output size management and it contains filter, stride and zero padding parameters. Hyperparameters of the convolutional layer are explained below.

Filter (Kernel): In CNN to train the network, features of input are detected by arrays of parameters (weights) which are called filters (Kernels). The region of input which a three-dimensional filter is applied on is called receptive field and its size is equal to a filter size. The output of convolving filters over input is called feature map (activation map) and the number of filters and feature maps of a convolutional layer are the same. Actually, the number of feature maps are the depth of output of a convolutional layer [101].

Stride: Stride size is the step of shifting by filters (kernels) on the input image. In convolutional layer by stride size 2, the filter is shifted by 2 pixels on the input [99].

Zero padding: In convolutional layer to provide the output volume in size of input volume, zero padding is used thus input size in width and height is controlled by the convolutional operation. Using of zero padding with stride size is given by Figure 2.11. In this figure input size is 5x5. Filter by 3x3 size and 1 border for zero padding with 1 for stride size are applied on the input. After convolutional operation output size is same as the input size (5x5), it means input size is retrained after using zero padding.

Formulation of hyperparameters of convolutional layer: By considering parameters of convolutional layer where,

N: Number of neurons in output

K: Numbers of filters (kernels)

F: Numbers of the receptive field size (filter size)

S: Stride size

P: Number of zero padding

W_1: Width of input

H_1: Height of inputs

D_1: Depth of inputs

W_2: Width of output

H_2: Height of inputs

D_2: Depth of inputs

W_k: Number of parameters for each filter

T_p: Sum of parameters

The width, height and depth size of output are formulated by Eq. 2.46, Eq. 2.47 and Eq. 2.48 equations respectively. The acceptable value of output volume is to be an integer, otherwise, stride size must to be changed. By considering formula of zero padding as Eq. 2.49 equation, the secure stride size is one and it guarantees the input and output are in the same size. For each filter in convolutional layer number of parameters (weights) and the sum of all parameters (weights) are formulated by Eq. 2.50 and Eq. 2.51 equations.

Furthermore, numbers of biases and numbers of filters in the convolutional layer are the same [72].

$$W_2 = \left(\frac{(W_1 - F + 2P)}{S}\right) + 1 \tag{2.46}$$

$$H_2 = \left(\frac{(H_1 - F + 2P)}{S}\right) + 1 \tag{2.47}$$

$$D_2 = K \tag{2.48}$$

$$P = \frac{(F-1)}{2} \tag{2.49}$$

$$W_k = F \cdot F \cdot D_1 \tag{2.50}$$

$$T_p = (F \cdot F \cdot D_1) \times K \tag{2.51}$$

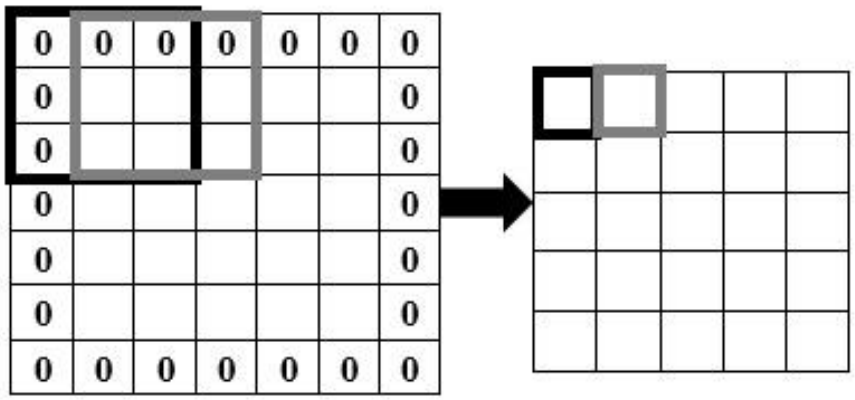

Figure 2.11. Input size of 5×5 with filter by 3×3 size and 1 for zero padding (left), output by 5×5 size (right)

Local connection

Filters in the convolutional layer are applied on width and height of input and over, all depth of input as well, therefore the connections between feature maps and the previous layer are local connections. In local connection regions of the previous layer are mapped to the feature map locally. Decreasing amount of parameters is the most significant advantage of local connections. The first convolutional layer is shown by Figure 2.12 and it shows that each neuron in the convolutional layer connected to the region of input in width and height.

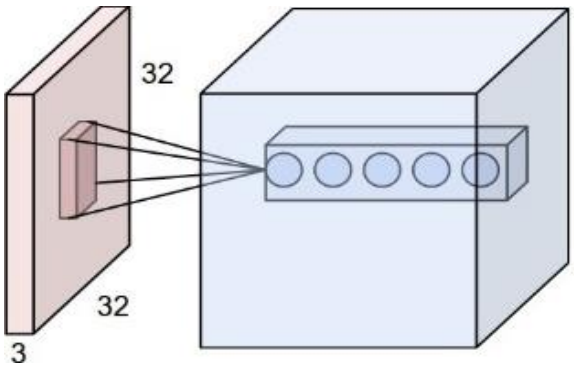

Figure 2.12. Example of local connection in the first convolutional layer

For more explanation of local connection and reducing amount of parameters, in Figure 2.12, input size is $32 \times 32 \times 3$ and filter by 5×5 size are convolved. After the convolutional

operation by considering 1 for bias, 76 is all parameters (weights) of each neuron ((5×5×3) +1=76). In contrast to the local connection if connections were fully connection, each neuron had more parameter than local connection. ((32×32×3) +1=3073 for each neuron. Fully connected of ordinary ANN and local connections of CNN are shown by Figure 2.13 [71, 101].

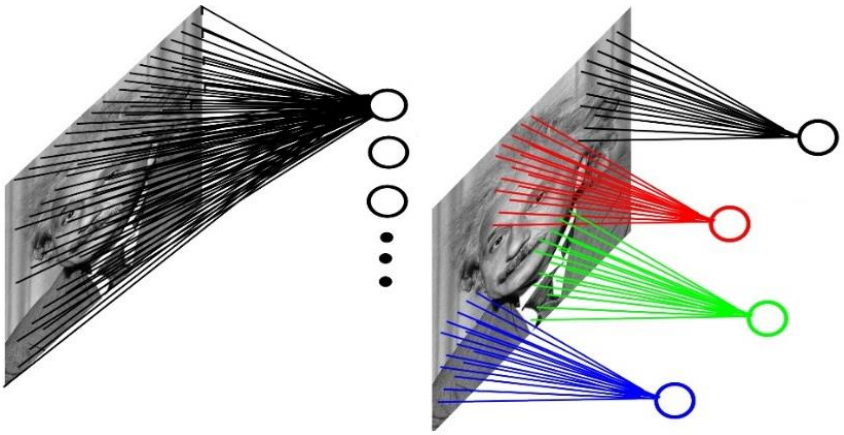

Figure 2.13. The fully connected architecture (left), local connections (right)

Parameters (weight) sharing

In ordinary ANN one weight matrix is applied on input for once and then the output is calculated, however, in convolutional layer by parameter (weight) sharing, one weight matrix is used in all over the input frequently to produce the output. In other words, an acceptable feature in a spatial location can be used in the other spatial locations as well. Count of parameters is controlled by using the same neuron with its parameters (weights) in a depth slice of each convolutional layer. The result of parameter sharing in each depth slice is feature map. The output of each convolutional layer is a collection of feature maps. The advantage of weigh sharing is reducing the number of parameters in training and complicated calculation of a network. Moreover, because of using weigh sharing, translation invariance of input cannot change the outputs of convolutional layer [101, 102].

Implementation of convolutional operation

The output of convolutional operation in a convolutional layer is given by Eq. 2.52. In this equation F is denoted as filter size, m denoted as feature maps, B is denoted as bias and

weight of a filter is denoted as W_j. The output of convolutional layer is denoted as Y_i^l where i indicates i^{th} feature map in a layer that it denoted by l.

$$Y_i^l = B_i^l + \sum_{j=1}^{m_l^{(l-1)}} {F_{i,j}}^l * W_j^{(l-1)} \tag{2.52}$$

By considering that the convolutional operations take place in two dimensions of layer l by ${m_2}^l . {m_3}^l$ units in (r, s) location the output of convolutional layer for MLP is given by Eq. 2.53 equation [103].

$$(Y_i^{(l)})_{r,s} = (B_i^{(l)})_{r,s} + \sum_{j=1}^{m_l^{(l-1)}} (F_{i,j}^{(l)} \times W_j^{(l-1)})_{r,s} = (B_i^{(l)})_{r,s} + \sum_{j=1}^{m_l^{(l-1)}} \sum_{u=-h_1^{(l)}}^{h_1^{(l)}} \sum_{v=-h_2^{(l)}}^{h_2^{(l)}} (F_{i,j}^{(l)})_{u,v} (W_j^{(l-1)})_{r+u,s+v} \tag{2.53}$$

Pooling layer (Subsampling)

To decrease the amount of parameters and network calculation, generally pooling (subsampling) layer is used among convolutional layers; Consequently, by subsampling operation, input size is decreased in all depth parts and it prevents overfitting through network training. Pooling operation is called down sampling as well. Since the spatial size of input is decreased by pooling operation the depth dimension is not changed. According to the example of CNN architecture in all pooling layer spatial dimensions are decreased and depth are the same as the depth of the previous layer (Figure 2.14).

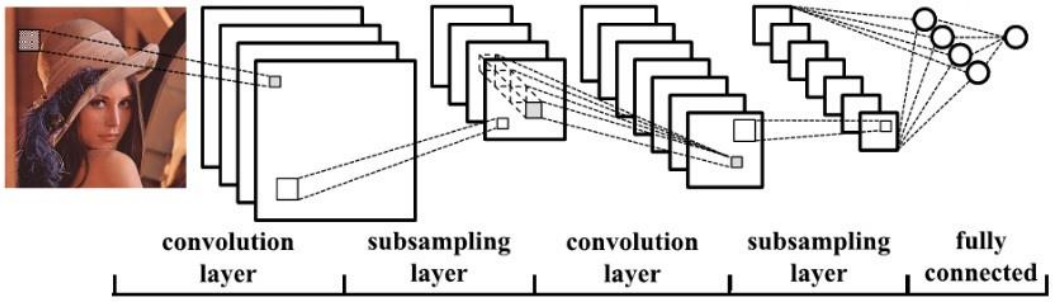

Figure 2.14. Example of CNN architecture

Max pooling and Average pooling are two most commonly used types of pooling operations.

Example of both pooling operations is shown by Figure 2.15.

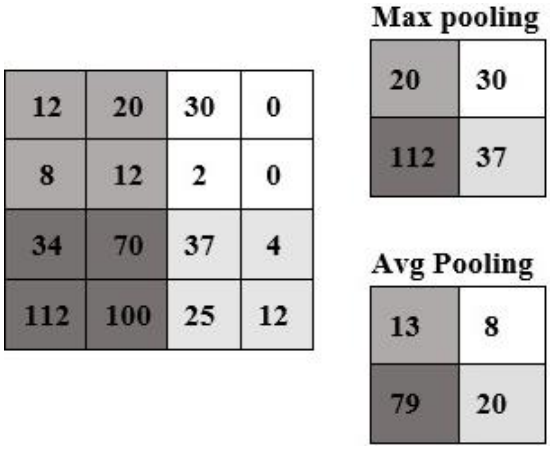

Figure 2.15. Examples of max and average pooling

By considering Figure 2.15, 4×4 is the input size and down sampling operation is applied by 2×2 filter size. The outputs of subsampling operations are the average and maximum values of input values. The width and height of output in pooling layer are achieved by Eq. 2.54 and Eq. 2.55 equations. In formula of volume size of width and height W_1, H_1, D_1 are the width, height and the depth size of input respectively. S is denoted as stride size and F is denoted as filter size. It is considerable that in pooling layer the depth of input after pooling operations is not changed.

$$W_2 = \left(\frac{W_1 - F}{S}\right) + 1 \tag{2.54}$$

$$H_2 = \left(\frac{H_1 - F}{S}\right) + 1 \tag{2.55}$$

Overlapping pooling appears when stride size is smaller than filter size ($S < F$) and pooling by the same value of stride size and filter size is called non-overlapping pooling. Because of eliminating more features choosing large receptive fields in pooling layers is not acceptable for down sampling of input [104].

Fully connected layer

The last layer of CNN architecture is fully connected layer. Similar to ordinary ANN, in the fully connected layer all neurons of a layer are connected to all neurons of the previous layer.

Through training, the score of a class between all classes of the dataset is presented in the fully connected layer. Fully connected operations are given by Eq. 2.56 equation which l and $(l-1)$ are denoted as fully connected layers. Output of the last fully connected is y_i^l which is indicated by i^{th} unit in layer l. In layer l, feature maps of $m_1^{(l-1)}$ by $m_2^{(l-1)} \times m_3^{(l-1)}$ size are denoted as inputs. $W_{i,j,r,s}^{(l)}$ is the weights connections of i^{th} unit in layer l and Y_j which is denoted as j^{th} unit of layer $l-1$ in (r,s) location [103].

$$y_i^{(l)} = f\left(z_i^{(l)}\right) \quad \text{with } z_i^{(l)} = \sum_{j=1}^{m_1^{(l-1)}} \sum_{r=1}^{m_2^{(l-1)}} \sum_{s=1}^{m_3^{(l-1)}} W_{i,j,r,s}^{(l)} \left(Y_j^{(l-1)}\right)_{r,s} \tag{2.56}$$

Normalization layer

In case of demand, besides the main three layers of CNN architecture normalization layers are used after the other layers except fully connected layer. Because of the low effect of normalization layers in CNN architecture, they are used when it is needed. Local response normalization and batch normalization are two most popular normalization types which are used in different CNN architectures [91, 94].

Local response normalization

One of the advantages of using Relu activation function in CNN architectures is to make CNN architecture independent of the normalization layers. Local response normalization is used when their inputs of Relu activation are positive values. Generally, local response normalization algorithm is used to implement the lateral inhabitation [104] of real neurons to improve the contrast of vision. The formula of local response normalization algorithm is given by Eq. 2.57 equation. In this equation $b_{x,y}^i$ is denoted as the local response normalization of kernel i in $(x;\ y)$ location and $a_{x,y}^i$ is denoted as the output of applied kernel i in $(x;\ y)$ location. N is denoted as sume of leyer's kernels and n is denoted as is the number of adjacent convolutional kernels. Other parameters of local response normalization equation are , β and k which have constant values [91, 106].

$$b_{x,y}^i = a_{x,y}^i \Big/ \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2\right)^{\beta} \tag{2.57}$$

Batch normalization

Ioffe and Szegedy (2015), have proposed batch normalization method to improve the learning rate of deep learning. Internal Covariate shift is decreased by applying batch normalization algorithm and network training is improved greatly as well. Generally, similar to the others normalization algorithms, batch normalization is applied after convolutional layers. Learning of mean and variance parameters of batch normalization take place in back propagation algorithm.

For m numbers of activation value $\mathcal{B} = \{x_{1\ldots m}\}$ batch normalization transform was given by pseudo code below:

Input**:** Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;

γ, β would be learned parameters

Output: $y_i = \{BN_{\gamma,\beta}(x_i)\}$

$\mu_{\mathcal{B}} \longleftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ // mini-batch mean

$\sigma_{\mathcal{B}}^2 \longleftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$ // mini-batch variance

$\hat{x}_i \longleftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$ // normalize

$y_i \longleftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$ // scale and shift

In this pseudo code, $\hat{x}_{1\ldots m}$, $y_{1\ldots m}$ and ϵ are the normalized values, transformations of normalized values and the constant value respectively [107].

2.2.2. Softmax function

In the last layer of CNN architecture, softmax function is used to calculate the probability of each ground truth labels of outputs between 0 and 1 and output values convert to perceptible values. Actually, softmax function is the generalized version of binary logistic regression

and it is used for multiple classes. The formula of softmax function is given by Eq. 2.58 equation. In this equation K is denoted as dimensional of random values (z) which are converted to the meaningful values between 0 and 1 by softmax function $f(z)$ [108].

$$f(z)_i = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad For\ j = 1, \dots, K \tag{2.58}$$

2.2.3. Different architectures of CNN

AlexNet

Krizhevsky et al was the winner of ILSVRC-2012 by proposing the AlexNet architecture for the first time. Their proposed architecture contained 5 convolutional layers, three max pooling layers and three fully connected layers. The training duration of AlexNet architecture for ImageNet dataset was about six days. In AlexNet architecture two, GTX 580 GPUs were used to achieved fast training process. The architecture of AlexNet is given by Figure 2.16. In this structure, first GPU is activated on the top of architecture and the second GPU is activated on the bottom of the architecture. Training model is divided into two part and eventually in the last fully connected they join with each other and applied filters are divided in two part on all depth of samples as well [91].

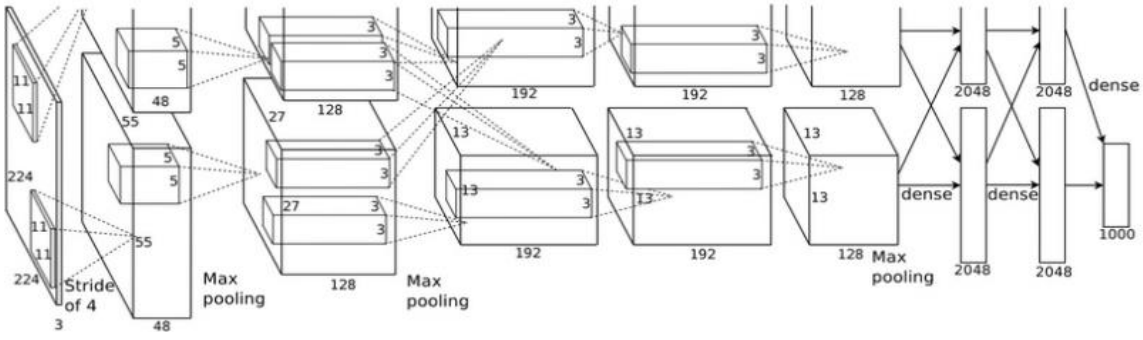

Figure 2.16. AlexNet architecture by using two GPU [91]

GoogleNet

Winners of ILSVRC14 proposed a CNN architecture that was called GoogleNet. They inspired by the inception model of the network in network structure [109] in their own architecture. In network approach in order to use the ability of ANN they used small patches of MLP and shared MLP between all receptive fields of convolutional layers. They increased

the depth of network by adding 1×1 convolutional layers. In general network in network structure stacks various MLP convolutional layers and uses global average pooling instead of fully connected layer at the end of the network. The output of global average pooling is fed to softmax classifier. The figure 2.17 shows the structure of network in network structure. In this structure, three MLP convolutional layers and one global average pooling layer is used.

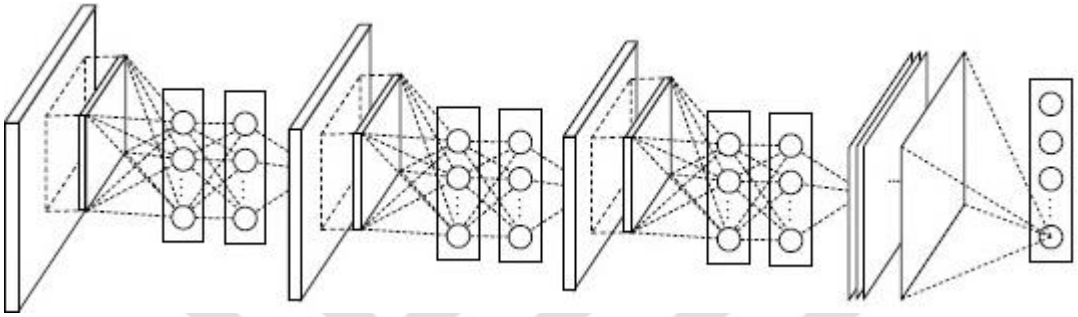

Figure 2.17. The structure of network in network [109]

The most important section of GoogleNet architecture is inception modules. GoogleNet utilizes 9 inception modules which consist of convolutional layers and max pooling layer. In inception model, instead of deciding about the size of kernels (filters) in convolutional layers, the mixture of filters is used. In order to learn more features and having a deeper network, the mixture of filters by 3×3 and 5×5 sizes are applied. In order to decrease the dimension of input in each inception modules, a filter by 1×1 size is used before applying larger filters by 3×3 and 5×5 sizes. Another purpose of using a filter of 1×1 size is the benefit of more linearity by using Relu activation function after each 1×1 filter. Although using the mixture of large filters cause to increasing the convolutional computation, using 1×1 filter reduce the computation before applying larger filters. By using inception structure GoogleNet architecture reduced the number of parameters 12 times less than AlexNet architecture. The inception is shown by Figure 2.18. In this module before applying larger filters, the filter of 1×1 size applied in each convolutional layers. To achieve the perfect results of convolutional layers, beside the convolutional layers max pooling layer is used. After applying inception modules, the concatenation of all used convolutional layers is fed to the next layer.

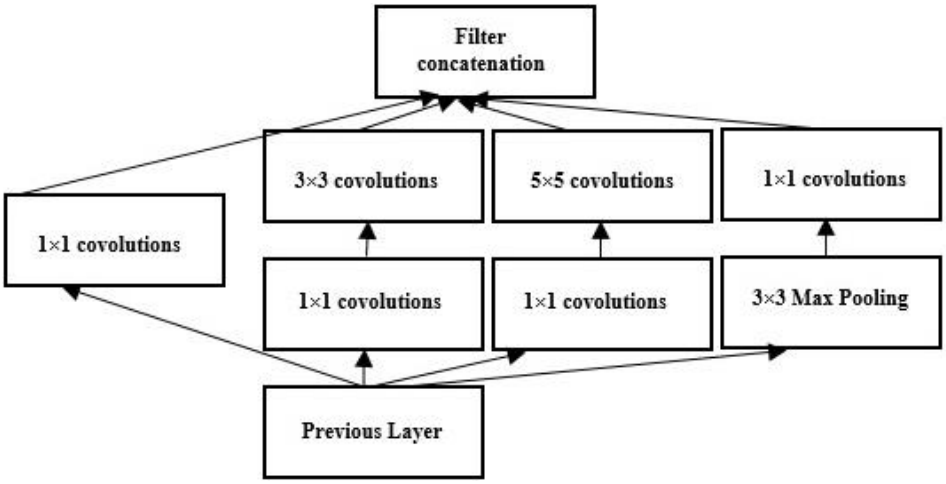

Figure 2.18. Inception module structure [94]

Figure 2.19, shows the GoogleNet architecture. Convolutional layers, pooling layers and softmax were showed by blue, red and yellow rectangles respectively. Green rectangles presented concatenations. The depth of GoogleNet architecture is 22 layers without considering 5 pooling layers. The GoogleNet architecture consists of the beginning section, inception, and the output section. In the beginning section, convolutional operations are applied. 9 inception modules in GoogleNet is designed. In two middle layers that used classification function and the last layer, one average pooling layer is performed. The proposed GoogleNet architecture was called inception-v_1. In this architecture for discrimination of features in training process two auxiliary classifiers were used in the middle inception layers. The main purpose of using two auxiliary classifiers in the middle layers was increase the power of gradient during the propagation. In training process of inception-v_1 the value of loss of auxiliary classifiers and the value of main loss are aggregated and gradient will be propagated [94].

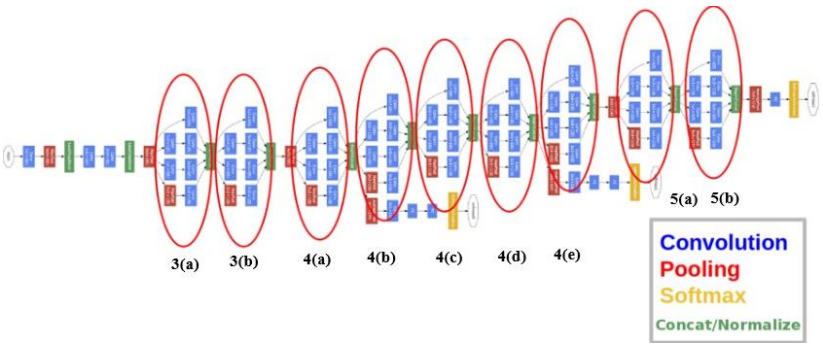

Figure 2.19. GoogleNet architecture [94]

In the other GoogleNet architecture that is called inception-BN (inception-v_2) they proposed other alternatives in the architecture of GoogleNet. Through training the CNN, parameters and the distribution of inputs are changed in all layers and it leads to decrease the training by using lower training rate. They called this problem as internal covariate shift and proposed using the normalization layer. Moreover, to reduce the complicated calculations they proposed using two filters by 3×3 size instead of using filter by 5×5 size in inception modules (Figure 2.20) [107].

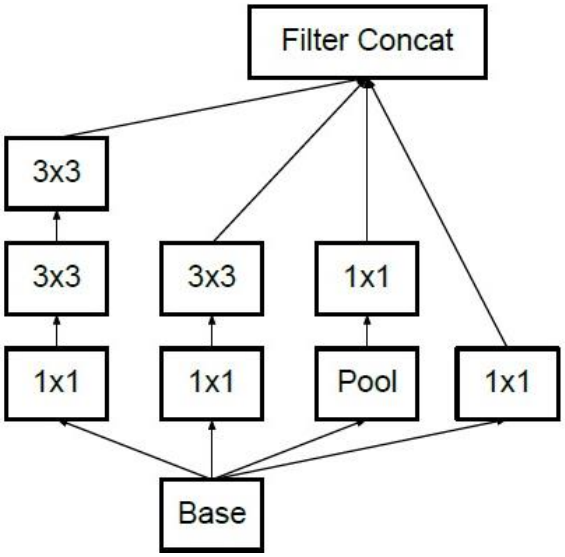

Figure 2.20. Replacement of each 5×5 filter size by two 3×3 filter size in inception module [107]

In inception-v_3 the other version of GoogleNet architecture factorization method was proposed. In factorization method followed by inception-v_2 [107] instead of using $n \times n$ filters of convolutions $n \times 1$ and $1 \times n$ filters could be applied. They have proved using smaller kernals decreased the cost of computional of middle layers. Factorization method of inception module is shown by Figure 2.21. It is obvious that each 3×3 filters in an inception module of Figure 2.20 was replaced by 1×3 and 3×1 convolutional filters in Figure 2.21. In this paper they argued that the auxiliary classifiers did not affect accuracy more. By batch normalization of auxiliary branches, the, final classifier achieved high accuracy [110]. In the recent paper about GoogleNet architecture two models namely inception-v_4 and inception ResNet were proposed. Using the idea of residual connections of ResNet architecture [63] and inception modules [110] provided the recent architecture of GoogleNet. They proposed inception-v_4 without using residual connections and made it deeper by using more inception modules. In their proposed inception ResNet, instead of concatenation of filters, residual

connections were used and their proposed models decreased the computation of training [111].

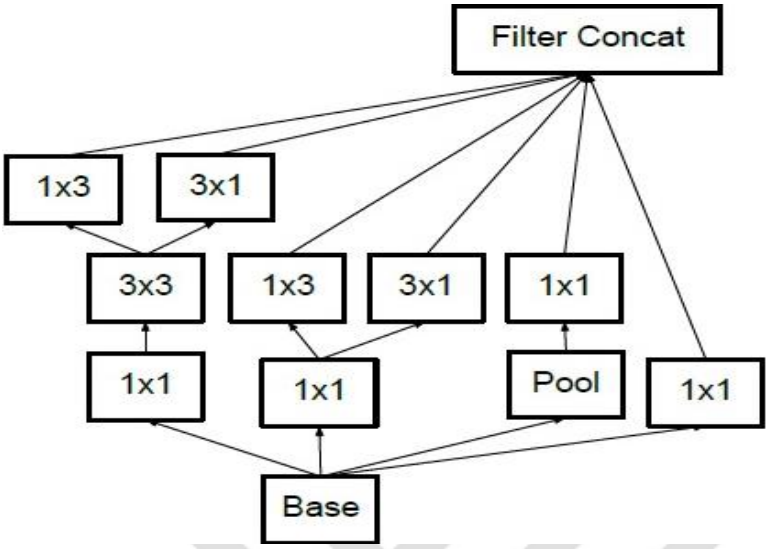

Figure 2.21. Factorized filters of inception module [110]

3. EXPERIMENTAL RESULTS

As mentioned in the introduction section, the lung cancer death toll is roughly 225 000 every year in the United States. National Institute of Health acknowledged the costs of care and diagnose of lung cancer in 2010 was 12 billion dollars. The main purpose is to provide tools for early diagnose and increase care services. By providing tools and data in cancer fields diagnose processes were improved impressively [112]. In biomedical fields using machine learning tools could help and accelerate experts in diagnosis of diseases. Each year different challenges in technological fields are organized about universal important problems. In March 2017 Data Science Bowl group [113] organized challenge to improve diagnosis of lung cancer. Dataset of lung CT scan images of Data Science Bowl and Kaggle is used for this thesis. In this section detail of lung CT scan dataset, AlexNet and GoogleNet architecture for classification of lung CT scan images and their results are described.

Dataset: Since deep learning algorithms in contrast with shallow algorithms achieved considerable and high performance on thousands and millions of data, a large dataset of lung CT scan images is used in this thesis in order to diagnose lung cancer by CNN as a deep learning method.

For classification task, lung CT scan images of Data Science Bowl and Kaggle challenge [113] are used. They gathered the large lung CT scans from different sources and this dataset is the first large lung CT scan images in data science field. The challenge contained two stages. Because we notified late, we could not take part in this challenge therefore, we just used the dataset for our this thesis. The lung CT scan dataset contains 285 058 low-dose CT scan images of 1595 patients in Digital Imaging and Communications in Medicine (DICOM) format. This dataset contains 85 138 samples as malignant label and 199 920 samples as benign label. Because of unbalanced samples of dataset, 59 497 samples of benign labels and 43 656 samples of malignant labels are used in this thesis. For calssification of lung CT scan images 49 599 benign samples and 34 267 malignant samples are considered as train dataset. Moreover, 9898 benign samples and 9389 malignant samples are considered as test dataset.

Preparing dataset: One of the main advantages of using CNN is independence of hand-crafted features and other preprocessing algorithms [114]. In this thesis preprocessing

methods which generally were used by shallow algorithms for feature extraction did not applied. Because of the high quality of DICOM formats, all images of medical systems (i.e. Xray, PET, CT scan) are in DICOM format [115].

Since the format of the CT scan images of lung dataset is medical format (DICOM), in this thesis, all images were converted to Portable Network Graphics (PNG) format. The Python code was applied in order to convert the DICOM format to PNG format. example of benign and malignant samples (in PNG format) are shown in Figure 3.1.

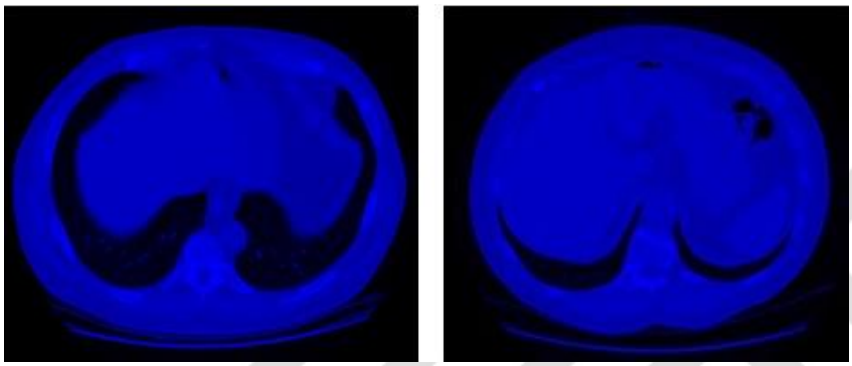

Figure 3.1. Example of malignant samples (left) and benign samples (right)

3.1. AlexNet Architecture For Classification Of Lung CT Scan Images

AlexNet architecture of CNN contains five convolutional layers, five pooling layers, two normalization layers and three fully connected layers. To determine the probability of two classes of dataset softmax is used as the last layer. To continue learning of the network Relu activation function is applied on the output of each convolutional layer and fully connected layer. Batch size (number of samples through one training cycle) and the learning rate of the network are determined 32 and 0.01 respectively.This model of CNN is trained in 30 epochs.

First, the input images by [227×227×3] size are fed to the network. In the first convolutional layer, 96 kernels (filters) by 11×11 filter size are applied on the input images. In the first convolutional layer, stride size is initialized by 4 pixels furthermore zero padding is not used in the first convolutional layer (zero padding=0). Each kernel is connected to the receptive field of the previous layer only (local connections). The output of the first convolutional layer obtained from Eq. 2.46. By considering W_1=227 as input size, F=11 as receptive field size, P=0 as zero padding, S=4 as stride size, W_2 =(((227-11)+(2×0))/4)+1=55 is the output

size. The size of the first covolutional layer will be [96×55×55]. This layer has 55×55×96=290 400 neurons and each neuron has 11×11×3=363 weights with one bias. Figure 3.2, shows filters by [96, 3, 11, 11] size and output of the first convolutional layer after applying filters by [96×55×55] size. After calulating covolutional operations the Relu nonlinearity activation function is used on the output of the first convolutional layer. Although the motifs learned by filters are not more obvious in the first layer, in deeper layers they will learn more features of image gradually. Local response normalization is used to make more brightness on output of convolutional layer.

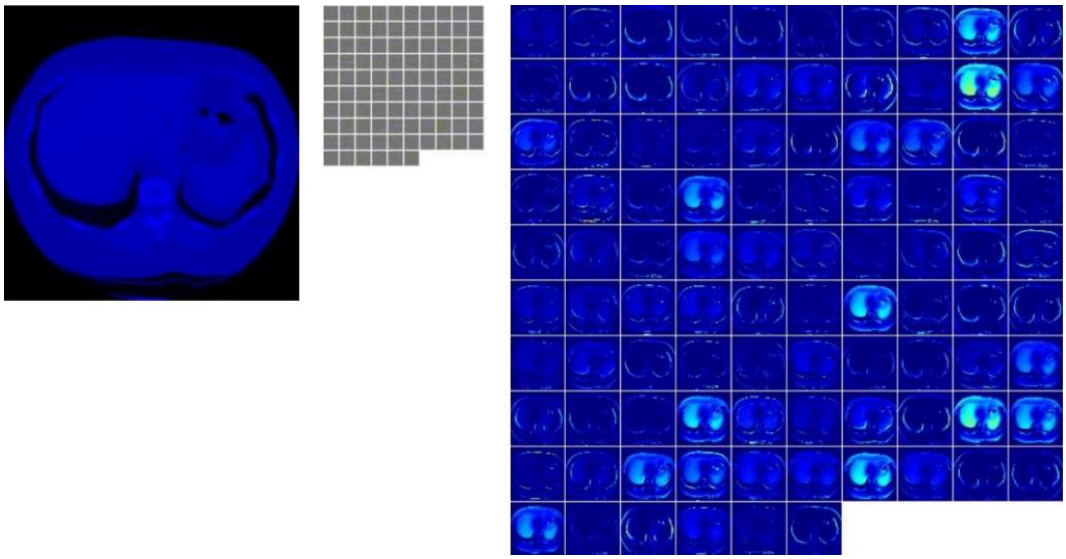

Figure 3.2. Input image (left), Applied filters (middle), the output of the first convolutional layer after applying filters by [96×55×55] size (right)

The size of the output of convolutional layer is not affected by normalization layer. Normalization layer makes the output othe convolutional layer more clear and bright. The first normalization layer by [96×55×55] size is shown by Figure 3.3.

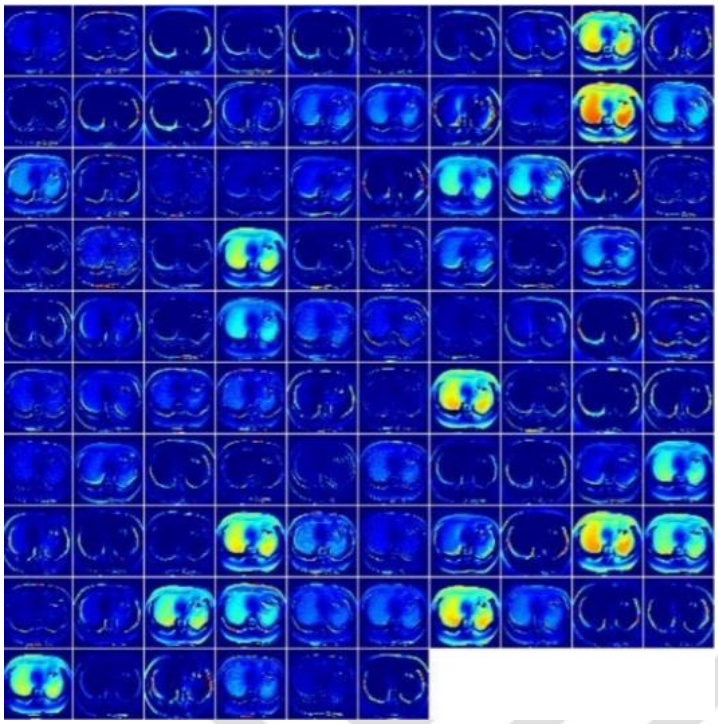

Figure 3.3. The first normalization layer by [96×55×55] size

The second main layer of AlexNet architecture is pooling layer. For subsampling of the output of convolutional layer, overlapping max pooling by 3×3 filter and two strides (S=2) is applied. The output of the first pooling layer is obtained by Eq. 2.54. By considering W_1=55 as input and F=3 as receptive field size, W_2= ((55-3)/2)+1=27 is the output size. Since pooling is applied on each layer and it causes the size of images to be subsample. Consequently,the depth size will be the same as the depth size of the previous layer.The output of pooling layer is [96×27×27]. The first pooling layer is shown by Figure 3.4, and it is obvious in pooling layer data dimension is reduced.

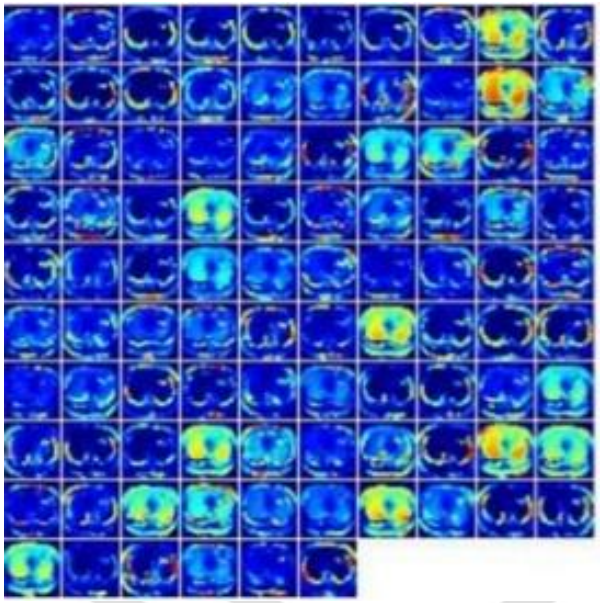

Figure 3.4. The first pooling layer by [96×27×27]

The next layer is the second convolutional layer. In this layer 256 kernels (filter) by 5×5 filter size and 1 for stride size are applied on the inputs by [96×27×27]size. Two pixels of zero padding is considered for the second convolutional layer. The output of layer is calculated as $W_2 = ((27\text{-}5+(2\times2))/1)+1=27$. The original size of the previous layer is restored by using 2 for zero padding and the output size is [256×27×27]. It is noticeable that in the original AlexNet architecture of CNN [91] due to lack of powerful GPUs, they have used two GPUs for training of the large imagenet dataset. In this architecture, the training model was divided into two section to benefit the power of two GPUs for training the whole dataset. Consequently, filters were applied on all depth of samples in two sections. Nowadays because of powerful GPUs, AlexNet architecture of CNN could be applied on one GPU.The size of the final output of the second convolutional layer is [256×27×27]. Relu activation function is applied on the output of the second convolutional layer.

The next step is take the advantage of the second response normalization layer. The dimension of samples (256) is not changed by normalization layer. The size of the second normalization layer is [256×27×27].

After normalization of the output of the second convolutional layer, the next pooling is applied. In the second poling layer, input images are fed to the overlapping max pooling with the filter size of 3×3 and stride of 2. Width and height of data dimension is reduced by pooling operations and the depth is not changed (256). The output of the second pooling

layer by Eq. 2.54 calculated as W_2 = ((27-3)/2)+1=13. The size of the output of this layer is [256×113.3×13]. The output of pooling layer is fed to the third convolutional layer of 384 kernels (filters) with 3×3 filter size. Stride size is 1 and moreover 1 zero padding is used. By using one pixel for zero padding the size of convolutional layer is not changed. Output is calculated as W_2 =(((13-3+(1×2))/1)+1=13. Output size of the third convolutional layer is [384×13×13]. Applying Relu activation function on the output of the convolutional layer is the next step.The next layer of AlexNet architecture is the forth convolutional layer. Input image are fed to the forth convolutional layer by applying 384 kernels (filters) by 3×3 size and 1 for both stride size and zero padding. The output (W_2) is the same as the previous layer by using 1 for zero padding and it is calculated as W_2= ((13-3+(1×2))/1)+1=13. Output size of the convolutional layer is [384×13×13]. Relu activation function is applied after convolutional layer.

For the fifth convolutional layer, 256 kernels with 3×3 filter size are applied. The stride size and the zero padding for both layers are 1. The output of the convolutional layer by applying 256 kernels by 3×3 size is [256×13×13] and it is shown by Figure 3.5. Relu activation function is applied on the output of the convolutional layer.

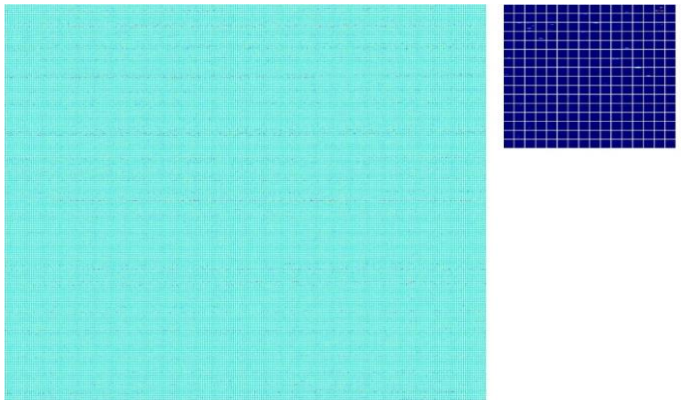

Figure 3.5. 256 kernels of 3×3 size (left), output of convolutional layer by [256×13×13] size (right)

Overlapping max pooling by 3×3 kernel size with stride size of 2 is the next layer. The output size of the pooling layer is [256×6×6] and it is shown by Figure 3.6.

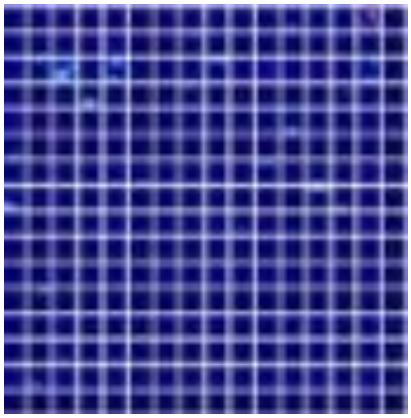

Figure 3.6. Output of the pooling layer by [256×6×6] size

By following AlexNet architecture, the output of the last layer by [256×6×6] size is fed to each of 4096 neurons of the first fully connected layer. The output of the first fully connected layer is fed to the second fully connected layer by 4096 neurons. After each fully connected layers, Relu activation functions is applied. In the third fully connected layer to calculate the probability of each label of lung dataset which includes benign and malignant labels, the output of the previous fully connected layer is fed to the two way softmax function. The first and the second fully connected layers are shown by Figure 3.7 and Figure 3.8, respectively. Two classes of lung CT scan images which are classified in the third fully connected are shown by Figure 3.9. Total learned parameters of lung CT scan images in AlexNet architecture is calculated as 56 876 418. Summary of layers and output size of lung images through training by AlexNet architecture are given by Table 3.1.

Figure 3.7. The first fully connected layer

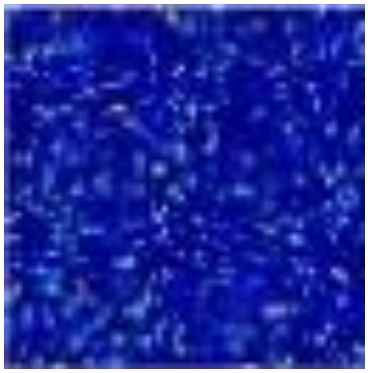

Figure 3.8. The second fully connected layer

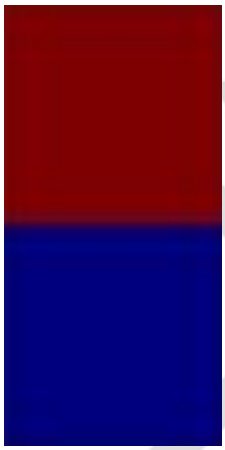

Figure 3.9. The third fully connected layer by softmax classifier

Table 3.1. Summary of AlexNet architecture in classification of lung CT scan images

Layer type	Number of kernels	Kernel size	Output size
Convolutional	96	11×11	96×55×55
Max pooling		3×3	96×27×27
Convolutional	256	5×5	256×27×27
Max pooling		3×3	256×13×13
Convolutional	384	3×3	384×13×13
Convolutional	256	3×3	256×13×13
Max pooling		3×3	256×6×6
Fully connected			4096×1×1
Fully connected			4096×1×1
Fully connected with softmax			2×1×1

3.2. GoogleNet Architecture For Classification Of Lung CT Scan Images

GoogleNet architecture, consists of nine different inception modules. 3(a), 3(b), 4(a), 4(b), 4(c), 4(d), 4(e), 5(a) and 5(b) are the 9 inception modules in GoogleNet architecture. At the end of the 3(b), 4(e) and 5(b) inception modules max pooling layer is applied and the output of the max pooling is fed to the next layer. Each inception module includes convolutional and pooling layers. Each filter in inception module comprises the part of information about the image. The output of each inception module is the concatenation of its layers that would be the input of the next inception module. Six convolutional layers and one max pooling layer are used in each inception module. Totally two local response normalization layers are used in this architecture. In all inception modules and the other convolutional layers, Relu activation functions are used. At the end of the architecture average pooling is used as a final pooling layer and on fully connected layer softmax classifier is applied. Batch size and the learning rate of the network are determined 32 and 0.01 respectively. The training of CNN in GoogleNet architecture for lung CT scan images is applied in 30 epochs. After two convolutional layers inception layers are applied consecutively. In this architecture input image by [224×224×3] size is fed the first convolutional layer by 64 kernels (filters) of 7×7 size with 3 for zero padding and 2 for stride size. By considering Eq. 2.46 and W_1=224 as input size, F=7 as receptive field size, P=3 as zero padding and S=2 as stride the output of the first convolutional layer is calculated as W_2= (((224-7)+(2×3))/2)+1=112.5≈ 112. (floor value of 112.5 is 112). In order to achieve the fractional of values, the ceiling and floor functions have been used in Caffe.

[64×112×112] is the size of the first convolutional layer. This layer has 112×112×64=874 496 neurons and each neuron has 7×7×3=147 weights with one bias. Figure 3.10, shows the input image by [224×224×3] size (left image), 64 filters by 7×7 on the same depth of image, [64, 3, 7, 7] (middle image) and the output of the first convolutional layer after applying kernels on each feature maps by [64×112×112] size. For nonlinearity Relu activation function is applied on the first convolutional layer.

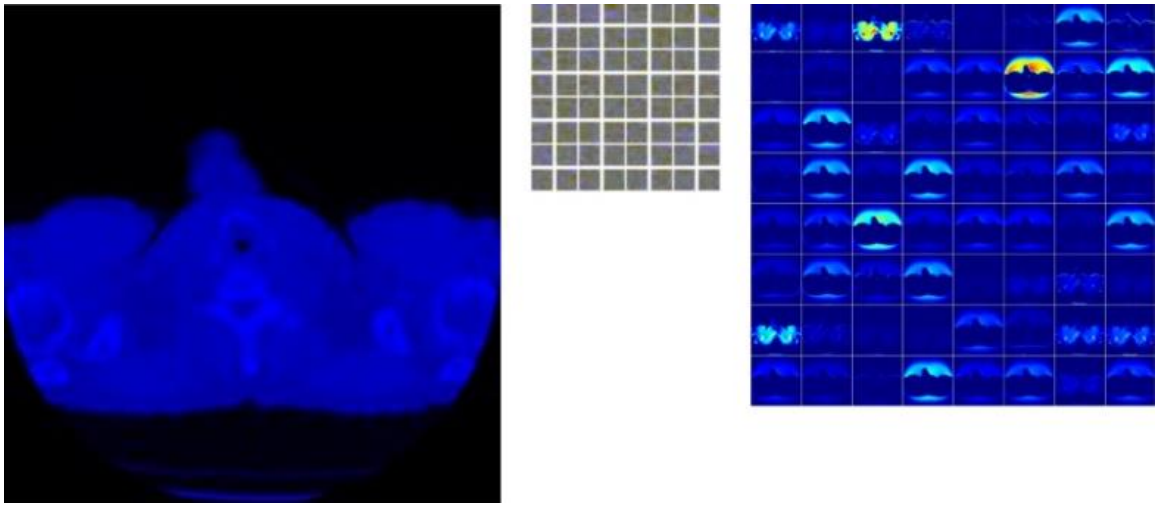

Figure 3.10. Input image (left), 64 kernels of 7×7 sizes (middle), output of convolutional layer by [64×112×112] size (right)

The second layer of GoogleNet architecture is the first pooling layer. Overlapping max pooling by 3×3 filter size with stride size of 2 is applied for subsampling. By considering W_1=112 as input, F=3 as receptive field size and stride (S=2), the output of the first pooling layer is obtained by Eq. 2.54. The ceiling value is concidered in pooling layer.

W_2= ((112-3)/2)+1=55.5≈56. The output of pooling layer is [64×56×56]. Although the result of pooling layer leads to subsample the size of image, the depth is the same as the depth of previous layer. The first pooling layer of the architecture is shown by Figure 3.11.

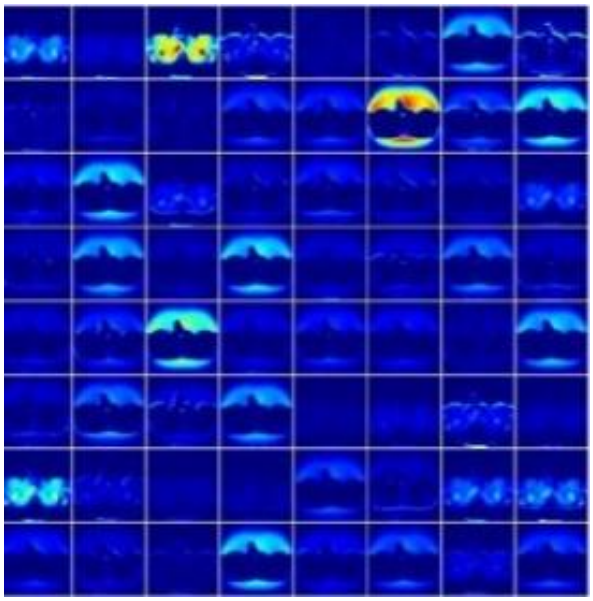

Figure 3.11. First pooling layer by [64×56×56] size

After the first pooling layer to make more brightness on the output of pooling layer, local response normalization layer is applied. The output of the first normalization layer is the same as the previous pooling layer by [64×56×56] size and it does not affect the overall size of the previous layer. The first normalization layer is shown by Figure 3.12.

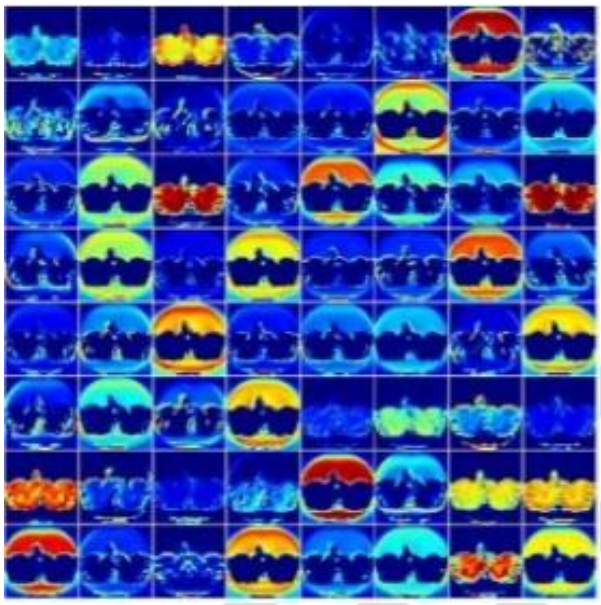

Figure 3.12. First normalization layer by [64×56×56] size

In the second convolutional layer, the dimension of channels is reduced by using small kernels of 1×1. Small kernels lead to creating fewer parameters.64 Kernels of 1×1 size with a stride of 1 and without any zero padding create the output by [64×56×56] which is calculated as W_2= ((56-1)/1)+1=56. Relu nonlinearity activation function is applied. For the second step of the second convolutional layer 192 kernels of 3×3 size with stide size of 1 and one zero pading is applied. The output (W_2) is calculated by Eq. 2.46 and it is calculated as W_2=((56-3+(2×1))/1)+1=56, therefore, the output size is [192×56×56]. The Relu activation function is applied on the output of the second layer. 64 filters of 1×1 size and the output of reduced convolutional layer are shown by Figure 3.13. 192 filters of 3×3 size and the output of the second convolutional layer by [192×56×56] size are shown by Figure 3.14.

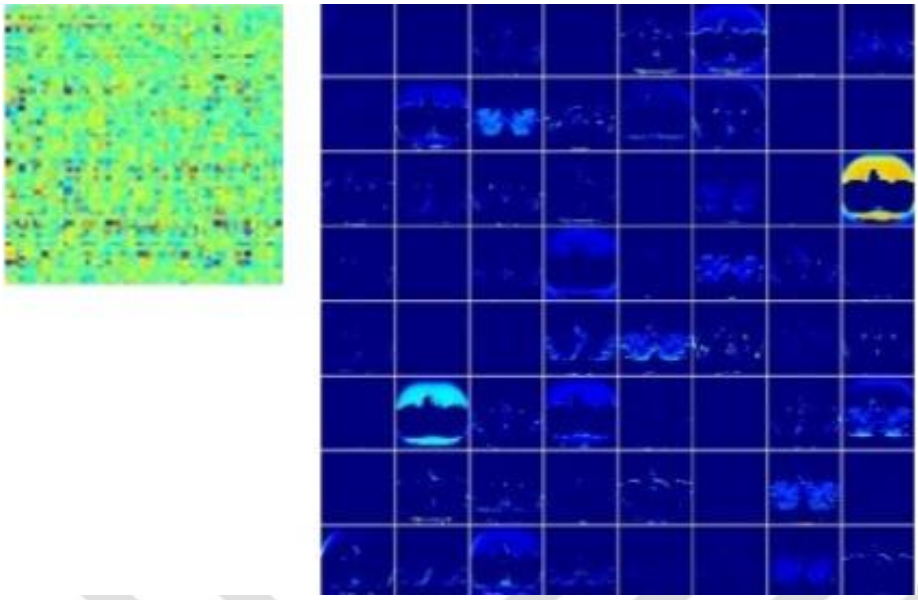

Figure 3.13. 64 kernels of 1×1 size (left), output of convolutional layer by [192×56×56] size (right)

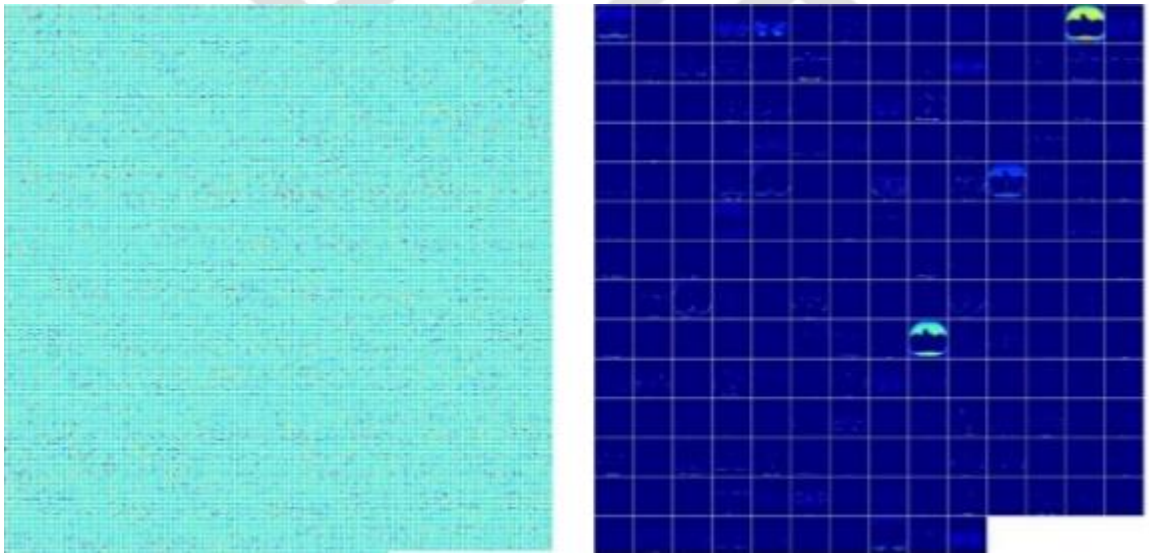

Figure 3.14. 192 kernels of 3×3 size (left), output of convolutional layer by [192×56×56] size (right)

After applying the Relu activation function normalization is applied on the output of the convolutional layer. The second normalization layer makes the output of the convolutional layer bright and its size is the same as the size of the convolutional layer. The second normalization layer by [192×56×56] size is shown by Figure 3.15.

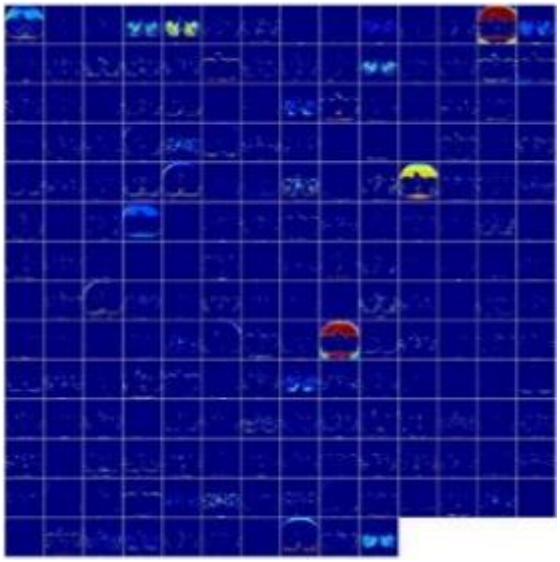

Figure 3.15. Output of the second normalization layer by [192×56×56] size

For subsampling of the output of the previous convolutional layer, the second overlapping max pooling by 3×3 filter size with stride size of 2 is applied. The output of pooling layer is calculated by Eq. 2.54 as W_2= ((56-3)/2)+1=27.5≈28.Therefore, the output size of pooling layer is [192×28×28]. It is noticiblae that the pooling operation dos not change the depth of its input. The second pooling layer is shown by Figure 3.16.

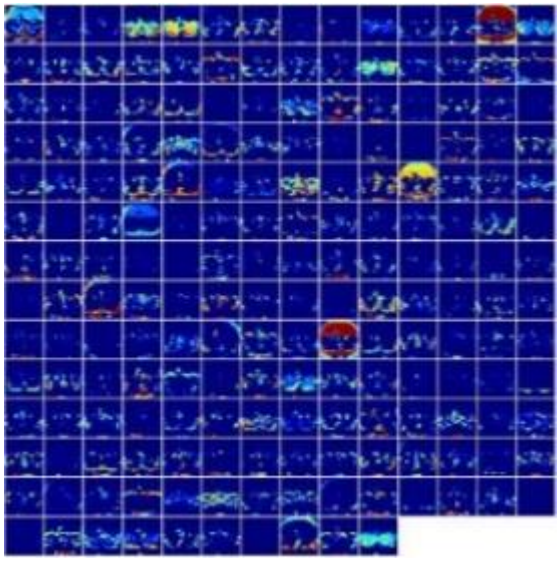

Figure 3.16. Output of the second pooling layer by [192×28×28] size

After reducing the dimension of the input image, nine inception layers are applied. In all inception modules the main purpose is using small kernels to prevent overfitting and before using larger kernels the tiniest kernels by 1×1 size is applied. After all 1×1, 3×3 and 5×5

kernels for nonlinearity Relu activation function is used. Beside convolutional layers, one max pooling layer is applied to achieve the reasonable result. At the end of each inception module, all small kernels are concatenated as one part that consists of all information about a part of the image. The concatenated part is an input of the next inception module. In inception module 3(a), 64 tiny kernels (filter) by 1×1 size with stride size of 1 and without any zero padding is used.

The output of 1×1 kernel size is calculated by Eq. 2.46 as $W_2=(((28-1)+0)/1)+1=28$. The output of the convolutional of 1×1 kernel size is [64×28×28]. Relu activation function is applied. 64 kernels by 1×1 size and the output by [64×28×28] size are shown by Figure 3.17.

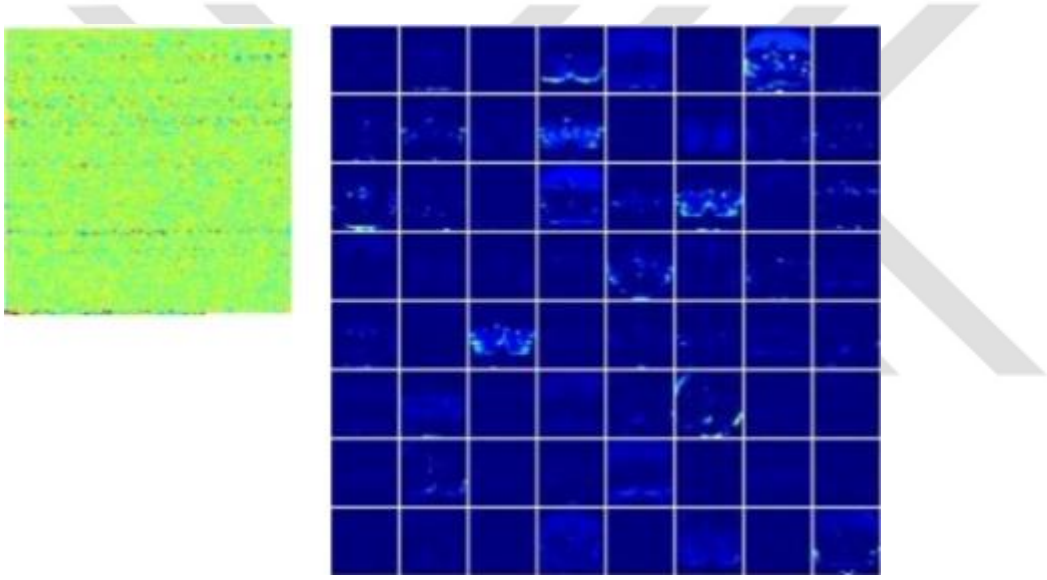

Figure 3.17. 1×1 kernel size (left), output of the convolutional by [64×28×28] size (right)

The first inception module that includes four convolutional layers is applied on its previous layer (pooling layer). The input of the other inception modules are the output of the concatenation of the previous inception module. By considering the inception module model, after applying the single 1×1 kernel, the second step of inception module is applying kernels by 1×1 and 3×3 on previous pooling layer. The output of convolutional by applying 96 kernels by 1×1 size with stride size of 1 and without zero padding on the output of the previous layer is calculated as $W_2=(((28-1)+0)/1)+1=28$. Therefore, the output of convolutional of 1×1 kernel size is [96×28×28]. The Relu activation function is applied on the output of the convolutional layer. After applying the tiny kernel by 1×1 size, 28 larger kernels by 3×3 size with 1 for stride size of and with 1 for zero padding, is applied. The output of convolutional layer is calculated as $W_2=(((28-3)+(2\times1))/1)+1=28$. Therefore, the

output size is [128×28×28]. Relu activation function is applied on the output of the second step of the 3(a) inception module. 96 kernels by 1×1 size and its output of convolutional layer by [96×28×28] size are shown by Figure 3.18. 128 kernels by 3×3 size and its output of convolutional layer by [128×28×28] size are shown by Figure 3.19.

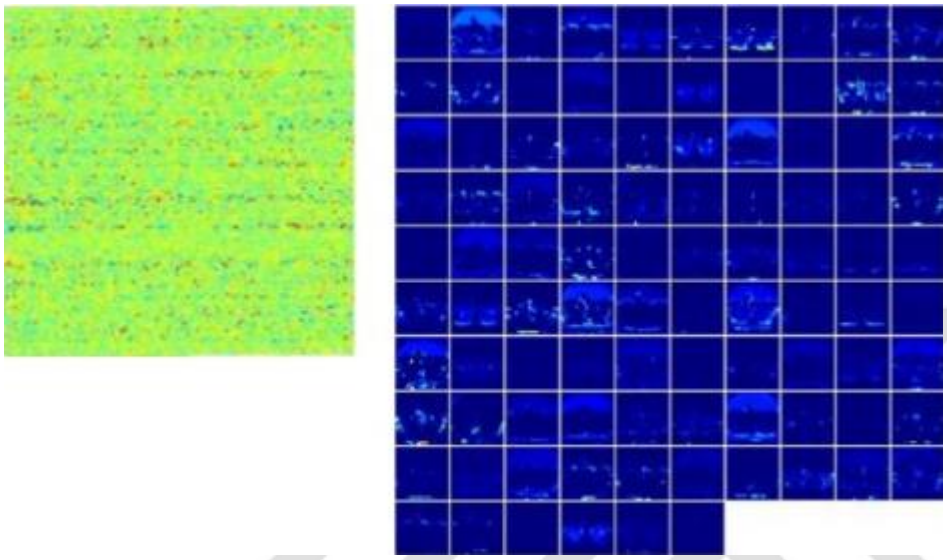

Figure 3.18. 96 kernels of 1×1 (left), output of convolutional by [96×28×28] size (right)

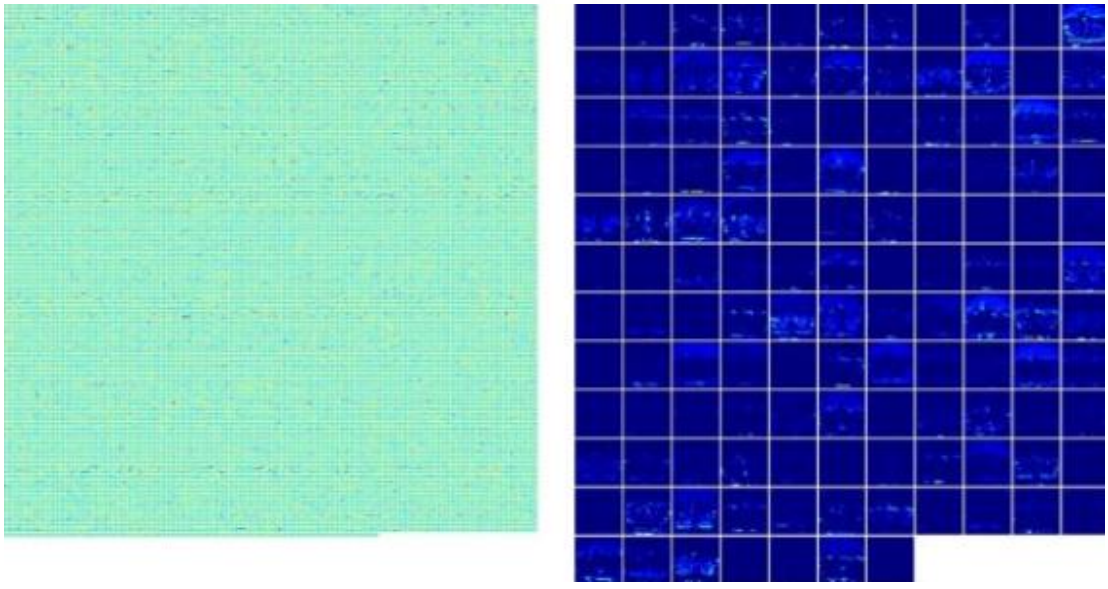

Figure 3.19. 128 kernels of 3×3 (left), output of convolutional by [128×28×28] size (right)

The third part of the 3(a) inception module is applying 1×1 and 5×5 kernels on its previous pooling layer. First, the tiny 16 kernels by 1×1 size with stride size of 1 and without zero padding and next 32 kernels by 5×5 size with stride size of 1 and with 2 for zero padding are applied. After each convolutional Relu activation function is applied. The output size of the

1×1 convolutional layer is [16×28×28] and the output size of the 5×5 convolutional layer is [32×28×28].16 kernels by 1×1 size and its convolutional output by [16×28×28] size are shown by Figure 3.20. 32 kernels by 5×5 size and its convolutional output by [32×28×28] size are shown by Figure 3.21.

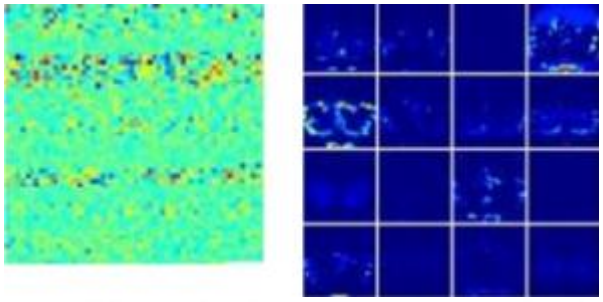

Figure 3.20. 16 kernels of 1×1 (left), output of convolutional by [16×28×28] size (right)

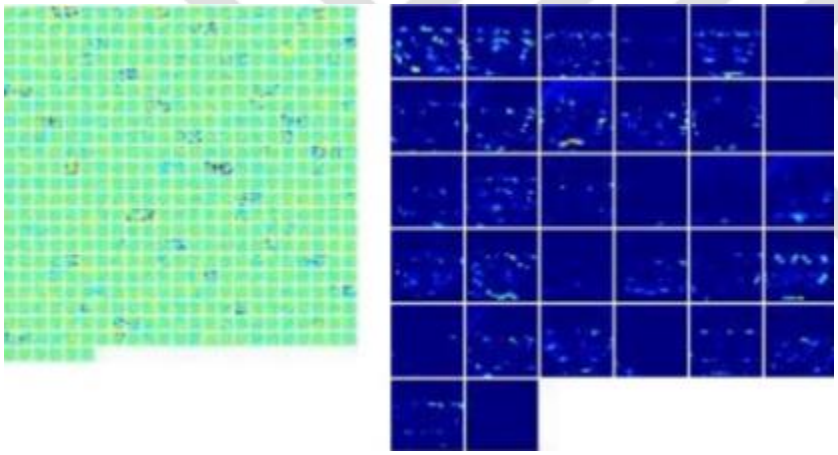

Figure 3.21. 32 kernels of 5×5 (left), output of convolutional by [32×28×28] size (right)

The fourth part of 3(a) inception module is applying the max pooling layer.The output size of max pooling by 3×3 kernel size with 1 for stride size and 1 for zero padding, is [32×28×28]. Max pooling layer is shown by Figure 3.22. Convolutional layer by 128 kernels of 1×1 size is the last operation of the fourth part of 3(a) inception module. The output of the convolutional layer is [32×28×28] and is shown by Figure 3.23.

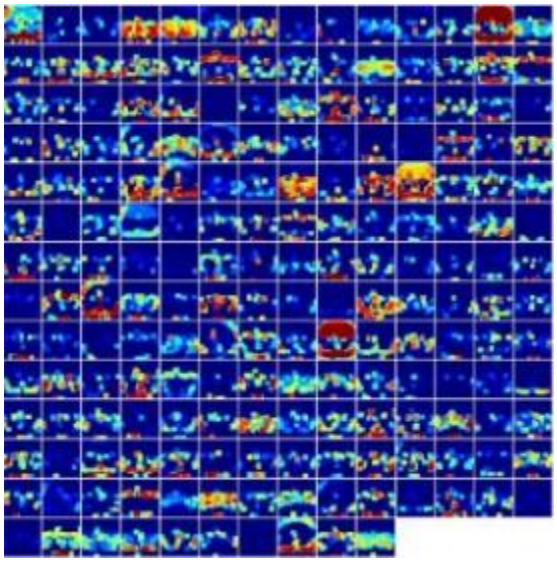

Figure 3.22. Output of pooling layer of inception 3(a) by [32×28×28] size

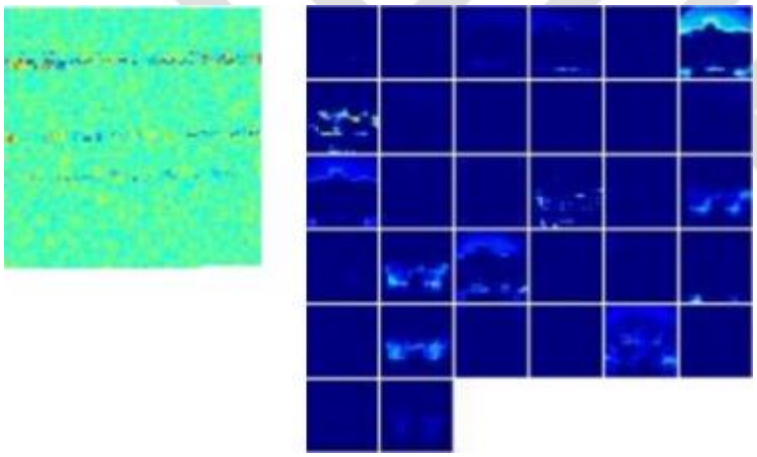

Figure 3.23. 128 kernels of 1×1 (left), output of convolutional by [32×28×28] size (right)

The last step is the concatenation of all layers in inception module 3(a). In concatenation layer sum of all depth of inception module is calculated as 64,128 and 32 for kernels by 1×1, 3×3 and 5×5 size respectively. 32 kernels for pooling layer is considered. Sum of all depths (256) calculated as the depth of last output of inception module. The output of concatenation of all layers in inception module 3(a) by [256×28×28] size is fed to the second inception module 3(b). the output of the inception module 3(a) is shown by Figure 3.24.

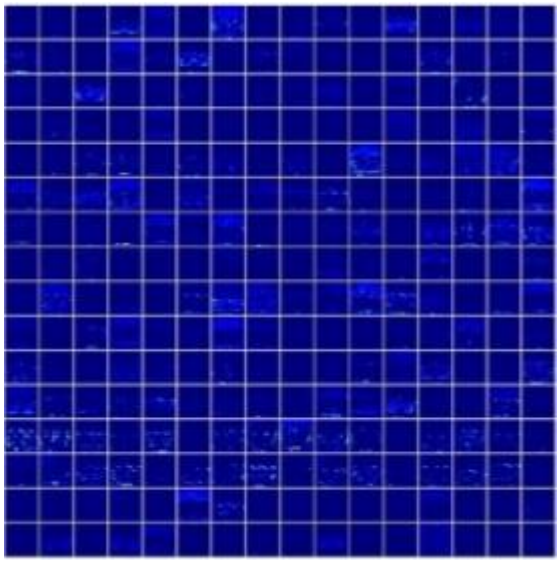

Figure 3.24. The output of inception module 3(a) by [256×28×28] size

Following inception module model, in inception module 3(b) the first 128 kernels by 1×1 size with stride size of 1 and without zero padding are applied on the output of the previous layer. The size of the convolutional layer of 1×1 kernels is [128×28×28]. Relu activation function is applied on the output of the first convolutional layer of inception module 3(b).

In the second step of inception module 3(b), 1×1 and 3×3 kernels are applied on the output of the previous layer by [256×28×28] size. The output of the convolutional layer by 128 kernels by 1×1 size is [128×28×28]. Relu activation function is applied on the output of the convolutional layer of the 1×1 kernels. In the second step of the inception module 3(b) after 1×1 kernels, 192 kernels by 3×3 size are applied on the output of the convolutional layer of 1×1 kernels. The output size of the convolutional layer of 3×3 kernel size is [192×28×28]. On the output of the convolutional by 3×3 kernel size, Relu activation function is applied.

In the third step of inception module 3(b), 1×1 and 5×5 kernels are applied on the output of the previous layer [256×28×28] .The size of the output convolutional of the 32 kernels by 1×1 size is [32×28×28] For nonlinearity, Relu activation function is applied on the output of the convolutional layer. On the output of the 1×1 convolutional layer, 96 kernels by 5×5 size with a stride of 1 and zero padding of 2 are applied. The output size of the convolutional layer is [96×28×28]. Relu activation function is applied on the output of the convolutional layer.

The fourth step of the second inception module 3(b) is applying max pooling. In this layer max poling by 3×3 kernel size with 1for stride size and zero padding is applied on the output of the previous layer [256×28×28]. Max pooling operation does not affect the depth of input and the output of the max pooling layer is calculated as W_2=(((28-3)+(2×1))/1)+1=28. The size of max pooling layer is [256×28×28]. Because of the used stride size in max poling output size is not changed.

Following the inception module model in GoogleNet architecture in the fourth step after pooling layer a convolutional layer is applied. 64 kernels by 1×1 size are applied on the output of the max poling layer. The output of the convolutional layer is [64×28×28] kernels by 1×1 size and the output of the convolutional layer are shown by Figure 3.25.

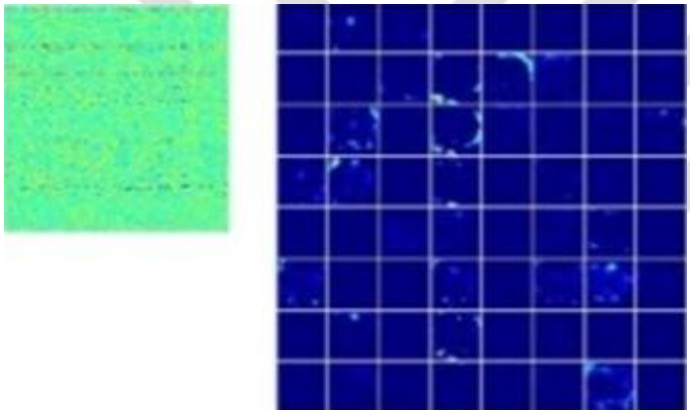

Figure 3.25. 64 kernels by 1×1 size (left), convolutional layer by [64×28×28] size (right)

After applying the Relu activation function, concatenation of all layers of inception module 3(b) is calculated as the output of the inception 3(b). Sum of 128, 192 and 96 for kernels by 1×1, 3×3 and 5×5 sizes and 32 for kernels of pooling layer is calculated as the depth of inception module 3(b) as 480. After applying max pooling layer on the output of concatenation of all layers of inception module 3(b), it is fed to the inception module 4(a). The output of the max pooling layer by 3×3 kernel size with stride size of 2 is achieved as [480×14×14].

In inception module 4(a), the first step is applying 192 kernels by 1×1 size on the input of [480×14×14] size. [192×14×14] is the output size of the convolutional layer. Relu function is applied on the output of the convolutional layer. In the second step of inception module 4(a) before applying kernels by 3×3 size, kernels by 1×1 size are applied. The output of the

96 kernels by 1×1 size with stride size of 1 on input by [480×14×14] size is [96×14×14]. After applying Relu activation function 208 kernels by 3×3 size with 1 for stride size and zero padding are applied on the output of the convolutional of 1×1 size. [208×14×14] is the output of the convolutional which is achieved by 3×3 kernel size. Relu activation function is used for nonlinearity. The third step is applying 1×1 and 5×5 kernels and using Relu activation function after each output convolutional layer. The output of the 16 Kernels by 1×1 size on input by [480×14×14] size is [16×14×14]. The output of the 48 kernels by 5×5 size with 2 for zero padding on the input by [16×14×14] size is [480×14×14].

The fourth step is applying max pooling on the output of the previous inception module and 1×1 kernel size of the convolutional layer. The output of the applying max pooling by 3×3 kernel size with 1 for zero padding and 1 for stride size is [480×14×14] and the output of the convolutional layer by 64, 1×1 kernel size is [64×14×14]. Relu activation function is applied on the output of the convolutional layer.The total output size of inception module 4(a) is achieved by concatenation of its all convolutional and pooling layers. The sum of depths is calculated as 192+208+48+64=512. The output size of the concatenation of inception module 4(a) is [512×14×14].

In the first step of inception module 4(b), 160 kernels by 1×1 size on the input of [512×14×14] size are applied [160×14×14] is the output size of the convolutional layer. Relu function is applied on the output of the convolutional layer. In the second step of inception module 4(b) kernels by 1×1 size are applied before applying kernels by 3×3 size. The output of the 112 kernels by 1×1 size with stride size of 1 on input by [512×14×14] size is [112×14×14]. After applying Relu activation function 224 kernels by 3×3 size with 1 for zero padding are applied on the output of the convolutional of 1×1 size. [224×14×14] is the output of the convolutional which is achieved by 3×3 kernel size. Relu activation function is used for nonlinearity. In the third step 1×1 and 5×5 kernels and Relu activation function are applied after each output of the convolutional layer. The output of the 24 Kernels by 1×1 size on input by [512×14×14] size is [24×14×14]. The output of the 64 kernels by 5×5 size with 2 for zero padding on the input by [24×14×14] size is [64×14×14]. Relu activation function is applied on the output of the convolutional layer as well. In the fourth step max pooling by the 3×3 kernel with 1 for zero padding and 1 for stride size is applied on the previous inception model by [512×14×14] size and its result is [512×14×14] In the convolutional layer of this step, 64 kernels by 1×1 size are applied on the input by

[512×14×14] size and the output of convolutional layer is [64×14×14] . On the output of the convolutional layer Relu activation function is applied.The total output size of inception module 4(b) is claculated by concatenation of its pooling layer by 64 kernels,160 kernels for 1×1, 224 kernels for 3×3 and 64 kernels for 5×5 convolutional layers. Sum of depths is 512. The size of the concatenation of inception module 4(b) is [512×14×14].

In the first step of inception module 4(c) of the convolutional layer 128 kernels by 1×1 size are applied on the output of the last inception module by [512×14×14] size. The size of the output of the convolutional layer is [128×14×14]. After the first convolutional layer of the inception module 4(c), Relu activation function is applied. In the second step of inception module 4(c) before applying 3×3 kernels, 128 kernels by 1×1 size are applied on the output of the previous inception module. [128×14×14] is the output of the 128 kernels by 1×1 size. After applying Relu activation function on the output of the convolutional with 1×1 kernels, 256 kernels by 3×3 size with 1 for zero padding are applied. [256×14×14] is the output of the convolutional achieved by 3×3 kernel size. After convolutional layer, Relu activation function is used for nonlinearity. After the convolutional layer of the second step of inception module 4(c), Relu activation function is applied. In the third step of the inception module 4(c) output of the previous inception modules by [512×14×14] size is fed to the 24 kernels by 1×1 size before applying 5×5 kernels. Convolutional layer of 1×1 kernels achieved the output by [24×14×14] size. Relu activation function is also used after convolutional layer. Subsequently 64 kernels by 5×5 size with 2 for zero padding is applied on the input by [24×14×14] size as the next convolutional layer. The output of the convolutional layer is [64×14×14] . Relu activation function is used after convolutional layer. In the fourth step of the inception module 4(c) max pooling by 3×3 kernel with 1 for zero padding and 1 for stride size is applied on the output of the previous inception model by [512×14×14] size and its result is [512×14×14]. 64 kernels by 1×1 size are applied on [512×14×14] as convolutional layer and the output of convolutional layer is [64×14×14]. The last Relu of the inception module 4(c) is applied after the last convolutional layer.

The last step of the inception module 4(c) is depth calculation by concatenation of pooling layer's depth and depths of the convolutional layers by 1×1, 3×3 and 5× 5 kernels. Sum of 64 kernels of the max pooling layer, 128 kernels for 1× 1 , 256 kernels for 3× 3 and 64 kernels for 5× 5 is calculated 512 as the depth of inception module 4(c). [512×14×14] is the total size of inception module 4(c).

The next inception module in GoogleNet architecture is inception module 4(d). Similar to the other modules of GoogleNet architecture it follows the process of inception modules. In the first step of the inception module 4(d), on the output of the previous module by [512×14×14] size, 112 kernels by 1×1 size are applied. The output of the convolutional layer by using 1×1 kernels is [112×14×14]. Relu activation function is applied after the convolutional layer of the first step. In the second step of inception module 4(d), 144 kernels by 1×1 size are applied on the output of the previous module by [512×14×14] and the output is [144×14×14]. Relu activation function is applied on the output of the convolutional layer.

After 1×1 kernels, 288 kernels by 3×3 size with 1 for zero padding are applied on [144×14×14] and the convolutional layer's output is achieved by [288×14×14] size. Relu activation function is used after convolutional as well. The third step is applying 32 kernels by 1×1 size on the input by [512×14×14] size and 64 kernels by 5×5 size with 1 for zero padding on the output of 1×1 kernels. The output size of the convolutional by using 1×1 kernel is [32×14×14] and the output of the convolutional by using 5×5 kernels is [64×14×14]. After each convolutional layer, Relu activation function is applied. By following the GoogleNet architecture the fourth step is applying max pooling layer with 5×5 Kernel size and 1 for both stride and zero padding on the input by [512×14×14] size. After applying max pooling layer in the fourth step, 64 kernels by 1×1 size are applied on [512×14×14] as a convolutional layer and the output of convolutional layer is [64×14×14]. Relu activation function is applied after convolutional layer as well.The depth of the inception module 4(d) is the sum of 64 for kernels of max pooling layer, 112 kernels for 1×1 Kernels, 288 kernels for 3×3 and 64 kernels for 5×5. The concatenation of depths in inception module 4(d) is [528×14×14].

In the first step of inception module 4(e) the output of the convolutional layer by 256 kernels of 1×1 size on the output of the previous layer by [528×14×14] is [256×14×14]. Relu activation function is applied after convolutional layer.The second step is applying 160 kernels by 1×1 size on the input by [528×14×14] size. Applying 320 kernels by 3×3 size and with 1 for zero padding on the output of the 160 kernels by 1×1 is the next section of the second step. The output size of 160 by 1×1 kernels in the convolutional layer is [160×14×14] and the output size of convolutional layer by 320 kernels with 3×3 size is [320×14×14] . Relu activation function is used after both convolutional layers by 1×1 and 3×3 kernels.In the third step of inception module 4(e) before applying kernels by 5×5 size,

32 kernels by 1×1 size are applied on the input by [528×14×14] size. The output size of the convolutional layer is [32×14×14] and the output size of the convolutional layer by applying 128 kernels with 5×5 size and with 2 for zero padding on the input by [320×14×14] size is [128×14×14]. Relu activation function is applied after both convolutional layers by 1×1 and 5×5 kernels. By following GoogleNet architecture the fourth step in the inception module 4(e) is applying max pooling and convolutional layer by 1×1 kernel size as well. Max pooling by 3×3 kernel size with 1 for both zero padding and stride size is applied on the input by [528×14×14] size. The size of the output of the max pooling layer is [528×14×14] and the size of the output of the convolutional layer by 128 kernels with 1×1 size on the output of the max pooling layer is [128×14×14] . After convolutional layer, Relu activation function is applied. The depth of output of the inception module 4 (e) is the concatenation of the depth of max pooling layer and the depth of the convolutional layers. Sum of 256, 320,128 and 128 for kernels of 1×1, 3×3, 5×5 of convolutional layers and pool layer is 832 as the depth of inception module 4(e). The output size of inception module 4(e) is [832×14×14]. The output of the inception module 4(e) is fed to the max poling layer then it is fed to the next inception module. Max poling layer by 3×3 kernel size with 2 for stride size is applied on the output of the concatenation by [832×14×14] size.

The first step of the inception module 5(a) is applying 256 kernels by 1×1 kernels on the input by [832×14×14] size. The output of the convolutional layer is calculated as [256×7×7]. Relu activation function is applied on the output of the convolutional layer. In the second step of the inception module 5(a) 160 kernels by 1×1 size are applied on the output of the previous layer by [832×7×7] size. After applying 1×1 kernels, 320 kernels by 3×3 size with 1 for zero padding are applied on the output of the convolutional by 1×1 kernel size. The output of the convolutional of 320 kernels by 3×3 size is [320×7×7]. After both convolutional layer of the inception module 5(a), Relu activation function is applied. In the third step of the inception module 5(a) before applying kernels by 5×5 size, 32 kernels by 1×1 size are applied on the output of the previous layer by [832×7×7] .The output size of the convolutional layer is [32×7×7]. Relu activation function is applied after convolutional layer. 128 kernels by 5×5 size with 2 for zero padding are applied after convolutional layer by 1×1 kernel size. The output of the convolutional layer of 128 kernels by 5×5 size on the input by [32×7×7] size is [128×7×7]. Relu activation function is applied after convolutional layer as well. In the fourth step of the inception module 5(a) max pooling by 3×3 kernel size with 1 for both stride size and zero padding is applied on the output of the previous layer by

[832×7×7] size. The output size of the max pooling layer is [832×7×7] . After max pooling layer one convolutional layer of 128 kernels by 1×1 size is aplied and the output size of the convolutional layer is [128×7×7] . After convolutional layer Relu activation function is applied. Concatenation of the all depths of convolutional layers and max pooling layer in inception module 5(a) is the depth of module. Sum of 256,320, 128, 128 for depth of convolutional by 1×1, 3×3, 5×5 kernels and max pooling layer is calculated as 832 for depth of inception module 5(a). The output size of the inception module 5(a) is [832×7×7].

The last inception module of GoogleNet architecture is inception module 5(b). By following all inception modules of GoogleNet architecture the first step is applying convolutional layer by 1×1 kernel size on the output of the previous layer by [832×7×7] size. The output of the convolutional of 384 kernels by 1×1 size is [384×7×7] The convolutional layer of 384 kernels by 1×1 size and its output are shown by Figure 3.26. Relu activation function is applied after convolutional layer.

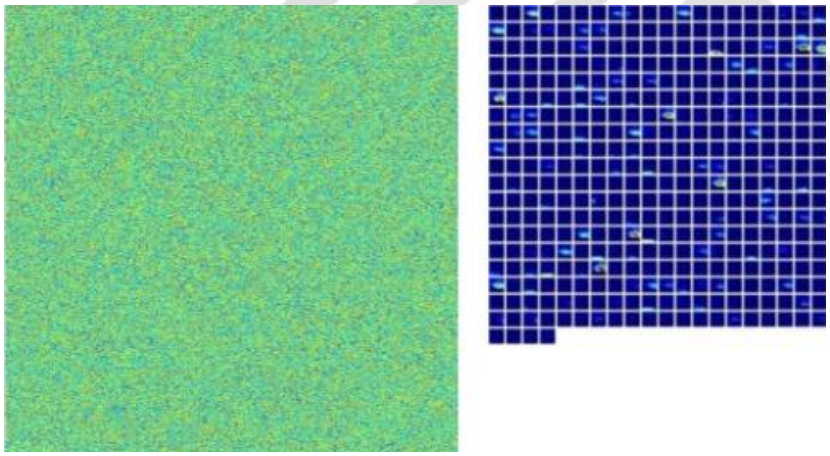

Figure 3.26. 384 kernels by 1×1 size (left), output of convolutional by [384×7×7] size (right)

In the second step of the inception module 5(b) before applying convolutional by 3×3 size 192 kernels by 1×1 size are applied on the output of previous layer by [832×7×7] size. The size of the output of the convolutional by the 3×3 kernel is [192×7×7] size and it is shown by Figure 3.27.

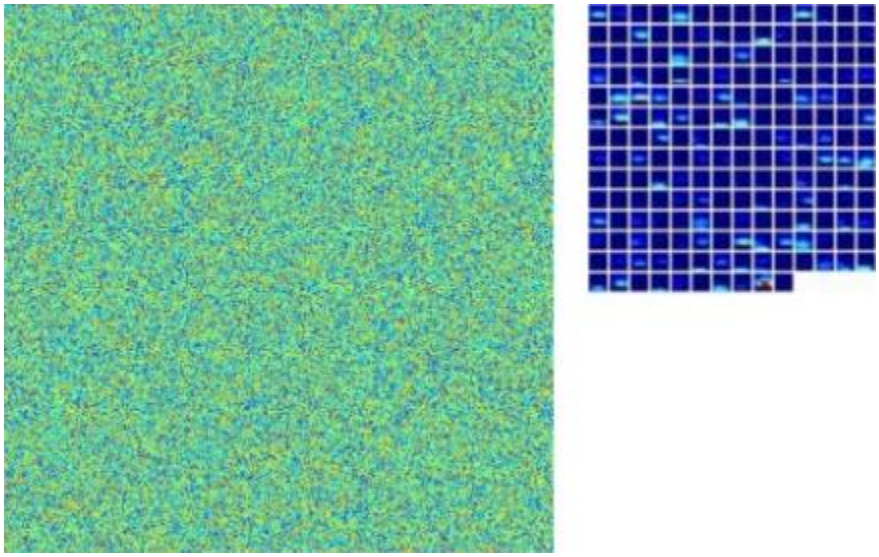

Figure 3.27. 192 kernels by 1×1 size (left), output of convolutional by [192×7×7] size (right)

After applying convolutional layer by 1×1 kernel size, 384 kernels by 3×3 size with 1 for zero padding are applied on the output of the previous convolutional layer by 1×1 kernel size. The output size of the convolutional layer is [384×7×7] and it is shown by Figure 3.28. After both convolutional layers, Relu activation function is applied.

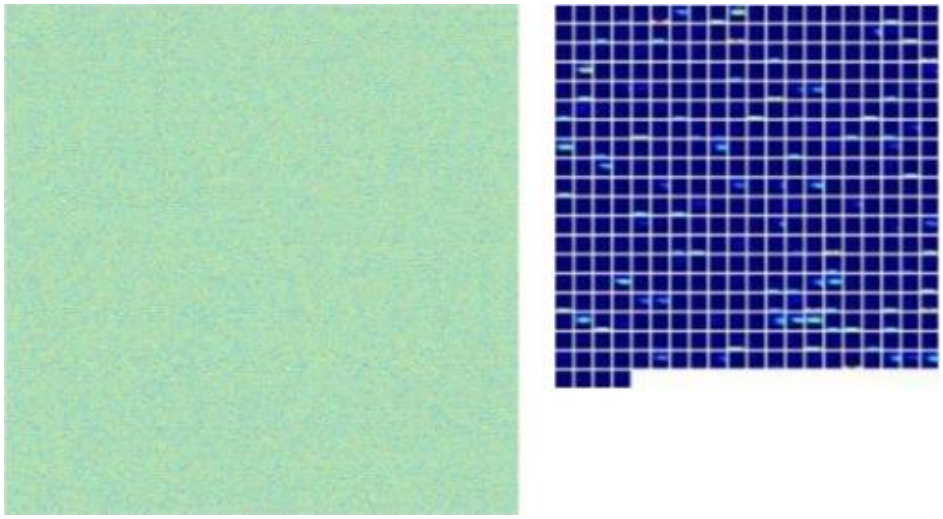

Figure 3.28. 384 kernels by 3×3 size (left), output of convolutional by [384×7×7] size (right)

The third step of the inception module 5(b) includes convolutional layers by 1×1 and 5×5 kernel size. 48 kernels by 1×1 size are applied on the output of the previous layer and the output size of the convolutional is [48×7×7] Convolutional of 48 kernels by 1×1 size and its output are shown by Figure 3.29. 128 kernels by 5×5size with 2 for zero padding are applied on the last convolutional layer by [48×7×7] size. The output size of convolutional by 5×5

kernel size is [48128×7×7]. A convolutional layer of 128 kernels by 5×5 size and its output are shown by Figure 3.30. After both convolutional layers, Relu activation function is applied.

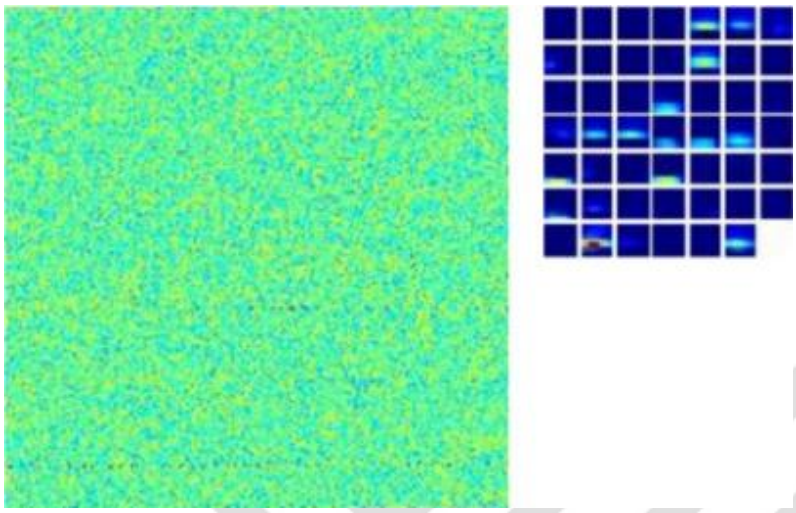

Figure 3.29. 48 kernels by 1 × 1 size (left), output of convolutional by [48×7×7] size (right)

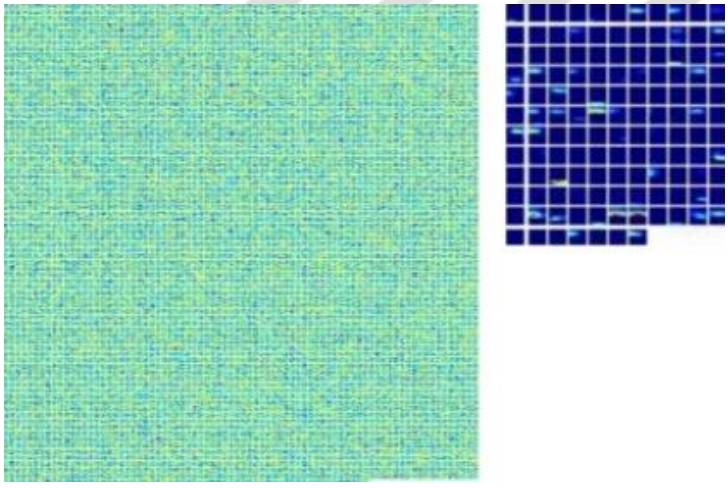

Figure 3.30. 128 kernels by 5×5 size (left), output of convolutional by [128×7×7] size (right)

In the fourth step of inception module 5(b), max pooling and convolutional layer are applied. The output size of the max pooling layer by 3×3 kernel size with 1 for both zero padding and stride on the output of the previous layer is [832×7×7]. Max pooling layer is shown by Figure 3.31. For convolutional layer, 128 kernels by 1×1 size are applied on the output of the max pooling layer. The output of the convolutional layer of 128 kernels by 1×1 size is [128×7×7] and it is shown by Figure 3.32. Relu activation function is applied after convolutional layer.

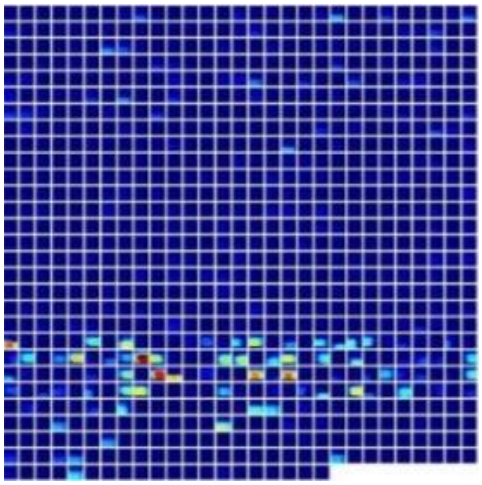

Figure 3.31. Output of the max pooling layer by [832×7×7] size

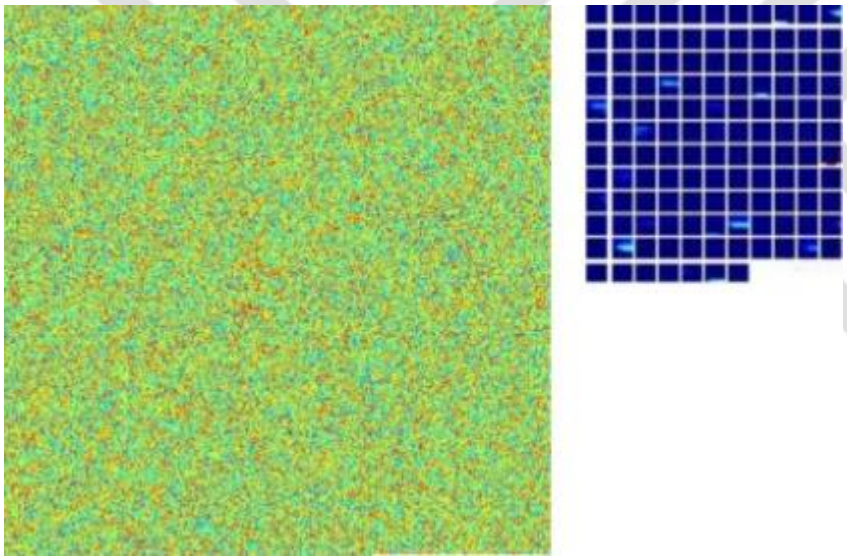

Figure 3.32. 128 kernels by 1×1 size (left), output of convolutional by [128×7×7] size (right)

The depth of inception module 5(b) is a concatenation of the depths in all layers of the module. Sum of 384,384,128 and 128 for depth of convolutional layers by 1×1, 3×3,5×5 kernels and depth of max pooling layer is calculated as 1024. The output size of inception module 5(b) is [1024×7×7] and it is shown by Figure 3.33.

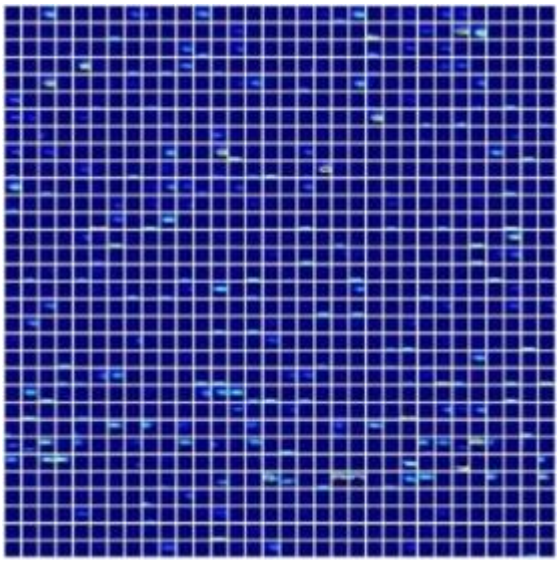

Figure 3.33. Concatenation of inception module 5(b) by [1024×7×7] size

The output of the previous layer by [1024×7×7] is fed to an average pooling layer and it is down sampled by 7×7 kernel size with stride size of 1. The output of the average pooling layer is [1024×7×7] and it is shown by Figure 3.34.

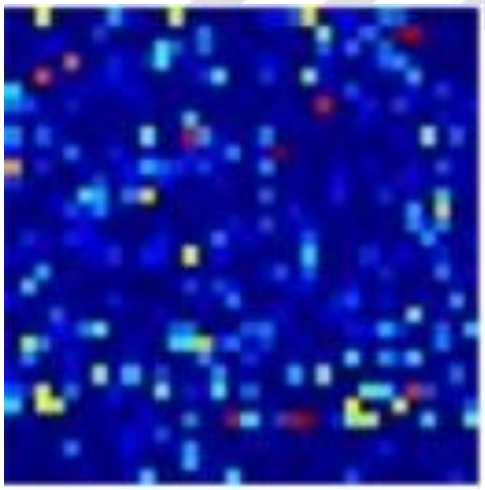

Figure 3.34. Output of avg pooling layer by [1024×1×1] size

The fully connected layer contains 1024 neurons with 1×1 size which is provided by average pooling operation. In fully connected layer instead of local connections, all connections are fully connection the output of the fully connected layer is fed to the softmax classifier. In order to make meaningful output, softmax classifier is applied to achieve the probability of each ground truth labels of the images. Lung dataset includes two classes of benign and

malignant and for each class probability distribution is calculated by softmax classifier. Two classes of lung dataset which are classified by softmax classifier are shown by Figure 3.35.

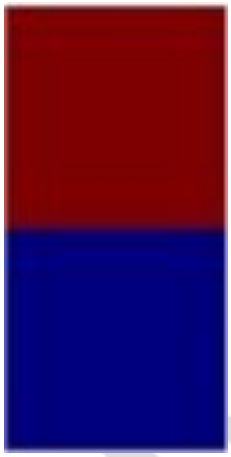

Figure 3.35. Classified lung CT scan images by softmax

Total learned parameters in this architecture is calculated as 5 975 602 parameters. Summary of layers and output size of lung images through training by GoogleNet architecture are given by Table 3.2.

Table 3.2. Summary of GoogleNet architecture in classification of lung CT scan images

Layer type	Number of kernels	Kernel size	Output size
Convolutional	64	7×7	64×112×112
Max pooling		3×3	64×56×56
Convolutional	192	3×3	192×56×56
Max pooling		3×3	192×28×28
Inception 3(a)			256×28×28
Inception 3(b)			480×28×28
Max pooling		3×3	480×14×14
Inception 4(a)			512×14×14
Inception 4(b)			512×14×14
Inception 4(c)			512×14×14
Inception 4(d)			528×14×14
Inception 4(e)			832×14×14
Max pooling		3×3	832×7×7
Inception 5(a)			832×7×7
Inception 5(b)			1024×7×7
Avg pooling		7×7	1024×1×1
Fully connected			1024×1×1
Fully connected with softmax			2×1×1

3.3. Performance Metrics

Generally, for performance evaluation of classification algorithm, real and predicted values of classes are compared by confusion matrix [116]. The confusion matrix is given by Table 3.3. Performance metrics which are used in this thesis are given below.

Table 3.3. Confusion matrix

Confusion Matrix		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

TP: Positive samples which are predicted accurately as a positive label.

FN: Positive samples which are predicted incorrectly as a negative label.

FP: Negative samples which are incorrectly predicted as a positive label.

TN: Negative samples which are correctly predicted as a negative label.

Accuracy: Performance evaluation of algorithm in the classification of class labels of each sample in dataset is calculated by accuracy. Formula of accuracy is given by Eq. 3.1.

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{TN}+\text{FN}} \tag{3.1}$$

Sensitivity or Recall: Indicates what proportion of real positive classes are labeled as positive class by classifier. Formula of sensitivity is given by Eq. 3.2.

$$\text{Sensitivity} = \text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{3.2}$$

Precision: Indicates what proportion of classified classes as positive label have actually positive class labels. Formula of precision is given by Eq. 3.3.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{3.3}$$

Classification performance metrics of used architectures

To evaluate the performance of two architectures, 19 287 samples are selected as test set (9898 samples as benign label and 9389 samples as malignant label). For each one of the architectures (AlexNet and GoogleNet) training samples are trained five times. Then the test set is fed to all five training models of each architecture. Confusion matrix of testing the five times trained for each AlexNet and GoogleNet architectures are given by Table 3.4.

Table 3.4. Confusion matrix of testing the five times trained AlexNet and GoogleNet

Confusion Matrix	Actual	Prediction	
		Malignant	Benign
AlexNet	Malignant	8816	573
	Benign	640	9258
AlexNet	Malignant	8911	478
	Benign	659	9239
AlexNet	Malignant	8766	623
	Benign	414	9484
AlexNet	Malignant	8923	466
	Benign	485	9413
AlexNet	Malignant	8914	475
	Benign	312	9586
GoogleNet	Malignant	9141	248
	Benign	970	8928
GoogleNet	Malignant	8978	411
	Benign	680	9218
GoogleNet	Malignant	8997	392
	Benign	667	9231
GoogleNet	Malignant	9043	346
	Benign	584	9314
GoogleNet	Malignant	9153	236
	Benign	466	9432

Summary of accuracy rate, sensitivity and precision for testing the five times trained AlexNet and GoogleNet architectures by 19 287 samples are given by Table 3.5. It can be demonstrated that by testing 19 287 samples of lung CT scan dataset for five times trained

architectures, accuracy rate of AlexNet by 95.919% is higher than the accuracy rate of other four AlexNet and the accuracy rate of GoogleNet by 96.360% is higher than the other accuracy rate of GoogleNets in test phase as well (Table 3.5). AlexNet with the highest accuracy rate (95.919%) could diagnose 8914 malignant samples (TP) accurately among total malignant of test dataset (9389) and it could diagnose 9586 benign samples (TN) among total 9898 benign samples. AlexNet with the highest accuracy rate value classifies 475 malignant samples as benign samples incorrectly (FN) and it classifies 312 benign samples as malignant samples (FP) incorrectly as well. The highest accuracy rate (96.360%) of GoogleNet is the result of acceptable diagnosis of 9153 malignant samples (TP) between total malignant test dataset (9389). Consequently, GoogleNet classifies 236 malignant samples as benign samples incorrectly (FN). Moreover, GoogleNet could classify benign samples (9432) as actual benign samples (TN) among total 9898 benign samples of the test set. In classification of total 9898 benign samples, GoogleNet with the highest accuracy rate classifies 466 benign samples incorrectly as malignant samples (FP). It is demonstrated that the highest value of TN and TP in both AlexNet and GoogleNet leads to the highest value of accuracy rate in classification of CT scan images (Table 3.4 and Table 3.5).

Table 3.5. Summary of AlexNet and GoogleNet accuracy rates

CNN Architecture	Sensitivity	Precision	Accuracy Rate (%)
AlexNet	0.938	0.932	93.669
AlexNet	0.949	0.931	94.104
AlexNet	0.933	0.954	94.623
AlexNet	0.950	0.948	95.069
AlexNet	0.949	0.966	95.919
GoogleNet	0.973	0.904	93.684
GoogleNet	0.956	0.929	94.343
GoogleNet	0.958	0.930	94.509
GoogleNet	0.963	0.939	95.178
GoogleNet	0.974	0.951	96.360

GoogleNet with the highest accuracy rate (96.360%) achieves the highest value of sensitivity and precision in classification of lung CT scan images as benign and malignant. AlexNet with the highest accuracy rate has the highest precision value while its value of sensitivity is the second highiest value between sensitivity values of the other four Alexnet architectures

(Table 3.5). It is demonstrated that the lowest value of FP and FN and the highest value of TP increased the value of sensitivity and precision. Table 3.6 illustrates training time for each AlexNet and GoogleNet with the highest accuracy rate among their five times trained architectures up to 30 epochs. AlexNet architecture with the highest accuracy rate (95.919%) could train training dataset in 26 minutes and 48 seconds and its training time is less than training time of GoogleNet architecture with the highest accuracy rate (96.360%) with 1 hour and 44 minutes. It is demonstrated that training time of deeper and more complex models is longer than training time of less complex models.

Table 3.6. Training time of AlexNet and GoogleNet for classification of lung CT scan

CNN Architecture	Training time
AlexNet (by 95.919% accuracy rate)	26 min 48 sec
GoogleNet (by 96.360% accuracy rate)	1h 44 min

Loss function is determined for performance evaluation of the training networks. Loss function calculates differentiate between the prediction of labels which are achieved by the algorithm and ground truth labels. In SGD algorithm weights are updated in direction of loss value and the objective of training process is decreasing the loss value to achieve the best trained model. Therefore, to evaluate the performance of the network, loss function is considered in this thesis. Loss diagrams through the learning process for all training dataset of AlexNet and GoogleNet which achieved the highest classification performance are shown by Figure 3.36 and Figure 3.37, respectively.

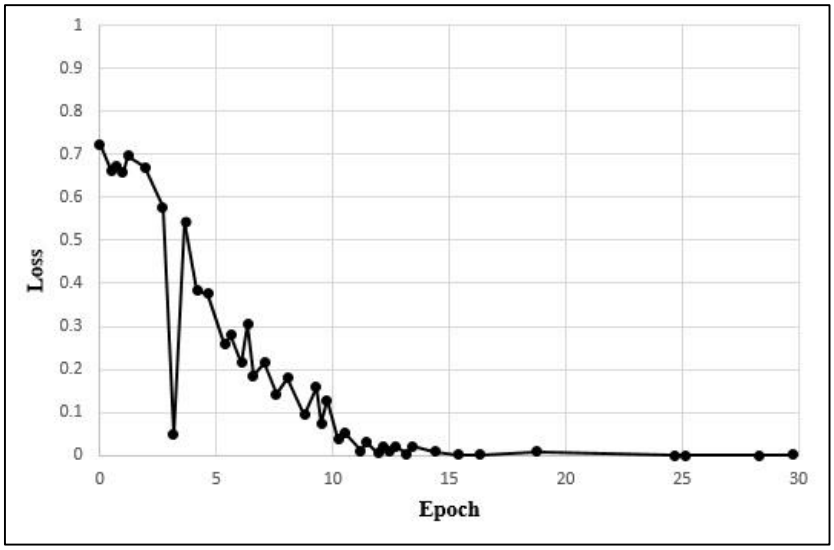

Figure 3.36. The loss diagram of the training dataset for AlexNet

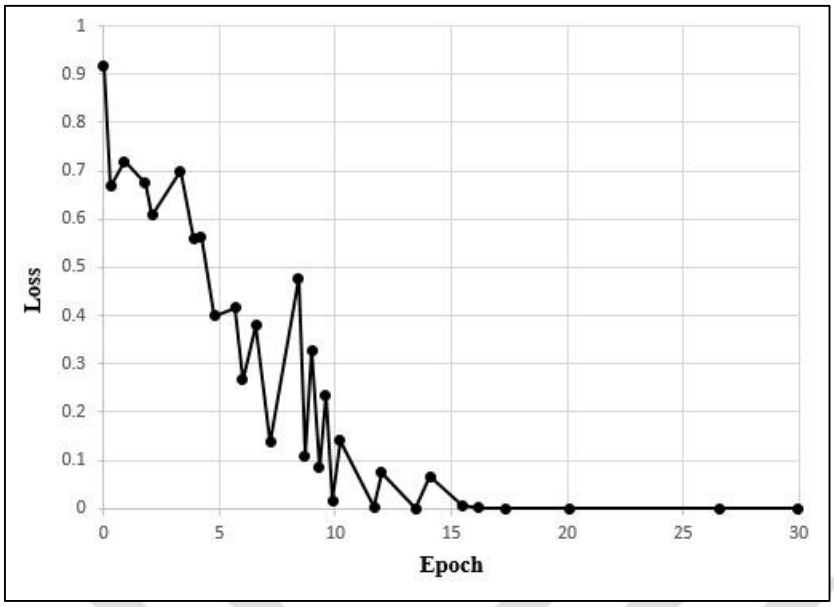

Figure 3.37. The loss diagram of the training dataset for GoogleNet

For classification of lung CT scan images as benign and malignant loss of the tarining phase in AlexNet performed as well as loss of the training phase in GoogleNet architecture. The values of training loss in both algorithms are changed frequently to achieve the least value at the end of training process. Loss of train in AlexNet architecture, after epoch 4 fell down and it is not changed dramatically after epoch 10. At the end of the training of AlexNet loss value dropped down to 0.00075. Loss of train in GoogleNet is not changed dramatically after epoch 15 and it dropped down to 0.000078 at the end of training phase. Both AlexNet and GoogleNet could train samples as well by minimizing loss train through training and loss train in both architecture is close to zero (Figure 3.36 and Figure 3.37). To minimize the loss function, SGD algorithm is used and the learning rate is the most effective parameter of SGD. Through the successful training process, learning rate must be decreased. In this thesis performance of the network by learning rate are shown by figure 3.38 and figure 3.39 for AlexNet and GoogleNet architectures with the highest accuracy rate. In the beginning of training, a learning rate for both network architectures is considered as 0.01. During the training learning rate of AlexNet and GoogleNet drops dramatically in every 10 epochs. Learning rate of both AlexNet and GoogleNet architectures reached 0.00001 after epoch 29. Both architectures could be successful in training of samples by the least learning rate at the end of the training phase (Figure 3.38 and Figure 3.39).

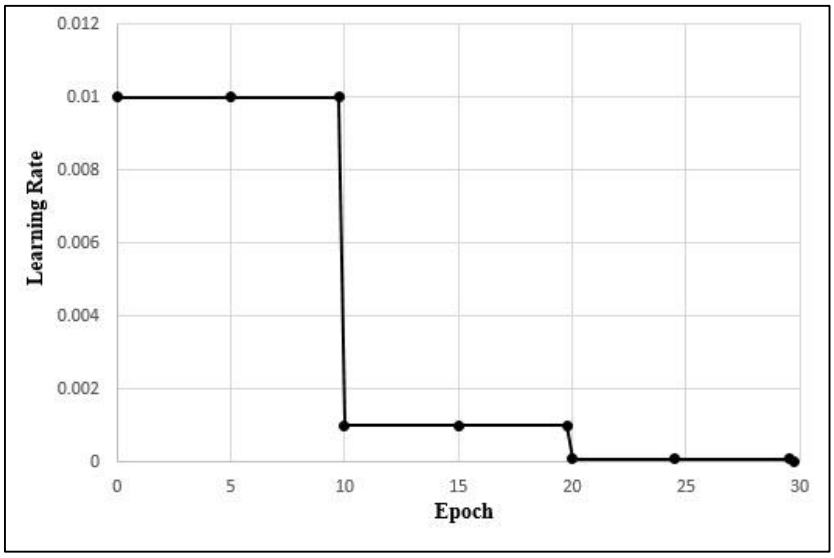

Figure 3.38. Learning rate of training phase by AlexNet

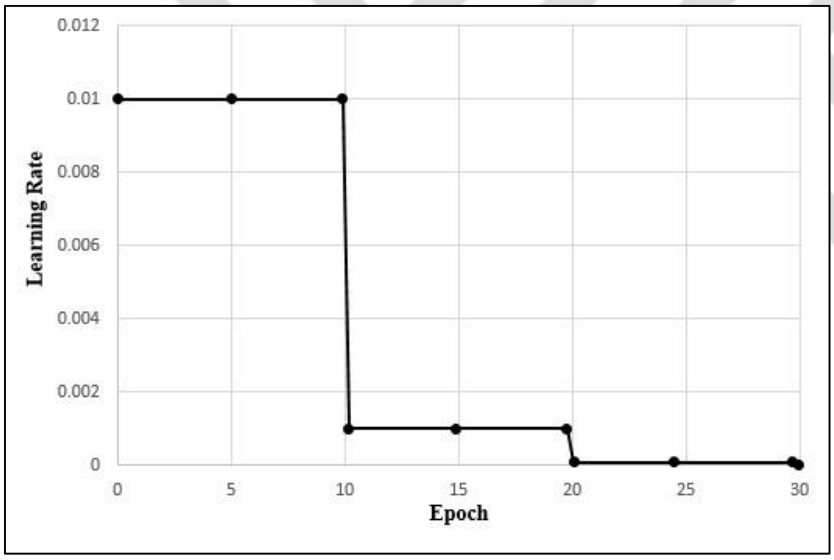

Figure 3.39. Learning rate of training phase by GoogleNet

By comaprison two architectures of CNN it is evident that GoogleNet architecture by 96.360% accuracy rate could diagnose more benign and malignant samples than AlexNet architecture by 95.919% accuracy rate. Results of diferent models of deep learning method which are explained in the literature are shown by Table 3.7. Despite most of the other methods in the literature were used on different lung CT scan datasets, in this thesis the used architectures of CNN have got higher accuracy rate than the methods used in the literature for classification of lung CT scan images. Moreover, accuracy rates of the used AlexNet and GoogleNet architecture in this thesis are higher than the accuracy rate of CNN with U-Net architecture which was used in the other paper [49] on the same dataset. (Table 3.7).

Table 3.7. Results of different methods for classification of lung CT scan images

Dataset	Deep learning method	Compared method	Accuracy (%)	Sensitivity	Precision	Ref.
LIDC	CNN (can)	ANN, LeNet	76			[48]
LIDC &IDRI	DBN	SDAE, CAD	81.19			[41]
LIDC &IDRI	DNN		82.10			[44]
LIDC &IDRI	CNN	DNN,SAE	84.15	0.8396		[50]
ILD	CNN	LeNet, AlexNet, VGG	85.61			[43]
Data science and kaggle	CNN (U-net)		86.60			[49]
LIDC &IDRI	CNN with RF	CNN with SVM	86.84			[47]
ILD	CNN	SVM		0.88	0.93	[46]
Data science and kaggle	CNN (GoogleNet)	AlexNet	96.360 (GooglNet) 95.919 (AlexNet)	0.974 (GoogleNet)	0.951 (GoogleNet)	This thesis

4. CONCLUSION

In this thesis to diagnose lung CT scan images as benign and malignant CNN one of the algorithms of deep learning which is the state- of- art machine learning method is utilized. In this regard, a large dataset of lung CT scan images which includes benign and malignant samples is used. Unlike shallow machine learning methods, preprocessing and feature extraction methods are not separable stages in deep learning methods and deep learning methods take the advantage of automatic feature extraction. In order to train and classify lung CT scan images, AlexNet and GoogleNet which are two architectures of CNN methods are used. For each one of the AlexNet and GoogleNet architectures training dataset is trained five times and test dataset is fed to all trained architectures. In AlexNet architecture five convolutional, five pooling and three fully connected layers are used as a standard structure. In GoogleNet architecture, one fully connected layer and nine inception modules which include different convolutional and pooling layers are used as well. Both architectures of CNN are used in 30 epochs. The results of test phase show that the acceptable AlexNet by 95.919% accuracy rate achieved higher accuracy rate than the other four AlexNet architectures and the acceptable GoogleNet by 96.360% accuracy rate achieved higher accuracy rate among other four trained GoogleNet architectures. Despite the higher value of FP (466) in acceptable GoogleNet than FP value (312) of the acceptable AlexNet, GoogleNet achieved the most acceptable values of TP, TN and FN in classification of samples. GoogleNet classified 9153 malignant samples (TP) correctly among total 9389 malignant samples and it could classify 9432 benign samples (TN) correctly among total 9898 benign samples. Through training phase by minimizing loss train both AlexNet and GoogleNet could calculate weights as well through the network and loss train in both architecture is close to zero. Learning rate in both AlexNet and GoogleNet architectures gradually drop to a minimum value of their first initialized value. Learning rate of both AlexNet and GoogleNet architectures reached 0.00001 after epoch 29. Despite the same performance of AlexNet and GoogleNet architectures in loss train and learning rate through the training phase, the acceptable GoogleNet achieved higher classification accuracy rate (96.360%) than AlexNet architecture (by 95.919% accuracy rate). It can be demonstrated that the GoogleNet is performed better performance than AlexNet in classification of lung CT scan images as benign and malignant samples and it can be helpful for physicians in diagnosis of large amount of CT scan images in terms of time. Moreover, both used architectures of CNN in this thesis achieved higher accuracy rate than the other methods which were used in the

literature. It is noticeable that using larger or different CT scan images and the other CNN architectures cause to achieve different accuracy rate.

In conclusion, it has been proved that with the classification of the lung CT scan images using deep learning methods, more information concerning early diagnosis of lung cancer may be obtained with a noninvasive method.

REFERENCES

1. Koyama, S., Sato, E., Nomura, H., Kubo, K., Miura, M., Yamashita, T., Nagai, S. and Izumi, T. (1998). Bradykinin Stimulates Type II Alveolar Cells to Release Neutrophil and Monocyte Chemotactic Activity and Inflammatory Cytokines. *American Journal of Pathology,* 153(6), 1885-1893.

2. Parkin, D. (2001). Global cancer statistics in the year 2000. *The Lancet Oncology,* 2(10), 533-543.

3. Mitsuuchi, Y. and Testa, J. R. (2002). Cytogenetics and molecular genetics of lung cancer. *Journal of Medical Genetics,* 115(3), 183-188.

4. Firmino, M., Angelo, G., Morais, H., Dantas, M. R. and Valentim, R. (2016). Computer-aided detection (CADe) and diagnosis (CADx) system for lung cancer with likelihood of malignancy. *BioMedical Engineering OnLine,* 15, 2.

5. Nguyen, H. T., Worring, M. and van den Boomgaard, R. (2003). Watersnakes: energy-driven watershed segmentation. *The Institute of Electrical and Electronics Engineers Transactions on Pattern Analysis and Machine Intelligence,* 25(3), 330 - 342.

6. Parkina, D. M., Bray, F.I. and Devesa, S.S. (2001). Cancer burden in the year 2000. The global picture. *European Journal of Cancer,* 37, 4-66.

7. Takkouche, B. and Gestal-Otero, J. J. (1996). The epidemiology of lung cancer: Review of risk factors and Spanish data. *European Journal of Epidemiology,* 12(4), 341-349.

8. Maheswaran S. and Haber, D. A. (2010). Circulating tumor cells: a window into cancer biology and metastasis. *Current Opinion in Genetics & Development,* 20, 96-99.

9. Wingo, P. A., Ries, L. A. G., Giovino, G. A., Miller, D. S., Rosenberg, H. M., Shopland, D. R., Thun, M. J. and Edwards, B. K. (1999). Annual Report to the Nation on the Status of Cancer, 1973–1996, With a Special Section on Lung Cancer and Tobacco Smoking. *Journal of the National Cancer Institute,* 91(8), 675-690.

10. Soria, J. C., Kim, E. S., Fayette, J., Lantuejoul, S., Deutsch, E. and Hong, W. K. (2003). Chemoprevention of lung cancer. *The Lancet Oncology,* 4, 659-669.

11. Rekhtman, N. (2010). Neuroendocrine Tumors of the Lung: An Update. *Archives of Pathology & Laboratory Medicine*, 134, 1628-1638.

12. Devesa S. S., Bray F., Vizcaino A. P. and Parkin D. M. (2005). International lung cancer trends by histologic type: male:female differences diminishing and adenocarcinoma rates rising. *International Journal of Cancer*, 117(2), 294-299.

13. Lin D. T. and Yan, C. R. (2002). *Lung Nodules Identification Rules Extraction With Neural Fuzzy Network.* Paper presented at the Proceedings ofthe 9th International Conference on Neural Information Processing (ICONIP'02), Singapore, Singapore.

14. Moran, T. C., Kaye, A. D., Rao A. and Bueno, F. R. (2016). The roles of Xrays and other types of electromagnetic radiation in evaluating paintings for forgery and restoration. *Journal of Forensic Radiology and Imaging,* 5, 38-46.

15. Specht, L. and Berthelsen, A. K. (in press). PET/CT in radiation therapy planning. *Seminars in Nuclear Medicine,* 48(1), 67–75.

16. Fass, L. (2008). Imaging and cancer: A review. *Molecular Oncology*, 2, 115–152.

17. Andreea, G. I., Pegza, R., Lascu, L., Bondari, S., Stoica, Z. and Bondari, A. (2011). The Role of Imaging Techniques in Diagnosis of Breast Cancer. *Current Health Sciences Journal,* 37(2), 55-61.

18. Hubersa, A. J., van der Drift, M. A., Prinsen, C. F.M., Witte, B. I., Wang, Y., Shivapurkar, N., Stastny, V., Bolijn, A. S., Hol, B. E. A., Feng, Z., Dekhuijzen, P. N. R., Gazdar, A. F. and Thunnissen, E. (2014). Methylation analysis in spontaneous sputum for lung cancer diagnosis. *Lung Cancer,* 84, 127-133.

19. Wright, J. D., Cham, S., Chen, L., Burke, W. M., Hou, J. Y., Tergas, A. I., Desai, V., Hu, J. C., Ananth, C. V., Neugut, A. I. and Hershman, D. L. (2017). Utilization of sentinel lymph node biopsy for uterine cancer. *American Journal of Obstetrics and Gynecology*, 216(594), 1-13.

20. Revel, M. P., Bissery, A., Bienvenu, M., Aycard, L., Lefort, C. and Frija, G. (2004). Are two-dimensional CT measurements of small noncalcified pulmonary nodules reliable?. *Radiology*, 231(2), 453-458.

21. Sharma, D. and Jindal, G. (2011). *Identifying Lung Cancer Using Image Processing Techniques.* Paper presented at the International Conference on Computational Techniques and Artificial Intelligence (ICCTAI'2011), Landran, India.

22. Pedersen, J. H., Ashraf, H., Dirksen, A., Bach, K., Hansen, H., Toennesen, P., Thorsen, H., Brodersen, J., Skov, B. G., Døssing, M., Mortensen, J., Richter, K., Clementsen, P. and Seersholm, N. (2009). The Danish Randomized Lung Cancer CT Screening Trial—Overall Design and Results of the Prevalence Round. *Journal of Thoracic Oncology,* 4(5), 608-614.

23. Sagara, Y., Hara, A. K., Pavlicek, W., Silva, A. C., Paden, R. G. and Wu, Q. (2010). Abdominal CT: Comparison of Low-Dose CT With Adaptive Statistical Iterative Reconstruction and Routine-Dose CT With Filtered Back Projection in 53 Patients. *American Journal of Roentgenology,* 195(3), 713-719.

24. Hammen, I. (2015). Tuberculosis mimicking lung cancer. *Respiratory Medicine Case Reports,* 16, 45–47.

25. Suzuki, K., Yan, P., Wang, F. and Shen, D. (2012). Machine Learning in Medical Imaging. *International Journal of Biomedical Imaging,* 2012(2), 1.

26. Singh, S., Maxwell, J., Baker, J. A., Nicholas, J. L. and Lo, J. Y. (2011). Computer-aided Classifi cation of Breast Masses: Performance and Interobserver Variability of Expert Radiologists Versus Residents. *Radiology,* 258(1), 73-80.

27. Cheng, J. Z., Ni, D., Chou, Y. H., Qin, J., Tiu, C. M., Chang, Y. C., Huang, C. S., Shen, D. and Chen, C. M. (2016). Computer-Aided Diagnosis with Deep Learning Architecture: Applications to Breast Lesions in US Images and Pulmonary Nodules in CT Scans. *Scientific Reports*, 6, 24454.

28. Hua, K. L., Hsu, C. H., Hidayati, S. C., Cheng, W. H. and Chen, Y. J. (2015). Computer-aided classification of lung nodules on computed tomography images via deep learning technique. *OncoTargets and Therapy*, 5(8), 2015–2022.

29. Niki, N., Kawata, Y. and Kubo, M. (2001). A CAD system for lung cancer based on CT image. *International Congress Series*, 1230, 631–638.

30. Matsuki, Y., Nakamura, K., Watanabe, H., Aoki, T., Nakata, H., Katsuragawa, S. and Doi, K. (2002). Usefulness of an Artificial Neural Network for Differentiating Benign from Malignant Pulmonary Nodules on High-Resolution CT: Evaluation with Receiver Operating Characteristic Analysis. *American Journal of Roentgenology*, 178(3), 657-663.

31. Penedo, M. G., Carreira, Mosquera, M. J. A. and Cabello, D. (1998). Computer-Aided Diagnosis: A Neural-Network-Based Approach to Lung Nodule Detection. *The Institute of Electrical and Electronics Engineers Transactions on Medical Imaging*, 17(6), 872-880.

32. Teramoto, A. and Fujita, H. (2013). Fast lung nodule detection in chest CT images using cylindrical nodule-enhancement filter. *International Journal of Computer Assisted Radiology and Surgery*, 8, 193–205.

33. Kakar, M. and Olsen, D. R. (2009). Automatic segmentation and recognition of lungs and lesion from CT scans of thorax. *Computerized Medical Imaging and Graphics*, 33(1), 72–82.

34. Chen, H., Zhang, J., Xu, Y., Chen, B. and Zhang, K. (2012). Performance comparison of artificial neural network and logistic regression model for differentiating lung nodules on CT scans. *Expert Systems with Applications*, 39(13), 11503–11509.

35. Wang, Q., Kang, W., Wu, C. and Wang, B. (2013). Computer-aided detection of lung nodules by SVM based on 3D matrix patterns. *Clinical Imaging*, 37(1), 62–69.

36. Kulkarni, A. and Panditrao, A. (2014). *Classification of Lung Cancer Stages on CT Scan Images Using Image Processing*. Paper presented at the 2014 Institute of Electrical and Electronics Engineers International Conference on Advanced Connnunication Control and Computing Teclmologies (lCACCCT), Ramanathapuram, India.

37. Arimura, H., Katsuragawa, S., Suzuki, K., Li, F., Shiraishi, J., Sone, S. and Doi, K. (2004). Computerized Scheme for Automated Detection of Lung Nodules in Low-Dose Computed Tomography Images for Lung Cancer Screening. *Academic Radiology*, 11(6), 617–629.

38. Suzuki, K., Armato, S. G., Li, F., Sone, S. and Doi, K. (2003). Massive training artificial neural network (MTANN) for reduction of false positives in computerized

detection of lung nodules in low-dose computed tomography. *Medical Physics*, 30(7), 1602–1617.

39. Jacobs, C., van Rikxoort, E. M., Scholten, E. T., de Jong, P. A., Prokop, M., Schaefer-Prokop, C. and van Ginneken, B. (2015). Solid, part-solid, or non-solid?: classification of pulmonary nodules in low-dose chest computed tomography by a computer-aided diagnosis system. *Investigative Radiology*, 50(3), 168-173.

40. Way, T. W., Sahiner, B., Chan, H. P., Hadjiiski, L., Cascade, P. N., Chughtai, A., Bogot, N. and Kazerooni, E. (2009). Computer-aided diagnosis of pulmonary nodules on CT scans: Improvement of classification performance with nodule surface features. *Medical Physics*, 36(7), 3086–3098.

41. Sun, W., Zheng, B. and Qian, W. (2016). *Computer aided lung cancer diagnosis with deep learning algorithms.* Paper presented at the Proceedings of the International Society for Optics and Photonics Conference, California, United States.

42. Ginneken, B. V., Setio, A. A. A., Jacobs, C. and Ciompi, F. (2015). *Off-The-Shelf Convolutional Neural Network Features For Pulmonary Nodule Detection In Computed Tomography Scans.* Paper presented at 2015 Institute of Electrical and Electronics Engineers 12th International Symposium on Biomedical Imaging (ISBI), New York, United States.

43. Anthimopoulos, M., Christodoulidis, S., Ebner, L., Christe, A. and Mougiakakou, S. (2016). Lung Pattern Classification for Interstitial Lung Diseases Using a Deep Convolutional Neural Network. *The Institute of Electrical and Electronics Engineers Transactions on Medical Imaging*, 35(5), 1207-1216.

44. Gruetzemacher, R. and Gupta, A. (2016). *Using Deep Learning for Pulmonary Nodule Detection & Diagnosis.* Paper presented at the Twenty-second Americas Conference on Information Systems, San Diego, United States.

45. Ciompi, F., Chung, K., van Riel, S. J., Setio, A. A. A., Gerke, P. K., Jacobs, C., Scholten, E. T., Prokop, C. S., Wille, M. M. W., Marchianò, A., Pastorino, U., Prokop, M. and van Ginneken, B. (2017). Towards automatic pulmonary nodule management in lung cancer screening with deep learning. *Scientific Reports*, 7(46479), 1-10.

46. Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D. D. and Chen, M. (2014). *Medical Image Classification with Convolutional Neural Network.* Paper presented at the 2014 13th International Conference on Control, Automation, Robotics & Vision, Marina Bay Sands, Singapore.

47. Shen, W., Zhou, M., Yang, F., Yang, C. and Tian, J. (2015). Multi-scale Convolutional Neural Networks for Lung Nodule Classification. *Information Processing in Medical Imaging*, 24, 588-599.

48. Rao, P., Pereira, N. A. and Srinivasan, R. (2016). *Convolutional Neural Networks for Lung Cancer Screening in Computed Tomography (CT) Scans.* Paper presented at the 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, India.

49. Alakwaa, W., Nassef, M. and Badr, A. (2017). Lung Cancer Detection and Classification with 3D Convolutional Neural Network (3D-CNN). *International Journal of Advanced Computer Science and Applications (IJACSA)*, 8(8), 409-417.

50. Song, Q. Z., Zhao, L., Luo, X. K. and Dou, X. C. (2017). Using Deep Learning for Classification of Lung Nodules on Computed Tomography Images. *Journal of Healthcare Engineering*, 2017, 1-7.

51. Bondfale N. and Banait, S. (2017). Lung Pattern Classification for Interstitial Lung Diseases Using a Deep Convolutional Neural Network. *International Journal of Innovative Research in Computer and Communication Engineering*, 5(5), 9851-9856.

52. Fyfe, C. (2005). *Do Smart Adaptive Systems Exist? Artificial Neural Networks.* Berlin*:* Springer,57–79.

53. Basheer, I.A. and Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43, 3–31.

54. Pratap, K., and Shelja. (2013). Artificial Neural Network (Ann) Inspired From Biological Nervous System. *International Journal of Application or Innovation in Engineering & Management*, 2(1), 227-231.

55. Jain, A. K., Mao, J. and Mohiuddin, K. (1996). Artificial neural networks: A tutorial. *The Institute of Electrical and Electronics Engineers Computer*, 29, 31-44.

56. Mcculloch, W. S., and Pitts, W. (1990). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mothemnticnl Biology*, 52(1/2), 99-115.

57. Rochester, N., Holland, J. H., Haibt, L.H., and Duda, W.L. (1956). Tests on A Cell Assembly Theory of The Action of The Brain, Using A Large Digital Computer. *Institute of the Radio Engineers Transactions on Information Theory*, 2(3), 80 - 93.

58. Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model For Information Storage And Organization in The Brain. *Psychological Review*, 65(6), 65-386.

59. Klopf, A. H. (1972). Brain Function and Adaptive Systems: A Heterostatic Theory, Air Force Cambridge Research Laoratories, *Special Peports*, 133.

60. Werbos, P. (1974). *Beyond regression : new tools for prediction and analysis in the behavioral sciences*, Doctorate Thesis, Harvard University Mathematical Engineering and Applied Physics, United States.

61. Graupe, D. (2013). *Principles of artificial neural networks.* Singapur: World Scientific, 17-36.

62. Sibi, P., Jones, S. A. and Siddarth, P. (2013). Analysis of Different Activation Functions Using Back Propagation Neural Networks. *Journal of Theoretical and Applied Information Technology*, 47(3), 1264-1268.

63. He, K., Zhang, X., Ren, S. and Sun, J. (2016). *Deep Residual Learning for Image Recognition*. Paper presented at the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA.

64. Glorot, X., and Bengio, Y. (2010). *Understanding the difficulty of training deep feedforward neural networks*. Paper presented at theProceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Qu´ebec, Canada.

65. Karlik, B., and Olgac, A. V. (2011). Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence And Expert Systems*, 1(4), 111-122.

66. Debes, K., Koenig, A., and Gross, H. M. (2005). Transfer functions in artificial neural networks-a simulation-based tutorial. *Supplementary Material for Urn*, 1, 1-11.

67. Maas, A. L., Hannun, A. Y. and Ng, A. Y. (2013). *Rectifier nonlinearities improve neural network acoustic models.* Paper presented at the Proceedings of the 30 th International Conference on Machine Learning, Atlanta, Georgia, United States.

68. Svozil, D., Eka, V. K. and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39, 43-62.

69. Hagiwara, M. (1992). *Theoretical derivation of momentum term in back-propagation.* Paper presented at the International Joint Conference on Neural Networks (IJCNN), Baltimore, MD, United States.

70. Satapathy, S.C., Udgata, S.K. and Biswal, B.N. (2014). *Advances in intelligent systems and computing* (vol. 247). Switzerland: Springer International Publishing.

71. Internet: Nielsen, M. A. (2015). *Neural networks and deep learning*. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fneuralnetworksanddeeplearning.com%2F&date=2017-12-28, Last Access Date: 04.11.2017.

72. Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. United States: MIT Press.

73. Ruder, S. (2016). An overview of gradient descent optimization algorithms. *Clinical Orthopaedics and Related Research*, 1609, 04747.

74. LeCun, Y., Bottou, L., OrrKlaus, G. B., and Müller, R. (1998). Efficient BackProp. *Lecture Notes in Computer Science (LNCS)*, 1524, 1-44.

75. Bryson A. E. and Ho Y. C. (1969). *Applied optimal control: optimization, estimation, and control.* Waltham, MA: Blaisdell.

76. Parker, D. B. (1986). *A comparison of algorithms for neuron-like cells*. Paper presented at the American Institute of Physics (AIP) Conference Proceedings, Utah, United States.

77. Nielsen, R. H. (1989). *Theory of the backpropagation neural network*. in International Joint Conference on Neural Networks (IJCNN), Washington, DC, United States.

78. LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *Nature*, 521, 436-444.

79. Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science,* 313, 504-507.

80. Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning,* 2(1), 1-127.

81. Dong, C., Loy, C. C., He, K. and Tang, X. (2014, September). Learning a deep convolutional network for image super-resolution. Paper presented at the European Conference on Computer Vision (pp. 184-199), Hong Kong, China.

82. Hinton, G., Osindero, S. and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation,* 18, 1527–1554.

83. Ranzato, M. A., Huang, F. J., Boureau, Y. L. and LeCun, Y. (2007). *Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition*. Paper presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07), Minneapolis, MN, United States.

84. Lee, H., Largman, Y., Pham, P. and Ng, A. Y. (2009). *Unsupervised feature learning for audio classification using convolutional deep belief networks*. Paper presented at the Neural Information Processing Systems (NIPS'09) Proceedings of the 22nd International Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada.

85. Hubel, D. H. and Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology.,* 195, 215-243.

86. Turaga, S. C. (2010). Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation,* 22, 511–538.

87. Ning, F., Delhomme, D., LeCun, Y., Piano, F., Bottou, L. and Barbano, P. E. (2005). Toward automatic phenotyping of developing embryos from videos. *The Institute of Electrical and Electronics Engineers Transactions on Image Processing,* 14(9), 1360-1371.

88. Cireşan, D., Meier, U., Masci, J., and Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks,* 32, 333–338.

89. Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the The Institute of Electrical and Electronics Engineers,* 86(11), 2278 - 2324.

90. Cireşan, D. C., Meier, U., Masci, J., Gambardella, L. M. and Schmidhuber, J. (2011). *Flexible, high performance convolutional neural networks for image classification*. in IJCAI'11 Proceedings of the Twenty-Second international joint conference on Artificial Intelligence, Barcelona, Catalonia, Spain.

91. Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks*. in Neural Information Processing Systems (NIPS'12) Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, Nevada.

92. Simonyan, K. and Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Paper presented at the Published as a conference International Conference on Learning Representations, Oxford, United Kingdom.

93. Internet: Image Net. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fimage-net.org%2F&date=2017-12-27, Last Access Date: 04.11.2017.

94. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2015). *Going Deeper with Convolutions*. Paper presented at the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, United States.

95. László, E., Szolgay, P., and Nagy, Z. (2012). *Analysis of a GPU based CNN implementation*. Paper presented at the 2012 13th International Workshop on Cellular Nanoscale Networks and Their Applications, Turin, Italy.

96. Cireşan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2011). *Handwritten Digit Recognition with a Committee of Deep Neural Nets on GPUs*. Istituto Dalle Molle di Studi Sull'Intelligenza Artificiale / Università della Svizzera İtaliana, Manno, Switzerland.

97. Internet: NVIDIA. Deep Learning AI. URL: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.nvidia.com%2Fen-us%2Fdeep-learning-ai%2F&date=2017-12-27, Last Access Date: 14.10. 2017.

98. LeCun, Y., Kavukcuoglu, K. and Farabet, C. (2010, May). Convolutional networks and applications in vision. Paper presented at the *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on* (pp. 253-256). Paris, France.

99. Ketkar, N. (2017). *Deep learning with python: A hands-on introduction.* Bangalore: A Press.

100. Gollapudi, S. (2016). *Practical machine learning*. United Kingdom: Packt Publishing Ltd.

101. Aghdam, H. H. and Heravi, E. J. (2017). *Guide to Convolutional Neural Networks*. Switzerland: Springer.

102. Abdel-Hamid, O., Deng, L. and Yu, D. (2013). *Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition*. Paper presented at Interspeech 2013, Lyon, France.

103. Stutz, D. (2014). *Understanding convolutional neural networks*. Germany: Fakultät für Mathematik, Informatik und Naturwissenschaften.

104. Scherer, D., Müller, A. and Behnke, S. (2010). *Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition*. Paper presented at the 20th International Conference on Artificial Neural Networks (ICANN), Thessaloniki, Greece.

105. Kramer, R. H. and Davenport, C. M. (2015). Lateral Inhibition in the Vertebrate Retina: The Case of the Missing Neurotransmitter. *Public Library of Science Biology*, 13(2), 1002322.

106. Jarrett, K., Kavukcuoglu, K., Ranzato, M. A. and LeCun, Y. (2009). *What is the best multi-stage architecture for object recognition?*. Paper presented at the 2009 Institute of Electrical and Electronics Engineers 12th International Conference on Computer Vision, Kyoto, Japan.

107. Ioffe, S. and Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* Paper presented at the Proceedings of the 32nd International Conference on Machine Learning, California, United States.

108. Bouchard, G. (2007). *Efficient Bounds for the Softmax Function and Applications to Approximate Inference in Hybrid models*. Paper presented at the Presentation at The Workshop For Approximate Bayesian Inference in Continuous/Hybrid Systems at Neural Information Processing Systems (NIPS), Meylan, France.

109. Internet: Lin, M., Chen, Q. and Yan, S. (2014). *Network In Network*. arXiv:1312.4400v3 [cs.NE]. URL: http://www.webcitation.org/query?url=https%3A%2F%2Farxiv.org%2Fpdf%2F1312.4400.pdf&date=2017-12-28, Last Access Date: 14.10. 2017.

110. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016). *Rethinking the Inception Architecture for Computer Vision*. Paper presented at the The Institute of Electrical and Electronics Engineers Conference on Computer Vision and Pattern Recognition (CVPR), Nevada, United States.

111. Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A. A. (2017). *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning.* Paper presented at the Proceedings of the Thirty-First Conference on Artificial Intelligence (AAAI-17), Mountain View, California, United States.

112. Internet: National Institutes of Health. (2011). Cancer costs projected to reach at least $158 billion in 2020: New NIH study projects survivorship and costs of cancer care based on changes in the US population and cancer trends. National Institute of Health (NIH). URL: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.nih.gov%2Fnews-events%2Fnews-releases%2Fcancer-costs-projected-reach-least-158-billion-2020&date=2017-12-27, Last Access Date: 9. 11. 2017.

113. Internet: Data Science Bowl. Data Science Bowl. URL: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.kaggle.com%2Fc%2Fdata-science-bowl-2017&date=2017-12-28, Last Access Date: 6. 9. 2017.

114. Lan, Z., Yu, S. I., Lin, M., Raj, B. and Hauptmann, A. G. (2015). *Local Handcrafted Features Are Convolutional Neural Networks*. Paper presented at International Conference on Learning Representations, San Juan, Puerto Rico.

115. Cho, J., Lee, K., Shin, E., Choy, G. and Do, S. (2015). *Medical Image Deep Learning With Hospital Pacs Dataset,* Paper presented at the International Conference on Learning Representations, Boston, Massachusetts, United States.

116. Polat H., Danaei Mehr H. and Cetin A. (2017). Diagnosis of chronic kidney disease based on support vector machine by feature selection methods. *Journal of Medical Systems,* 41(4), 55.

GAZİ GELECEKTİR...