EARDRUM IMAGE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK

A PROJECT REPORT

Submitted by

DEBORAH DEVAKIRUBAI .M	**(211418107019)**
JASMINE .C	**(211418107039)**
RESHMA. V	**(211418107076)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND INSTRUMENTATION ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MAY 2022

PANIMALAR ENGINEERING COLLEGE

(An Autonmous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"EARDRUM IMAGE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK"** is the bonafide work of **"DEBORAH DEVA KIRUBAI .M (211418107019), JASMINE .C (211418107039) and RESHMA. V (211418107076)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. C. ESAKKIAPPAN, M.E., Ph.D.,	**Mr. V. VASUDHEVAN, M.E.,**
Head of the Department,	**SUPERVISOR**,
Professor,	Assistant Professor,
Department of Electronics and	Department of Electronics and
Instrumentation Engineering,	Instrumentation Engineering,
Panimalar Engineering College,	Panimalar Engineering College,
Chennai - 123	Chennai - 123

Certified that the above-mentioned students were examined in Anna University project viva-voice held on_________________________.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our beloved **Secretary and Correspondent Dr.P.CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation which inspired us a lot in completing this project. We express our sincere thanks to our **Directors Dr.C.SAKTHI KUMAR M.E.,Ph.d, Dr.SARANYA SAKTHI KUMAR and Mrs.C.VIJAYA RAJESWARI** for providing us with necessary facilities forcompletion of this project.

We also express gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.,** who has been source of constant encouragement and support.

We would also like to express our gratitude to **Dr.C.ESAKKIAPPAN., M.E., Ph.D.,** Head of the Department, Electronics and Instrumentation Engineering, for his valuable guidance, ideas and encouragement for successful completion of this project.

We would like to thank our project coordinator **Dr.R.MANIKANDAN., M.E., Ph.D.,** and our internal guide **Mr.V.VASUDHEVAN, M.E., Assistant Professor.,** for their valuable guidance.

We also extend our thanks to **Ms.VIJAYASARASWATHI, M.E., Assistant Professor** (Mentor) and **all other staff members and instructors** of our department for their support towards our successful completion of project completion.

We take this opportunity to thank our beloved parents and friends for their constant support and encouragement.

ABSTRACT

Acute infections of the middle ear are the most commonly treated childhood diseases. Because complications affect children's language learning and cognitive processes, it is essential to diagnose these diseases in a timely and accurate manner. Middle ear diseases are diagnosed using patient's history and otoscopic findings in the tympanic membrane (TM). Guidelines on otitis media highlight the usefulness of TM findings in diagnosing various types of otitis media. Clinical diagnosis of otitis media is mainly subjective. In addition, it shows limited reproducibility due to the high reliance on the clinician experience and cooperation of the patients. Artificial intelligence is often employed to perform high-level image analysis such as image classification, segmentation and matching. In particular, Convolutional Neural Networks (CNNs) have demonstrated good performance in automatic classification of medical images (e.g., skin lesions, eye lesions, radiologic and MRI images).

In this project, we design an image classification system for normal, middle ear effusion, and tympanostomy tube conditions, operating on eardrum images and comparing the accuracy and F1 Score using general CNN Algorithm, ResNet -18 and ALEXNET Algorithms. A database of 454 labeled eardrum images (179 normal, 179 effusions, and 96 tube cases) are being used to train and test the system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

ABBREVATION	**EXPANSION**
TM	Tympanic Membrane
DL	Deep Learning
AI	Artificial Intelligence
CNN	Convolutional Neural Network
ResNet	Residual Neural Network
MATLAB	Matrix Laboratory

CHAPTER 1

INTRODUCTION

1.1 GENERAL INTRODUCTION

Middle ear diseases are diagnosed using patient’s history and otoscopic findings in the tympanic membrane (TM). Guidelines on otitis media highlight the usefulness of TM findings in diagnosing various types of otitis media. The development of video endoscopes has enabled more accurate detection of diagnostic lesions. However, clinical diagnosis of otitis media is mainly subjective. In addition, it shows limited reproducibility due to the high reliance on the clinician experience and cooperation of the patients. Artificial intelligence is often employed to perform high-level image analysis such as image classification, segmentation and matching. The success of such an analysis highly depends on feature extraction, for which deep learning is widely used. In particular, convolutional neural networks (CNNs) have demonstrated good performance in automatic classification of medical images (e.g., skin lesions, eye lesions, radiologic and MRI images). Several studies have applied deep learning for making diagnoses based on otoscopic images. Accuracies from 73.11% to 91.41% have been reported when applying deep learning to distinguish acute otitis media (AOM) and otitis media with effusion (OME).

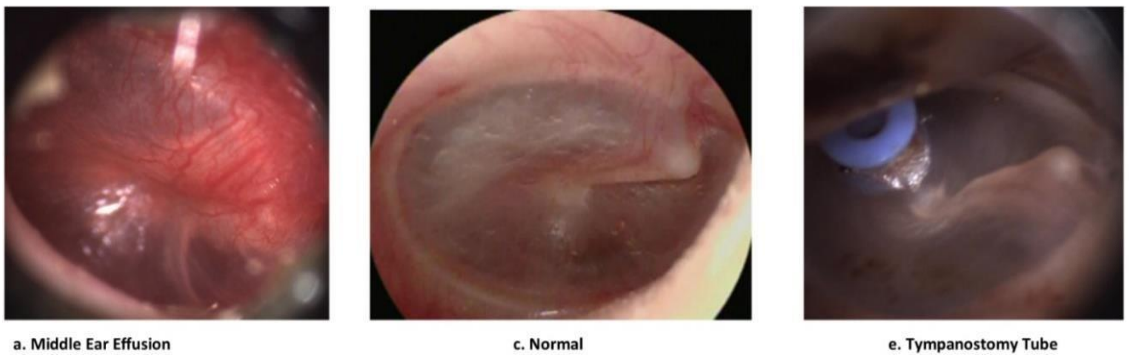


Fig 1.1 (a)Middle Ear Effusion, (b)Normal, (c)Tympanostomy Tube

1.2 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is a wide-ranging branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence.

1.2.1 EXAMPLES OF ARTIFICIAL INTELLIGENCE

- Alexa and other smart assistants
- Self-driving cars
- Robot-advisors
- Conversational bots
- Email spam filters
- Netflix’s recommendations

1.2.2 FOUR DIFFERENT APPROACHES THAT HAVE HISTORICALLY DEFINED THE FIELD OF AI

- Thinking humanly
- Thinking rationally
- Acting humanly
- Acting rationally

1.3 IMAGE PROCESSING

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

1.3.1 THREE STEPS IN IMAGE PROCESSING

- Importing the image via image acquisition tools;
- Analyzing and manipulating the image;

- Output in which result can be altered image or report that is based on image analysis.

There are two types of methods used for image processing namely, analogue and digital image processing. Analogue image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in manipulation of the digital images by using computers. The three general phases that all types of data have to undergo while using digital technique are pre-processing, enhancement, and display, information extraction. In this lecture we will talk about a few fundamental definitions such as image, digital image, and digital image processing. Different sources of digital images will be discussed and examples for each source will be provided. The continuum from image processing to computer vision will be covered in this lecture. Finally, we will talk about image acquisition and different types of image sensors.

1.4 DEEP LEARNING

Deep learning is a branch of machine learning that teaches computers to do what comes naturally to humans: learn from experience. Machine learning algorithms use computational methods to "learn" information directly from data without relying on a predetermined equation as a model. Deep learning is especially suited for image recognition, which is important for solving problems such as facial recognition, motion detection, and many advanced driver assistance technologies such as autonomous driving, lane detection, pedestrian detection, and autonomous parking. Deep learning uses neural networks to learn useful representations of features directly from data. Neural networks combine multiple nonlinear processing layers, using simple elements operating in parallel and inspired by biological nervous systems. Deep learning models can achieve state-of-the-art accuracy in object classification, sometimes exceeding human-level performance.

You train models using a large set of labelled data and neural network architectures that contain many layers, usually including some convolutional layers. Training these models is computationally intensive and you can usually accelerate training by using a high-performance GPU. This diagram shows how convolutional neural networks combine layers that automatically learn features from many images to classify new images.

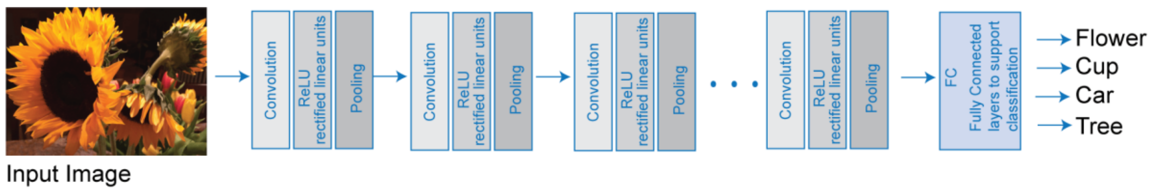

Input Image
Convolution
ReLU rectified linear units
Pooling
Convolution
ReLU rectified linear units
Pooling
Convolution
ReLU rectified linear units
Pooling
Convolution
ReLU rectified linear units
Pooling
FC Fully Connected layers to support classification
Flower
Cup
Car
Tree

Fig 1.2 Deep Learning Diagram

Many deep learning applications use image files, and sometimes millions of image files. To access many image files for deep learning efficiently, MATLAB provides the image Data store function. Use this function to:

- Automatically read batches of images for faster processing in machine learning and computer vision applications.
- Import data from image collections that are too large to fit in memory.
- Label your image data automatically based on folder names.

1.5 TYPES OF LAYERS IN DEEP LEARNING

- Input Layer
- Convolutional Layer
- Max Pooling Layer
- Relu Layer
- Batch Normalization Layer
- Soft Max Layer
- Fully Connected Layer

1.5.1 INPUT LAYER

The input layer of a neural network is composed of artificial input neurons, and brings the initial data into the system for further processing by subsequent layers of artificial neurons. The input layer is the very beginning of the workflow for the artificial neural network.

1.5.2 CONVOLUTIONAL LAYER

Convolution is an orderly procedure where two sources of information are intertwined; it’s an operation that changes a function into something else. Convolutions have been used for a long time typically in image processing to blur and sharpen images, but also to perform other operations. (e.g., enhance edges and emboss) CNNs enforce a local connectivity pattern between neurons adjacent layers. CNNs make use of filters (also known as kernels), to detect what features, such as edges, are present throughout an image. There are four main operations in a CNN:

- Convolution
- Non-Linearity (ReLU)
- Pooling or Sub Sampling
- Classification (Fully Connected Layer)

The first layer of a Convolutional Neural Network is alwaysa Convolutional Layer**.** Convolutional layers apply a convolution operation to the input, passing the result to the next layer. A convolution converts all the pixels in its receptive field into a single value. For example, if you would apply a convolution to an image, you will be decreasing the image size as well as bringing all the information in the field together into a single pixel. The final output of the convolutional layer is a vector. Based on the type of problem we need to solve and on the kind of features, we are looking to learn, we can use different kinds ofconvolutions.

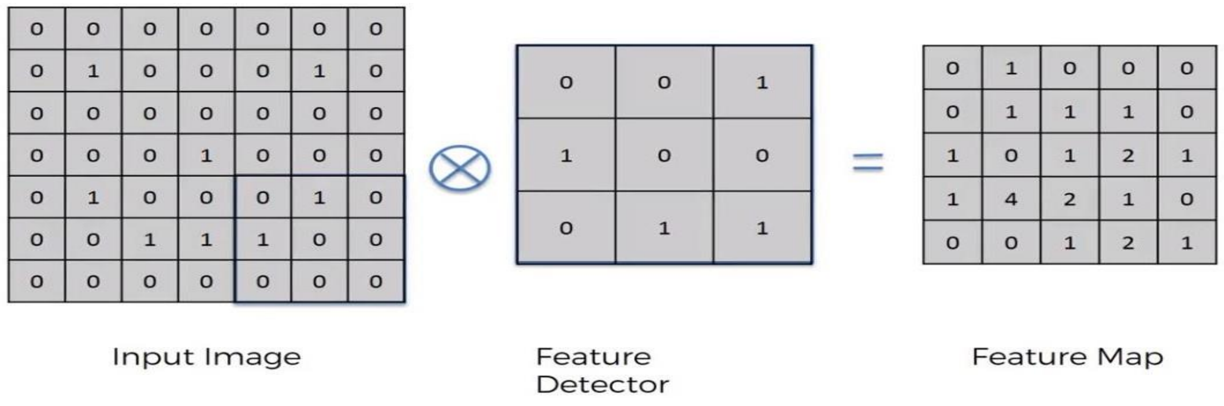


Fig 1.3 Convolutional Layer In DL

1.5.3 MAX POOLING LAYER

Max Pooling is a convolution process where the Kernel extracts the maximum value of the area it convolves. Max Pooling simply says tothe Convolutional Neural Network that we will carry forward only that information, if that is the largest information available amplitude wise. Max- pooling on a 4*4 channel using 2*2 kernel and a stride of 2: As we are convolving with a 2*2 Kernel. If we observe the first 2*2 set on which the kernel is focusingthe channel have four values 8,3,4,7. Max-Pooling picks the maximum value from that set which is “8”.

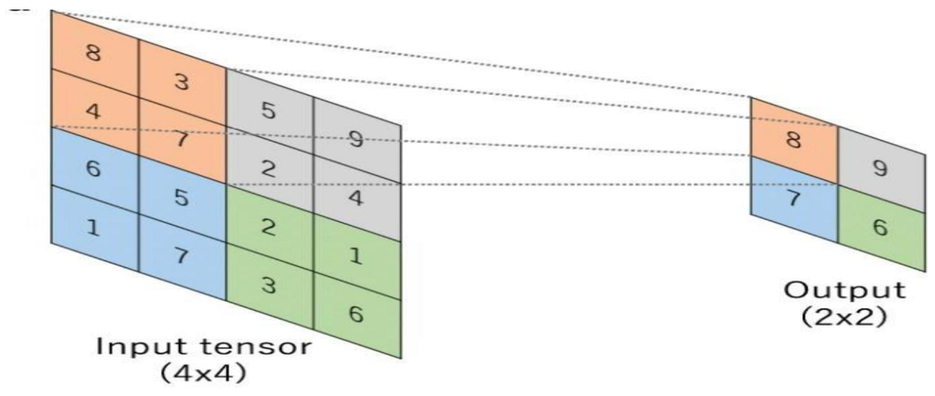


Fig 1.4 Maxpooling Layer in DL

Here in our context, we will make a kernel that amplifies the image of the cat's eye to such an extent that even after Max Pooling the predominant information is not lost. When Max Pooling now clips my pixels, the 25% pixels which are left are enough to get the information about the cat. So, there is going to be one channel or feature map which contains the information of the cat's eye no matter what happens at the benefit of reducing 75% pixels. In another way, we can say that we are filtering information that we don't want by building Kernels which can allow getting required information out through Max Pooling.

1.5.4 RELU LAYER

In the context of artificial neural networks, the rectifier or ReLU (Rectified Linear Unit) activation function is an activation function defined as the positive part of its argument:

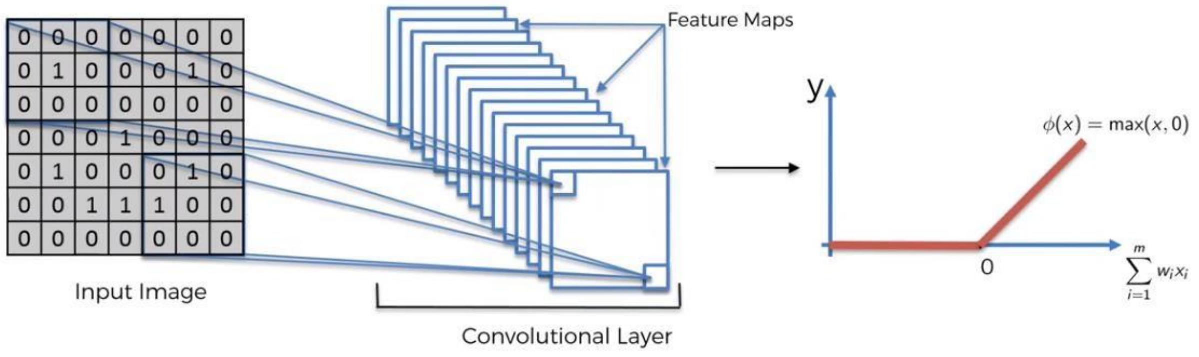


Fig 1.5 ReLU Layer in DL

where *x* is the input to a neuron. This is also known as a ramp function and is analogous to half-wave rectification in electrical engineering. This activation function started showing up in the context of visual feature extraction in hierarchical neural networks starting in the late 1960s. It was later argued that it has strong biological motivations and mathematical justifications. In 2011 it was found to enable better training of deeper networks, compared to the widely used activation functions prior to 2011, e.g., the logistic sigmoid (which is inspired by probability theory; see logistic regression) and its more practical counterpart, the hyperbolic tangent. The rectifier is, as of 2017, the most popular activation function for deep neural networks. Rectified linear units find applications in computer vision and speech recognition using deep neural nets and computational neuroscience.

1.5.5 BATCH NORMALIZATION LAYER

Training deep neural networks with tens of layers is challenging as they can be sensitive to the initial random weights and configuration of the learning algorithm. One possible reason for this difficulty is the distribution of the inputs to layers deep in the network may change after each mini-batch when the weights are updated. This can cause the learning algorithm to forever chase a moving target. This change in the distribution of inputs to layers in the network is referred to the technical name "internal covariate shift." Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks. In this post, you will discover the batch normalization method used to accelerate the training of deep learning neural networks.

- Deep neural networks are challenging to train, not least because the input from prior layers can change after weight updates.

- Batch normalization is a technique to standardize the inputs to a network, applied to ether the activations of a prior layer or inputs directly.

- Batch normalization accelerates training, in some cases by halving the epochs or better, and provides some regularization, reducing generalization error.

1.5.6 SOFT MAX LAYER

- From the figure about the detection of the dog, we found 0.95 Dog, and 0.05 Cat. The question is how these two values add up to 1. That is possible only introducing the SoftMax Function, the formula is the following:

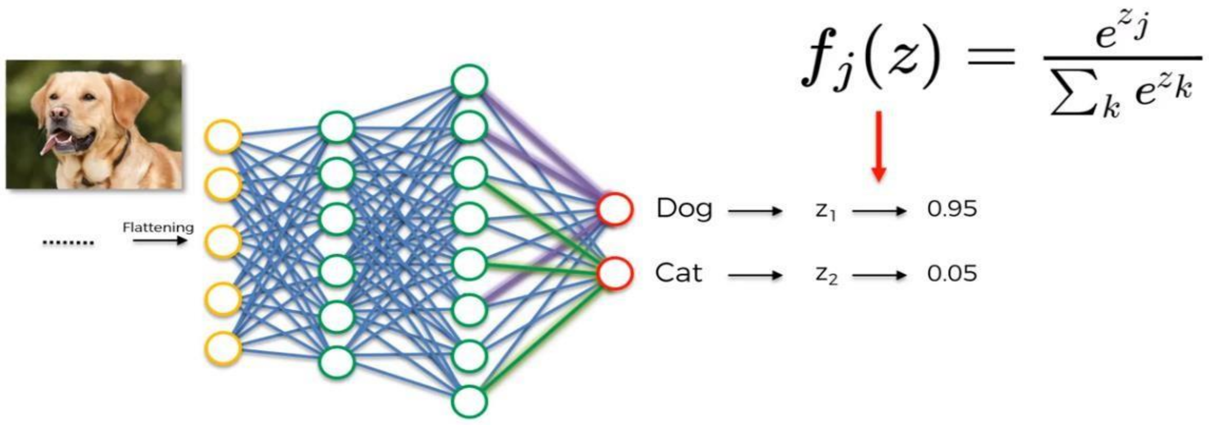


Fig 1.6 SoftMax Layer in DL

- As described by the formula in the figure above, the SoftMax Function is a generalization of the Logistic Function, and it makes sure that our prediction adds up to 1. Most of the time the Soft max Function is related to the Cross Entropy Function. In CNN, after the application of the Soft max Function, is to test the reliability of the model using as Loss Function the Cross Entropy Function, in order to maximize the performance of our neural network. There are several advantages to usingthe Cross Entropy Function. One of the best is that if for instance at the start of back propagation the output value is much smaller than the actual value, the gradient descent will be very slow. Because Cross Entropy uses the logarithm, it helps the network to assess even large errors.

1.5.7 FULLY CONNECTED LAYER

The Flattened vector that we described above, now is used as an input in a Fully Connected ANN. With fully connected we mean that the hidden layer is fully connected. This is by definition a CNN. The purpose of this is to combine our features into more attributes to predict the classes even better. In fact, combining more attributes (e.g., edge detect, blur detect, emboss detect) help to predict better the images. The figure below shows as an example the neurons activated for a Dog.

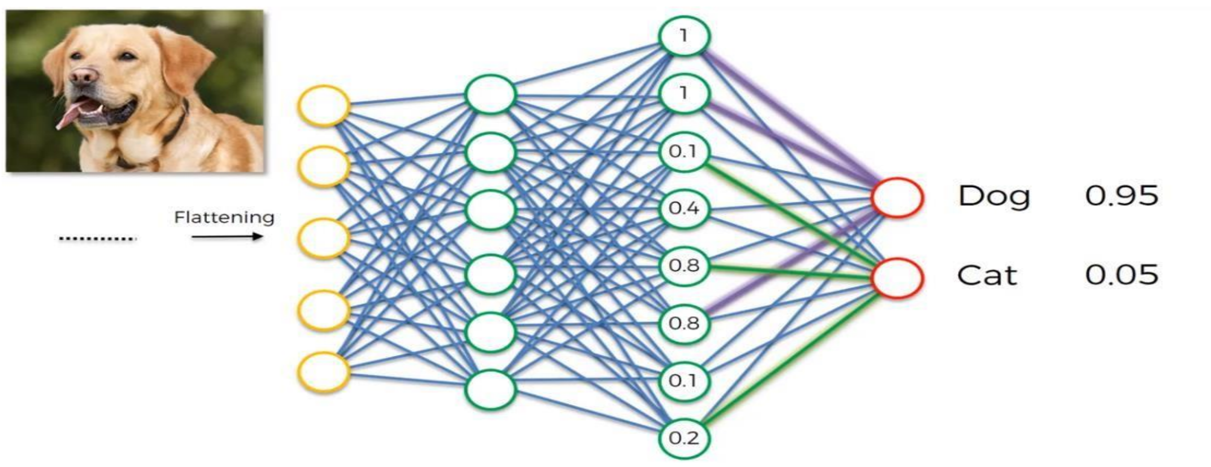


Fig 1.7 Fully Connected Layer in DL

The links between neurons and output coloured in violet for dog, and green for cat, tell us which are the important neurons for dog and cat respectively. The figure below, summarize all the steps till now considered.

1.5.8 SUMMARY OF ALL LAYERS IN DL

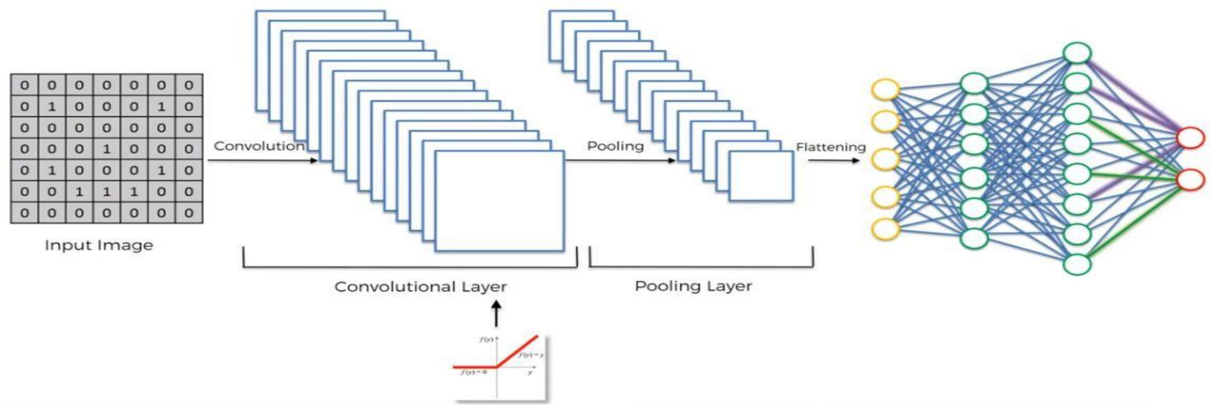


Fig 1.8 Summary of all Layers in DL

1.6 TRANSFER LAERNING AND WORKING

The reuse of a pre-trained model on a new problem is known as transfer learning in machine learning. A machine uses the knowledge learned from a prior assignment to increase prediction about a new task in transfer learning. You could, for example, use the information gained during training to distinguish beverages when training a classifier to predict whether an image contains cuisine.

The knowledge of an already trained machine learning model is transferred to a different but closely linked problem throughout transfer learning. For example, if you trained a simple classifier to predict whether an image contains a backpack, you could use the model’s training knowledge to identify other objects such as sunglasses.

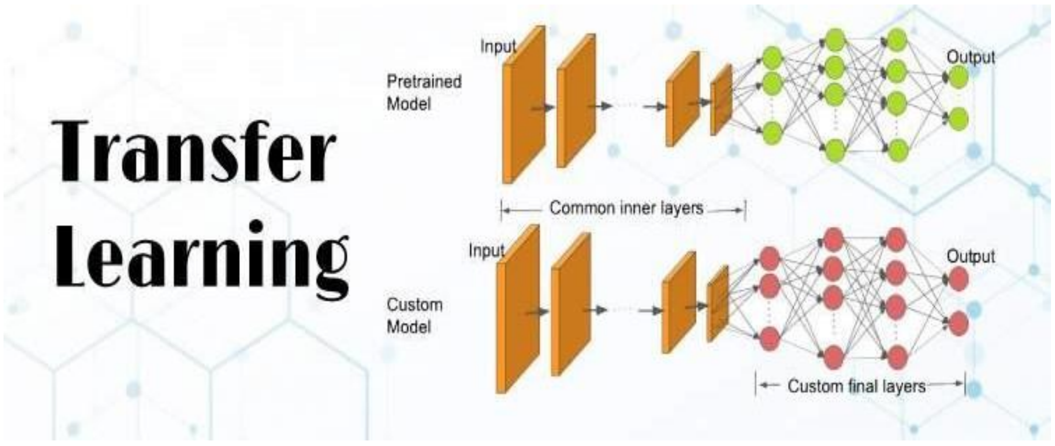


Fig 1.9 Transfer Learning

CHAPTER 2
LITERATURE SURVEY

AUTHOR	YEAR	TITLE	DESCRIPTION
Pandia Rajan Jeyaraj1· Edward Rajan Samuel Nadar1	2019	Computer-assisted medical image classification for early diagnosis of oral cancer employing deep learning algorithm	In this research work, we have developed a deep learning algorithm for automated, computer-aided oral cancer detecting system by investigating patient hyperspectral images. Methods To validate the proposed regression-based partitioned deep learning algorithm, we compare the performance with other techniques by its classification accuracy, specificity, and sensitivity. For the accurate medical

			image classification objective, we demonstrate a new structure of partitioned deep Convolution Neural Network (CNN) with two partitioned layers for labeling and classify by labeling region of interest in multidimensional hyperspectral image.
Shipu Xu 1,2, Chang Liu 3, Yongshuo Zong 4, Sirui Chen 4, Yiwen lu4, Longzhi Yang 5, (senior member, IEEE), Eddie y. k. ng 6, Yongtong wang 4, Yunsheng wang 2, Yong Liu 2, Wenwen hu 2, and Chenxi Zhang 1.	2019	An Early Diagnosis of Oral Cancer based on Three-Dimensional Convolutional Neural Networks	In this paper, we established a 3DCNNs-based image processing algorithm for the early diagnosis of oral cancers, which was compared with a 2DCNNs-based algorithm. The 3D and 2D CNNs were constructed using the same hierarchical

			structure to profile oral tumors as benign or malignant. Our results showed that 3DCNNs with dynamic characteristics of the enhancement rate image performed better than 2DCNNS with single enhancement sequence for the discrimination of oral cancer lesions.
Nabihah Haron, BSc,1 Rosnah binti Zain, MS,2,3 Anand Ramanathan, MDS,3,4 Mannil Thomas Abraham, MDS,5,6 Chee Sun Liew, PhD,7 Kheng Ghee Ng, BSc,7 Lai Choo Cheng, MCD,6 Rozihan binti Husin, MCM,6 Sherrie Mei Yee Chong, MClinDent,5,6 Logesvari A/P Thangavalu, MClinDent,6 Azizah Mat, MCD,6 Hasmah binti Ismail,	2020	m-Health for Early Detection of Oral Cancer in Low- and Middle-Income Countries	A mobile phone app named MeMoSA was developed and the feasibility of integrating this for documentation of oral lesions, and communication between dentists and specialists for management decisions were

BDS,6 Sumitha A/P Mahalingam, BDS,6 and Sok Ching Cheong, PhD1			evaluated. The experience of dentists and specialists in using MeMoSA was determined using qualitative questionnaires.
Seda CamalanID1 *, Muhammad Khalid Khan Niazi1, Aaron C. Moberly2, Theodoros Teknos3, Garth Essig2, Charles Elmaraghy2, Nazhat Taj-Schaal4, Metin N. Gurcan1	2020	OtoMatch: Content-based eardrum image retrieval using deep learning	We present a method that enables the conversion of any convolutional neural network (trained for classification) into an image retrieval model. As a proof of concept, we converted a pre-trained deep learning model into an image retrieval system. We accomplished this by changing the fully connected layers into lookup tables.

			LIMITATIONS: • Only retrieves similar images and does not classify the image to a particular category

- We improved on earlier similar algorithmic efforts by leveraging ALEXNET, a combatable and lasted CNN architecture.

- The limitations in the Otomatch are overcome by being able to identify the effusion, Normal and the Tympanostomy Tube

CHAPTER 3

PROJECT DESCRIPTION

3.1 RGB COLOR IMAGE

The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad arrayof colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue.

The main purpose of the RGB color model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography. Before the electronic age, the RGB color model already had a solid theory behind it, based in human perception of colors.

RGB is a device-dependent color model: different devices detect or reproduce a given RGB value differently, since the color elements (such as phosphors or dyes) and their response to the individual R, G, and B levels vary from manufacturer to manufacturer, or even in the same device over time. Thus, an RGB value does not define the same color across devices without some kind of color management.

Typical RGB input devices are color TV and video cameras, image scanners, and digital cameras. Typical RGB output devices are TV sets of various technologies (CRT, LCD, plasma,etc.), computer and mobilephone displays, video projectors, multicolor LED displays, and large screens such as Jumbotron. Color printers, on the other hand, are not RGB devices,but subtractive color devices (typically CMYK color model).

Fig 3.1 Example of RGB Color Image

3.2 GRAYSCALE

In photography and computing, a grayscale or greyscale digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of grey, varying from black at the weakest intensity to white at the strongest.

Greyscale images are distinct from one-bit bi-tonal black-and-white images, which in the context of computer imaging are images with only the two colors, black, and white (also called bilevel or binary images). Grayscale images have many shades of grey in between. Grayscale images are also called monochromatic, denoting the presence of only one (mono) color (chrome).

Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum (e.g., infrared, visible light, ultraviolet, etc.), and in such cases they are monochromatic proper when only a given frequency is captured. But also, they can be synthesized from a full color image; see the section about converting to grayscale.

Fig 3.2 Example of grayscale image

3.3 MORPHOLOGICAL OPERATIONS

To find the exact features we have to segment the lung region from the chest CT scan image for easy computation. For segmenting the lung region from the chest CT scan image morphological operation is carried out.

We defined an image as an (amplitude) function of two, real (coordinate) variables a(x,y) or two, discrete variables a[m,n]. An alternative definition of an image can be based on the notion that an image consists of a set (or collection) of either continuous or discrete coordinates. In a sense the set corresponds to the points or pixels that belong to the objects in the image. This is illustrated in figure below which contains two objects or sets **A** and **B**. Note that the coordinate system is required. For the moment we will consider the pixel values to be binaryas discussed in Section 2.1 and 9.2.1. Further we shall restrict our discussion to discrete space (Z^2). More general discussions can be found in.

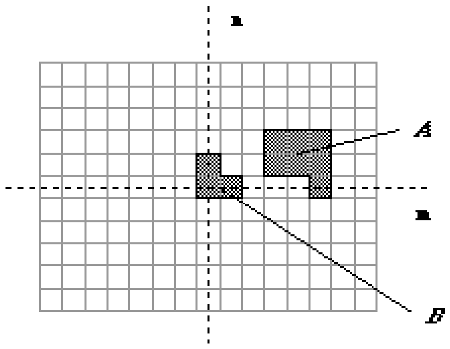


Fig 3.3 Example for Border corrected mask

A binary image containing two object sets **A** and **B**.

The object **A** consists of those pixels a that share some common property:

$$A = \{\alpha \; property(\alpha) == TRUE\}$$

As an example, object **B** consists of {[0,0], [1,0], [0,1]}.

The background of **A** is given by **A**^{c} (the complement of **A**) which is defined as those elements that are not in **A**:

$$A^C = \{\alpha | \alpha \notin A\}$$

We introduced the concept of neighbourhood connectivity. We now observe that if an object **A** is defined on the basis of C-connectivity (C=4, 6, or 8) then the background **A**^{c} has a connectivity given by 12 - C. The necessity for this is illustrated for the Cartesian grid in Figure 36.

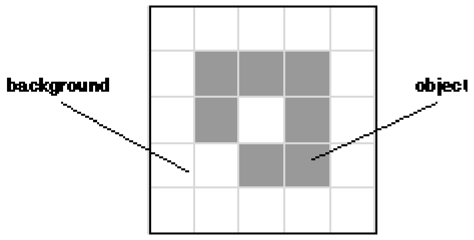


Fig 3.4 illustration for Cartesian grid

A binary image requiring careful definition of object and background connectivity.

3.3.1 FUNDAMENTAL DEFINITIONS

The fundamental operations associated with an object are the standard set operations union, intersection, and complement $\{\cup, \cap, ^c\}$ plus translation:

- Translation - Given a vector **x** and a set **A**, the translation, **A** + **x**, is defined as:

$$A + x = \{\alpha + x | \alpha \in A\}$$

Note that, since we are dealing with a digital image composed of pixels at integer coordinate positions (Z^2), this implies restrictions on the allowable translation vectors **x**.

The basic Minkowski set operations--addition and subtraction--can now be defined. First, we note that the individual elements that comprise **B** are not only pixels but also vectors as they have a clear coordinate position with respectto [0,0]. Given two sets **A** and **B:**

Minkowski addition - $A \oplus B = \cup_{\beta g B}(A + \beta)$

Minkowski subtraction - $A \ominus B = \bigcap_{\beta g B}(A + \beta)$

3.3.2 DILATION AND EROSION

From these two Minkowski operations we define the fundamental mathematical morphology operations dilation and erosion:

Dilation – $D(A, B) = A \oplus B = \bigcup_{\beta g B}(A + B)$

Erosion – $E(A, B) = A \ominus (-B) = \bigcap_{\beta g B}(A - \beta)$

where – $B = \{-\beta \beta \epsilon B\}$. These two operations are illustrated in figures below for the objects defined

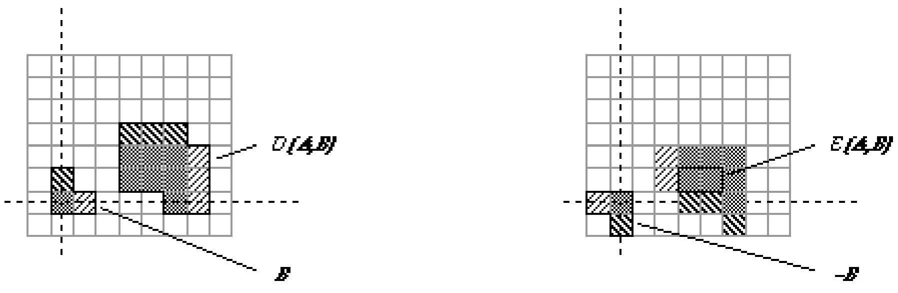


Fig 3.5 (a) Dilation D(A,B) **(b)** Erosion E(A,B)

A binary image containing two object sets **A** and **B**. The three pixels in **B** are "color-coded" as is their effect in the result.

While either set **A** or **B** can be thought of as an "image", **A** is usually considered as the image and **B** is called a structuring element. The structuring element is to mathematical morphology what the convolution kernel is to linear filter theory.

Dilation, in general, causes objects to dilate or grow in size; erosion causes objects to shrink. The amount and the way that they grow or shrink depend upon the choice of the structuring element. Dilating or eroding without specifying the structural element makes no more sense than trying to lowpass filter an image without specifying the filter. The two most common structuring elements (given a Cartesian grid) are the 4-connected and 8-connected sets, $\mathbf{N_4}$ and $\mathbf{N_8}$.

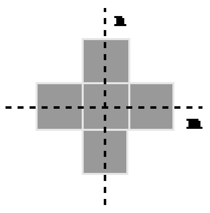
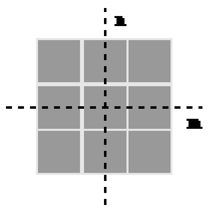



Fig 3.6 (a) 4-connected sets N_4 **(b)** 8-connected sets N_8The standard structuring elements $\mathbf{N_4}$ and $\mathbf{N_8}$.

Dilation and erosion have the following properties:

Commutative - $D(A, B) = A \oplus B = B \oplus A = D(B, A)$

Non-Commutative - $E(A, B) \neq E(B, A)$

Associative - $A \oplus (B \oplus C) = (A \oplus B) \oplus C$

Translation Invariance - $A \oplus (B + x) = (A \oplus B) + x$

Duality - $D^c(A, B) = E(A^c, -B)$

$E^c(A, B) = D(A^c, -B)$

With **A** as an object and $\mathbf{A}^c$ as the background, eq. says that the dilation of an object is equivalent to the erosion of the background. Likewise, the erosion of the object is equivalent to the dilation of the background.

Except for special cases:

Non-Inverses – $D(E(A, B), B) \neq A \neq E(D(A, B), B)$

Erosion has the following translation property:

Translation Invariance - $A \ominus (B + x) = (A + x) \ominus B = (A \ominus B) + x$

Dilation and erosion have the following important properties. For any arbitrary structuring element **B** and two image objects $\mathbf{A}_1$ and $\mathbf{A}_2$ such that $\mathbf{A}_1 \subset \mathbf{A}_2$ ($\mathbf{A}_1$ is a proper subset of $\mathbf{A}_2$):

Increasing in **A** - $D(A_1, B) \subset D(A_2, B)$

$$E(A_1, B) \subset E(A_2, B)$$

For two structuring elements $\mathbf{B}_1$ and $\mathbf{B}_2$ such that $\mathbf{B}_1 \subset \mathbf{B}_2$:

Decreasing in **B** – $E(A, B_1) \supset E(A, B_2)$

The decomposition theorems below make it possible to find efficient implementations for morphological filters.

Dilation - $A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C) = (B \cup C) \oplus A$

Erosion - $A \ominus (B \cup C) = (A \ominus B) \cap (A \ominus C)$

Erosion - $(A \ominus B) \ominus C = A \ominus (B \oplus C)$

Multiple Dilations – $nB = (\underbrace{B \oplus B \oplus B \cdots \oplus B}_{\text{n times}})$

An important decomposition theorem is due to Vincent. First, we require some definitions. A convex set (in R^2) is one for which the straight line joining any two points in the set consists of points that are also in the set. Care must obviously be taken when applying this definition to discrete pixels as the concept of a "straight line" must be interpreted appropriately in Z^2. A set is bounded if each of its elements has a finite magnitude, in this case distance to the origin of the coordinate system. A set is symmetric if **B** = **-B**. The sets $\mathbf{N_4}$ and $\mathbf{N_8}$ in Figure 38 are examples of convex, bounded, symmetric sets.

Vincent's theorem, when applied to an image consisting of discrete pixels, states that for a bounded, symmetric structuring element **B** that contains no holes and contains its own center, $[0,0] \in B$:

$$D(A, B) = A \oplus B = A \cup (\partial A \oplus B)$$

where ∂**A** is the contour of the object. That is, ∂ **A** is the set of pixels that have a background pixel as a neighbor. The implication of this theorem is that it is not necessary to process all the pixels in an object in order to compute a dilation or (using eq.) an erosion. We only have to process the boundary pixels. This also holds for all operations that can be derived from dilations and erosions. The processing of boundary pixels instead of object pixels means that, except for pathological images, computational complexity can be reduced from $O(N^2)$ to O(N) for an N x N image. A number of "fast" algorithms can be found in the literature that are based on this result. The simplest dilation and erosion algorithms are frequently described as follows.

- Dilation - Take each binary object pixel (with value "1") and set all background pixels (with value "0") that are C-connected to that object pixel to the value "1".
- Erosion - Take each binary object pixel (with value "1") that is C-connected to a background pixel and set the object pixel value to "0".

Comparison of these two procedures to eq. where $\mathbf{B} = \mathbf{N}_{C=4}$ or $\mathbf{N}_{C=8}$ shows that they are equivalent to the formal definitions for dilation and erosion.

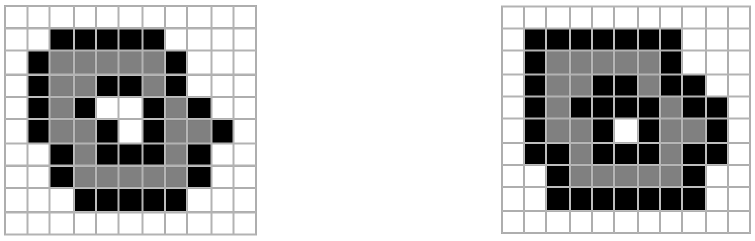


Fig 3.7 (a) B = N_4 **(b)** B= N_8

Illustration of dilation. Original object pixels are in gray; pixels added through dilation are in black.

3.3.3 BOOLEAN CONVOLUTION

An arbitrary binary image object (or structuring element) **A** can be represented as:

$$A \leftrightarrow \sum_{k=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} a[j,k] \cdot \delta[m-j, n-k]$$

where ∑ and * are the Boolean operations OR and AND as defined in eqs. (81) and (82), a[j,k] is a characteristic function that takes on the Boolean values "1" and "0" as follows:

$$a[j,k] = \{ \begin{matrix} 1 & a \in A \\ 0 & a \notin A \end{matrix}$$

and d[m,n] is a Boolean version of the Dirac delta function that takes on the Boolean values "1" and "0" as follows:

$$\delta[j,k] = \{ \begin{matrix} 1 & j = k = 0 \\ 0 & otherwise \end{matrix}$$

Dilation for binary images can therefore be written as:

$$D(A, B) = \sum_{k=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} a[j, k]. \delta[m - j, n - k]$$

which, because Boolean OR and AND are commutative, can also be written as

$$D(A, B) = \sum_{K=-\infty}^{+\infty} \sum_{J=-\infty}^{+\infty} a[m - j, n - k]. b[j, k] = b \otimes a = D(B, A)$$

Using De Morgan's theorem:

$\overline{(a+b)} = \overline{a} \cdot \overline{b}$ and $\overline{(a \cdot b)} = \overline{a} + \overline{b}$

on eq. together with eq., erosion can be written as:

$$E(A, B) = \prod_{k=-\infty}^{+\infty} \prod_{j=-\infty}^{+\infty} (a[m - j, n - k] + \overline{b}[-j, -k])$$

Thus, dilation and erosion on binary images can be viewed as a form of convolution over a Boolean algebra.

When convolution is employed, an appropriate choice of the boundary conditions for an image is essential. Dilation and erosion--being a Boolean convolution--are no exception. The two most common choices are that either everything outside the binary image is "0" or everything outside the binary image is "1".

3.3.4 OPENING AND CLOSING

We can combine dilation and erosion to build two important higher order operations:

Opening – $O(A, B) = A^{O}B = D(E(A, B), B)$

Closing – $C(A, B) = A \blacksquare B = E(D(A, -B), -B)$

The opening and closing have the following properties:

Duality – $C^c(A, B) = O(A^c, B)$

$O^c(A, B) = C(A^c, B)$

Translation – $O(A + x, B) = O(A, B) + x$

$C(A + x, B) = C(A, B) + x$

For the opening with structuring element **B** and images **A**, $\mathbf{A}_1$, and $\mathbf{A}_2$, where $\mathbf{A}_1$ is a sub image of $\mathbf{A}_2$ ($\mathbf{A}_1 \subseteq \mathbf{A}_2$):

Antiextensivity – $O(A, B) \subseteq A$

Increasing monotonicity – $O(A_1, B) \subseteq O(A_2, B)$

Idempotence – $O(O(A, B), B) = O(A, B)$

For the closing with structuring element B and images **A**, $\mathbf{A}_1$, and $\mathbf{A}_2$, where $\mathbf{A}_1$ is a sub image of $\mathbf{A}_2$ ($\mathbf{A}_1 \subseteq \mathbf{A}_2$):

Extensivity - $A \subseteq C(A, B)$

Increasing monotonicity – $C(A_1, B) \subseteq C(A_2, B)$

Idempotence – $C(C(A, B), B) = C(A, B)$

The two properties given by eqs. and are so important to mathematical morphology that they can be considered as the reason for defining erosion with -**B** instead of **B** in eq.

3.3.5 HIT AND MISS OPERATION

The hit-or-miss operator was defined by Serra but we shall refer to it as the hit-and-miss operator and define it as follows. Given an image **A** and two structuring elements $\mathbf{B}_1$ and $\mathbf{B}_2$, the set definition and Boolean definition are:

$$\text{hit-and-Miss} - HitMiss(A, B_1, B_2) = \begin{cases} E(A, B_1) \cap E^c(A^c, B_2) \\ E(A, B_1) \cdot \overline{E(\overline{A}, B_2)} \\ E(A, B_1) - E(\overline{A}, B_2) \end{cases}$$

where **B**$_1$ and **B**$_2$ are bounded, disjoint structuring elements. (Note the use of the notation from eq. (81).) Two sets are disjoint if **B**$_1 \cap$ **B**$_2 = \varnothing$, the empty set. In an important sense the hit-and-miss operatories the morphological equivalent of template matching, a well-known technique for matching patterns based upon cross-correlation. ere, we have a template **B**$_1$ for the object and a template **B**$_2$ for the background.

3.3.6 SUMMARY OF THE BASIC OPERATIONS

The results of the application of these basic operations on a test image are illustrated below. The various structuring elements used in the processing are defined. The value "-" indicates a "don't care". All three structuring elements are symmetric.

$$B = N_8 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad B_1 = \begin{bmatrix} - & - & - \\ - & 1 & - \\ - & - & - \end{bmatrix} \qquad B_2 = \begin{bmatrix} - & 1 & - \\ 1 & - & 1 \\ - & 1 & - \end{bmatrix}$$

(a) **(b)** **(c)**

Structuring elements **B**, **B**$_1$, and **B**$_2$ that are 3 x 3 and symmetric. The results of processing are shown in Figure 41 where the binary value "1" is shown in black and the value "0" in white.

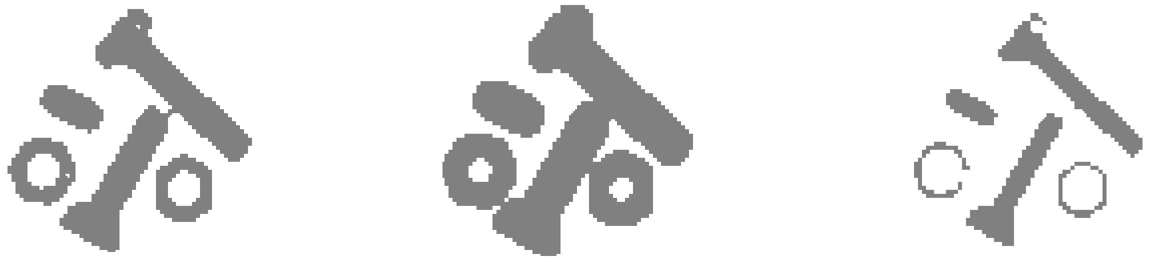
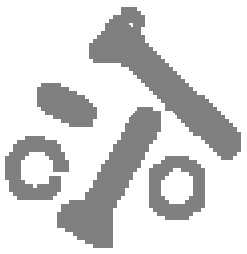
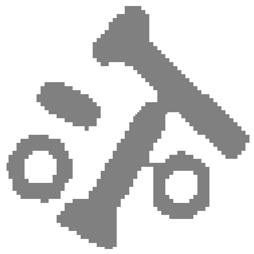


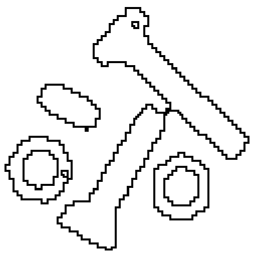
Fig 3.8 (a) Image **A** **(b)** Dilation with 2**B** **(c)** Erosion with 2**B**



(d) Opening with 2**B**



(e) Closing with 2**B**



(f) it-and-Miss with $\mathbf{B}_1$ and $\mathbf{B}_2$

Examples of various mathematical morphology operations.

The opening operation can separate objects that are connected in a binary image. The closing operation can fill in small holes. Both operations generate a certain amount of smoothing on an object contour given a "smooth" structuring element. The opening smoothes from the inside of the object contour and the closing smoothes from the outside of the object contour. The hit-and- miss example has found the 4-connected contour pixels. An alternative method to find the contour is simply to use the relation:

4-connected contour – $\partial A = A - E(A, N_8)$

or

8-connected contour – $\partial A = A - E(A, N_4)$

3.3.7 SKELETON

The informal definition of a skeleton is a line representation of an object that is:

i) one-pixel thick,

ii) through the "middle" of the object, and,

iii) preserves the topology of the object.

These are not always realizable.

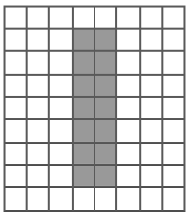
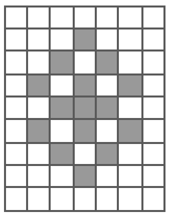



Fig 3.9 (a) One-pixel thick **(b)** Through the "Middle" of the object

Counter examples to the three requirements.

In the first example, it is not possible to generate a line that is one pixel thick and in the center of an object while generating a path that reflects the simplicity of the object. In Figure 42b it is not possible to remove a pixel from the 8-connected object and simultaneously preserve the topology--the notion of connectedness--of the object. Nevertheless, there are a variety of techniques that attempt to achieve this goal and to produce a skeleton.

A basic formulation is based on the work of Lantuéjoul. The skeleton subset $\mathbf{S}_k(\mathbf{A})$ is defined as:

Skeleton subsets - $S_k(A) = E(A, kB) - [E(A, kB)^\circ B] \qquad k = 0,1, \dots, K$

where K is the largest value of k before the set $\mathbf{S}_k(\mathbf{A})$ becomes empty. (From eq. $E(A,\ kB)^\circ B \subseteq E(A,\ kB)$). The structuring element **B** is chosen (in Z^2) to approximate a circular disc, that is, convex, bounded and symmetric. The skeleton is then the union of the skeleton subsets:

Skeleton – $S(A) = \bigcup_{k=0}^{K} S_k(A)$

An elegant side effect of this formulation is that the original object can be reconstructed given knowledge of the skeleton subsets $\mathbf{S}_k(\mathbf{A})$, the structuring element **B**, and K:

Reconstruction – $A = \bigcup_{k=0}^{K} (S_k(A) \oplus kB)$

This formulation for the skeleton, however, does not preserve the topology, a requirement described in eq.

An alternative point-of-view is to implement a thinning, an erosion that reduces the thickness of an object without permitting it to vanish. A general thinning algorithm is based on the hit-and-miss operation:

Thinning – $Thin(A. B_1, B_2) = A - HitMass(A, B_1, B_2)$

Depending on the choice of $\mathbf{B}_1$ and $\mathbf{B}_2$, a large variety of thinning algorithms--and through repeated application skeletonizing algorithms--can be implemented.

A quite practical implementation can be described in another way. If we restrict ourselves to a 3 x 3 neighborhood, similar to the structuring element $\mathbf{B} = \mathbf{N}_8$ in Figure 40a, then we can view the thinning operation as a window that repeatedly scans over the (binary) image and sets the center pixel to "0" under certain conditions. The center pixel is not changed to "0" if and only if:

i) an isolated pixel is found

ii) removing a pixel would change the connectivity

iii) removing a pixel would shorten a line

As pixels are (potentially) removed in each iteration, the process is called a conditional erosion. In general, all possible rotations and variations have to be checked. As there are only 512 possible combinations for a 3 x 3 window on a binary image, this can be done easily with the use of a lookup table.

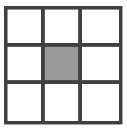
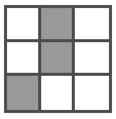
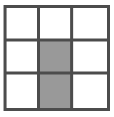




Fig 3.10 (a) Isolated pixel **(b)** Connectivity pixel **(c)** End pixelTest

conditions for conditional erosion of the center pixel.

If only condition;

(i) is used then each object will be reduced to a single pixel. This is useful if we wish to count the number of objects in an image. If only condition

(ii) is used then holes in the objects will be found. If conditions (i + ii) are used each object will be reduced to either a single pixel if it does not contain a hole or to closed rings if it does contain holes. If conditions (i + ii + iii) are used then the "complete skeleton" will be generated as an approximation to eq.

3.3.8 PROPAGATION

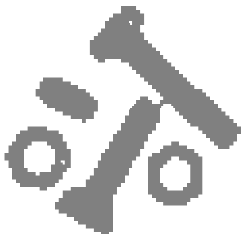
It is convenient to be able to reconstruct an image that has "survived" several erosions or to fill an object that is defined, for example, by a boundary. The formal mechanism for this has several names including region- filling, reconstruction, and propagation. The formal definition is given by the following algorithm. We start with a seed image $\mathbf{S}^{(0)}$, a mask image **A**, and a structuring element **B**. We then use dilations of **S** with structuring element **B** and masked by **A** in an iterative procedure as follows:

$$\text{Iteration k} - S^{(k)} = [S^{(K-1)} \oplus B] \cap A \qquad until \quad S^{(k)} = S^{(k-1)}$$

With each iteration the seed image grows (through dilation) but within the set (object) defined by **A**; **S** propagates to fill **A**. The most common choices for **B** are **N_4** or **N_8**. Several remarks are central to the use of propagation. First, in a straightforward implementation, as suggested by eq., the computational costs are extremely high. Each iteration requires O(N^2) operations for an N x N image and with the required number of iterations this can lead to a complexity of O(N^3). Fortunately, a recursive implementation of the algorithm exists in which one or two passes through the image are usually sufficient, meaning a complexity of O(N^2). Second, although we have not paid much attention to the issue of object/background connectivity until now, it is essential that the connectivity implied by **B** be matched to the connectivity associated with the boundary definition of **A** (see eqs. and). Finally, as mentioned earlier, it is important to make the correct choice ("0" or "1") for theboundary condition of the image. The choice depends upon the application.

3.3.9 SUMMARY OF SKELETON AND PROPAGATION

The application of these two operations on a test image, the skeleton operation is shown with the end pixel condition (eq. i+ii+iii) and without the end pixel condition (eq. i+ii). The propagation operation is illustrated in Figure 44c. The original image, shown in light grey, was eroded by E(**A**,6**N_8**) to produce the seed image shown in black. The original was then used as the mask image to produce the final result. The border value in both images was "0".



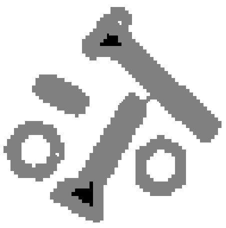
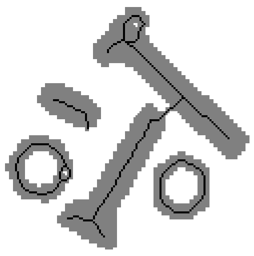


Fig 3.11(a) Skeleton with end pixels **(b)** Skeleton without end pixels

Original = light gray Mask = light gray

↓Skeleton = black ↓ ↓ Seed = black



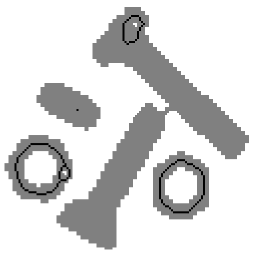
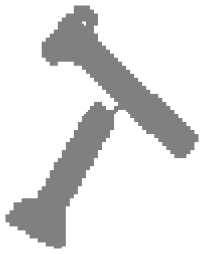


Fig3.12 (a) Skeleton with end pixels **(b)** Skeleton without end pixels



(c) Propagation with $\mathbf{N}_8$

3.4 GRAY-VALUE MORPHOLOGICAL PROCESSING

The techniques of morphological filtering can be extended to gray-level images. To simplify matters we will restrict our presentation to structuring elements, **B**, that comprise a finite number of pixels and are convex and bounded.

Now, however, the structuring element has grey values associated with every coordinate position as does the image **A**.

* Gray-level dilation, $D_G(*)$, is given by:

Dilation - $D_G(A, B) = \max_{[j,k]\in B} \{a[m - j, n - k] + b[j, k]\}$

For a given output coordinate [m,n], the structuring element is summed with a shifted version of the image and the maximum encountered over all shifts within the J x K domain of **B** is used as the result. Should the shifting require values of the image **A** that are outside the M x N domain of **A**, then a decision must be made as to which model for image extension, should be used.

* Gray-level erosion, $E_G(*)$, is given by:

Erosion - $E_G(A, B) = \min_{[j,k]\in B} \{a[m + j, n + k] - b[j, k]\}$

The duality between gray-level erosion and gray-level dilation--the gray-level counterpart of eq. --is somewhat more complex than in the binary case:

Duality - $E_G(A, B) = -D_G(-\tilde{A}, B)$

where " $-\tilde{A}$ " means that a[j,k] -> -a[-j,-k].

The definitions of higher order operations such as gray-level opening and gray-level closing are:

Opening - $O_G(A, B) = D_G(E_G(A, B), B)$

Closing - $C_G(A, B) = -O_G(-A, -B)$

The important properties that were discussed earlier such as idempotence, translation invariance, increasing in A, and so forth are also applicable to gray level morphological processing. The details can be found in Giardina and Dougherty.

In many situations the seeming complexity of gray level morphological processing is significantly reduced through the use of symmetric structuring elements where b[j,k] = b[-j,-k]. The most common of these is based on the use of **B** = constant = 0. For this important case and using again the domain [j,k] ⊂ **B**, the definitions above reduce to:

Dilation - $D_G(A,B) = \max\limits_{[j,k]\in B}\{a[m-j,n-k]\} = \max\limits_{0}(A)$

Erosion - $E_G(A,B) = \min\limits_{[j,k]\in B}\{a[m-j,n-k]\} = \min\limits_{0}(A)$

Opening - $O_G(A,B) = \max\limits_{B}(\min\limits_{B}(A))$

Closing - $C_G(A,B) = \min\limits_{B}(\max\limits_{B}(A))$

The remarkable conclusion is that the maximum filter andthe minimum filter, are gray-level dilation and gray-level erosion for the specific structuring element given by the shape of the filter window with the gray value "0" inside the window.

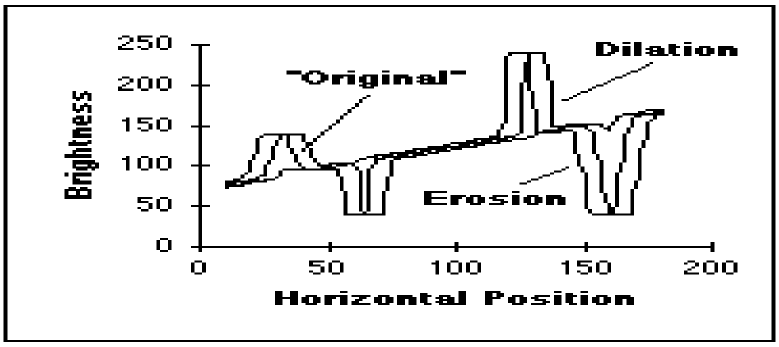


Fig 3.13 Effect of 15x1 dilation and erosion

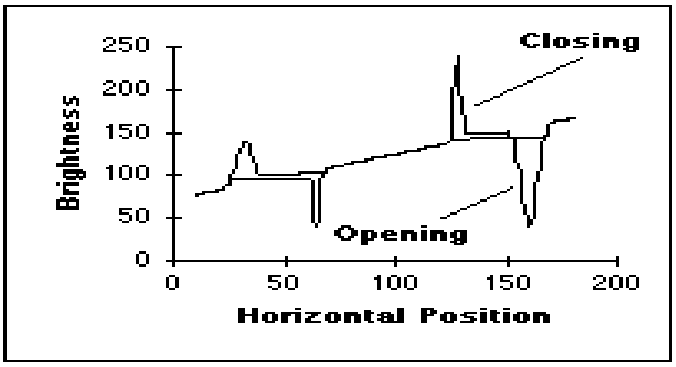


Fig 3.14 Effect of 15 x 1 opening and closing

3.4.1 MORPHOLOGICAL FILTERING

For a rectangular window, J x K, the two-dimensional maximum or minimum filter is separable into two, one-dimensional windows. Further, a one-dimensional maximum or minimum filter can be written in incremental form. This means that gray-level dilations and erosions have a computational complexity per pixel that is O(constant), that is, independent of J and K.

The operations defined above can be used to produce morphological algorithms for smoothing, gradient determination and a version of the Laplacian. All are constructed from the primitives for gray-level dilation and gray- level erosion and in all cases the maximum and minimum filters are taken over the domain $[j, k] \in B$.

3.4.2 MORPHOLOGICAL SMOOTHING

This algorithm is based on the observation that a gray- level opening smoothes a grey-value image from above the brightness surface given by the function a[m,n] and the grey-level closing smooths from below. Weuse a structuring element **B** based on eqs. And

$$MorphSmooth(A, B) = C_G(O_G(A, B), B)$$

$$= \min(\max(\max(\min(A))))$$

Note that we have suppressed the notation for the structuring element **B** under the max and min operations to keep the notation simple. Its use, however, is understood.

3.4.3 MORPHOLOGICAL GRADIENT

For linear filters the gradient filter yields a vector representation (eq. (103)) with a magnitude (eq. (104)) and direction (eq. (105)). The version presented here generates a morphological estimate of the gradient magnitude:

$$\begin{aligned} Gradient(a, b) &= 1/2(D_G(A, B) - E_G(A, B)) \\ &= 1/2(\max(A) - \min(A)) \end{aligned}$$

3.4.4 MORPHOLOGICAL LAPLACIAN

The morphologically-based Laplacian filter is defined by:

$$\begin{aligned} Laplacian(A, B) &= 1/2((D_G(A, B) - A) - (A - E_G(A, B))) \\ &= 1/2(D_G(A, B) + E_G(A, B) - 2A) \\ &= 1/2(\max(A) + \min(A) - 2A) \end{aligned}$$

3.4.5 SUMMARY OF MORPHOLOGICAL FILTERS

The effect of these filters is illustrated below. All images were processed with a 3 x 3 structuring element as described in eqs. through. Figure 46e was contrast stretched for display purposes and the parameters 1% and 99%.

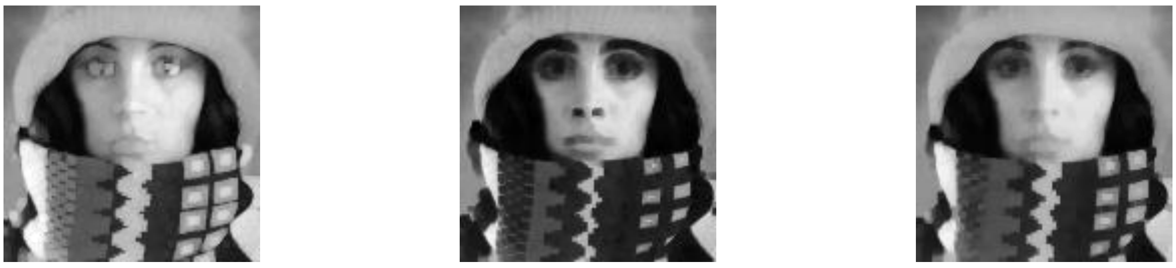
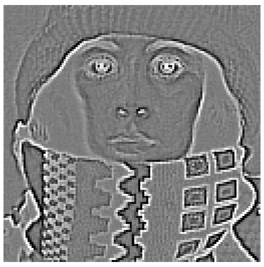


Fig 3.15 (a) Dilation **(b)** Erosion **(c)** Smoothing

(d) Gradient


(e) Laplacian

Examples of gray-level morphological filters.

3.5 BORDER CORRECTED MASK

A mask is a filter. Concept of masking is also known as spatial filtering. Masking is also known as filtering. In this concept we just deal with the filtering operation that is performed directly on the image. In image processing, a kernel, convolution matrix, or mask is a small matrix useful for blurring, sharpening, embossing, edge-detection, and more. This is accomplished by means of convolution between a kernel and an image.

The mask is created to find the exact operations in an image. So that we can identify the problems or the features which we need to find in an image.

The border corrected mask is a mask in which the edges are closed to find all the features of an image.

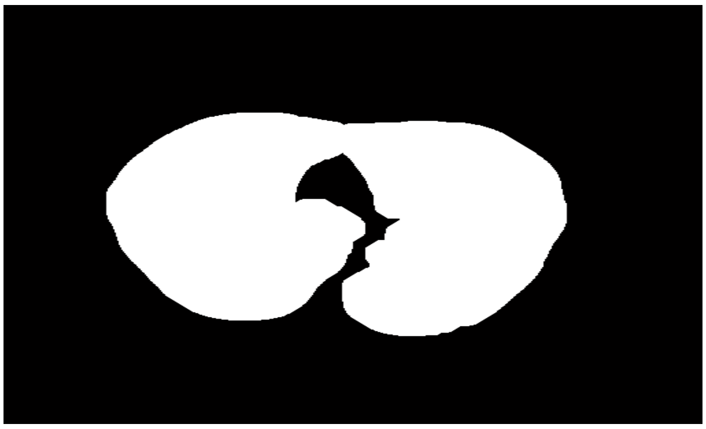


Fig 3.16 Example of border corrected mask

3.6 CONNECTED COMPONENT ANALYSIS (CCA) AND OBJECTS EXTRACTION

CCA is a well-known technique in image processing that scans an image and groups pixels in labelled components based on pixel connectivity. An eight-point CCA stage is performed to locate all the objects inside the binary image produced from the previous stage. The output of this stage is an array of N objects shows an example of the input and output of this stage.

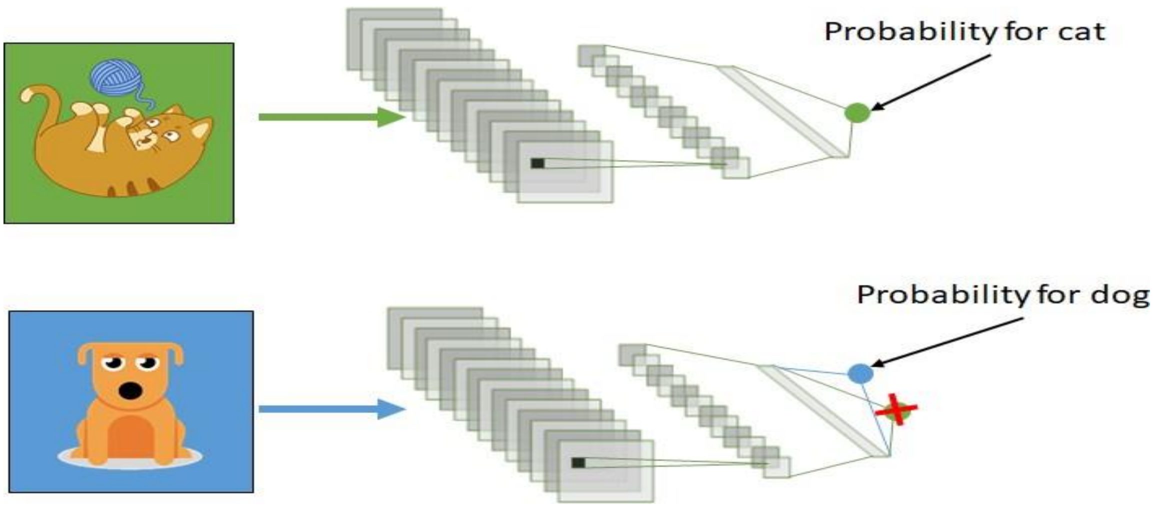


Fig 3.17 example of the input and output of CCA and object extraction

With transfer learning, we basically try to use what we've learned in one task to better understand the concepts in another. weights are being automatically being shifted to a network performing "task A" from a network that performed new "task B."

Because of the massive amount of CPU power required, transfer learning is typically applied in computer vision and natural language processing tasks like sentiment analysis.

CHAPTER 4

PROPOSED SYSTEM

4.1MODULES

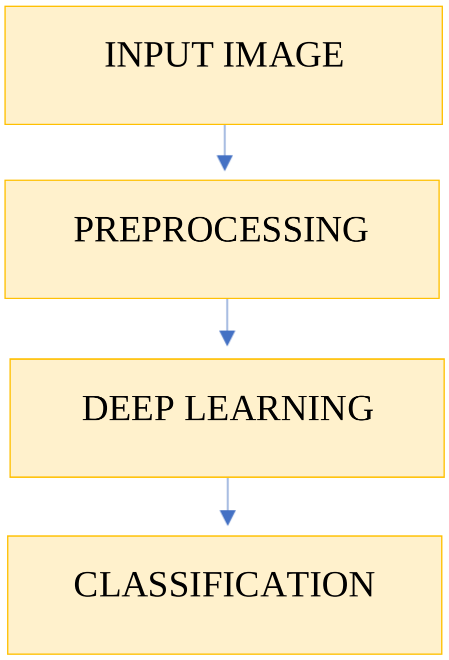


Fig 4.1 Modules flowchart

4.1.1 MODULE 1 - INPUT IMAGE

Here, we are giving the collected dataset as input from the system.

4.1.2 MODULE 2 – PREPROCESSING

The aim of pre-processing is an improvement of the image data that suppresses undesired distortions or enhances some image features relevant for further processing and analysis task.

There are 4 different types of Image Pre-Processing techniques and they are listed below.

1. Pixel brightness transformations/ Brightness corrections.

2. Geometric Transformations.
3. Image Filtering and Segmentation.
4. Fourier transform and Image re-saturation.

4.1.3 MODULE 3 – DEEP LEARNING

It is assumed that reader knows the concept of Neural Network. When it comes to Machine Learning, Artificial Neural Networks perform really well. Artificial Neural Networks are used in various classification task like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural Network. In this blog, we are going to build basic building block for CNN. Before diving into the Convolution Neural Network,let us first revisit some concepts of Neural Network. In a regular Neural Network, there are three types of layers:

1. **Input Layers:** It's the layer in which we give input to our model. The number of neurons in this layer is equal to total number of features in our data (number of pixels incase of an image).
2. **Hidden Layer:** The input from Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layers can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by addition of learnable biases followed by activation function which makes the network nonlinear.

3. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or soft max which converts the output of each class into probability score of each class.

The data is then fed into the model and output from each layer is obtained this step is called feed forward, we then calculate the error using an error function, some common error functions are cross entropy, square loss error etc. After that, we back propagate into the model by calculating the derivatives. Thisstep is called Back propagation which basically is used to minimize the loss. Here's the basic python code for a neural network with random inputs and two hidden layers.

4.1.4 MODULE 4 – CLASSIFICATION

Classification methods aim at identifying the category of a new observation among a set of categories on the basis of a labelled training set. Depending on the task, anatomical structure, tissue preparation, and features the classification accuracy varies.

4.2 ALGORITHMS

- CONVOLUTIOAL NEURAL NETWORK
- ALEXNET
- RESIDUAL NEURAL NETWORK (ResNet -18)

4.3 CONVOLUTIOAL NEURAL NETWORK

A convolutional neural network (CNN) is a form of artificial neural network that is specifically intended to process pixel input and is used in image recognition and processing.

CNNs are image processing, artificial intelligence (AI) systems that employ deep learning to do both generative and descriptive tasks, frequently using machine vision that includes image and video recognition, recommender systems, and natural language processing (NLP).

4.4 LAYERS IN CNN

1. Input Layer
2. Convolutional Layer
3. Pooling Layer
4. ReLu Layer
5. Batch Normalization Layer
6. Soft Max Layer
7. Fully Connected Layer

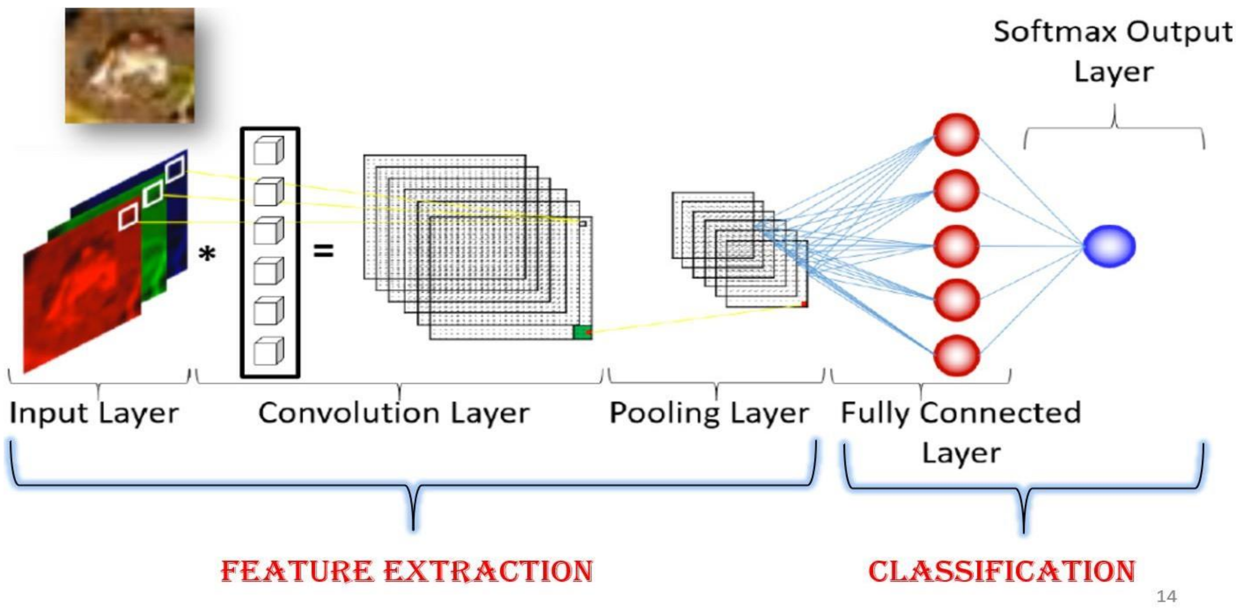


Fig 4.2 Layers of CNN

4.4.2 INPUT LAYER

Input layer in CNN should contain image data. The input layer brings the initial data into the system for further processing by subsequent layers. The input layer is the very beginning of the workflow.

4.4.3 CONVOLUTION LAYER

The first layer of a Convolutional Neural Network is always a Convolutional Layer**.** Convolution layer is also called feature extractor layer because features of the image are getting extracted within this layer. CNNs make use of filters (also known as kernels), to detect what features, such as edges, are present throughout an image. Then we slide the filter over the next receptive field of the same input image by a Stride and do the same operation again. We will repeat the same process again and again until we go through the whole image. The Convolution layer also contains ReLU activation to make all negative value to zero.

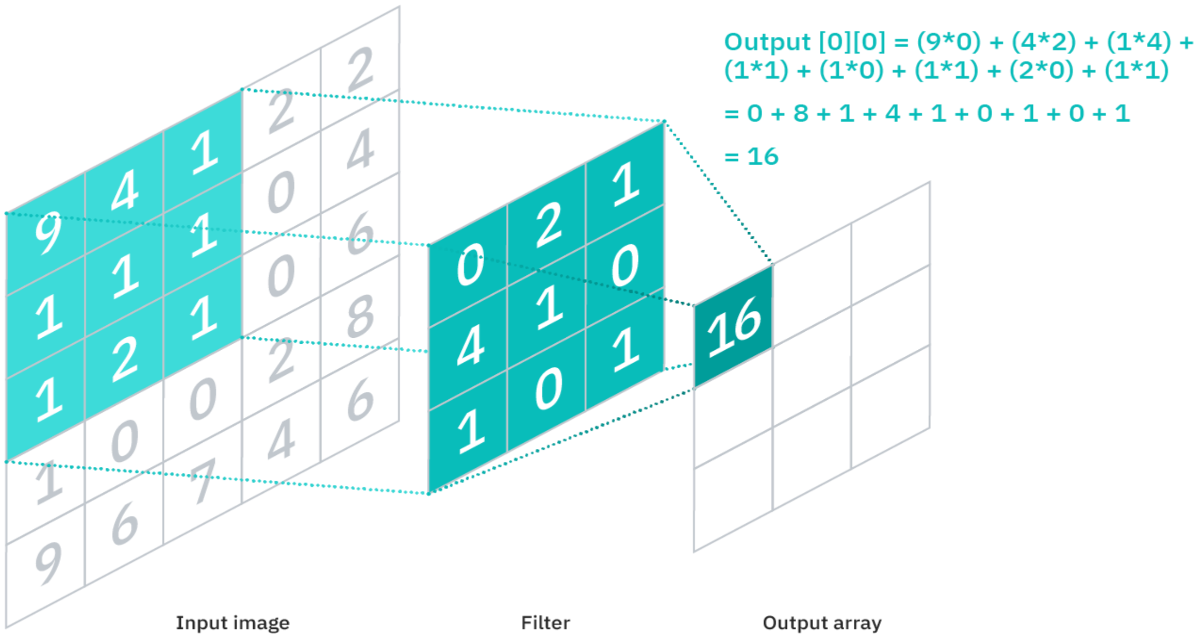


Fig 4.3 Calculation of Convolutional layer

FEATURE MAP CALCULATION:

Input image = 5*5

Filter/Kernel/Feature detector = 3*3

Output Feature map = (Size of input image - Size of kernel) +1

= (5-3) +1

= 2+1

= 3

Feature map /Output array = 3*3

STRIDE CALCULATION:

Stride = 2

Output stride = (Size of input image – Size of kernel / Stride) +1

= [(5-3)/2] +1

= (2/2) +1

= 1+1

= 2

Output Stride = 2*2

4.4.4 POOLING LAYER

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layers. We can observe the 4 x 4 dimension input is reduce to 2 x 2 dimension after max pooling is applied. There is no parameter in pooling layer but it has two hyper parameters — Filter(F) and Stride(S).

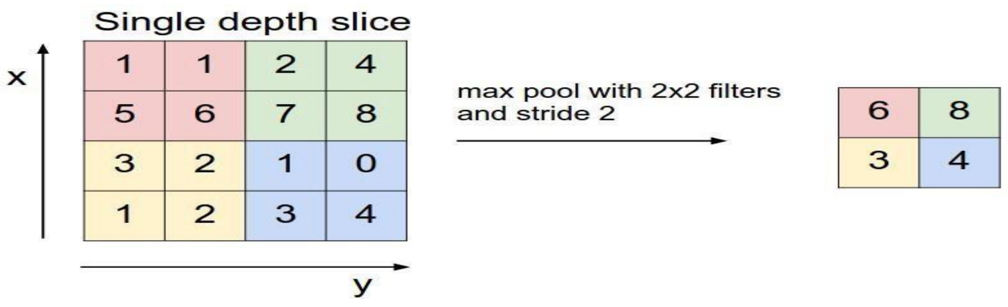


Fig 4.4 Pooling Layer in CNN

4.4.5 ReLU LAYER

Rectified Linear Unit (ReLU) this function simply returns 0 if your value is negative else it returns the same value you gave, nothing but eliminates negative outputs and maintains values between 0 to +infinity.

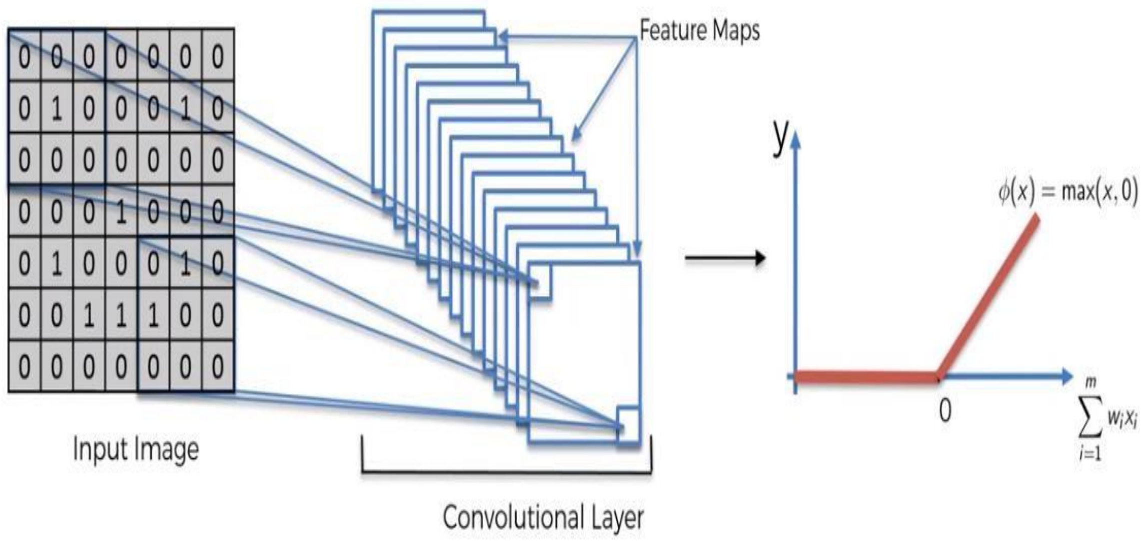


Fig4.5 ReLu Layer in CNN

4.4.6 BATCH NORMALIZATIO LAYER

Batch normalization is a technique to standardize the inputs to a network, applied to ether the activations of a prior layer or inputs directly. Batch normalization accelerates training, in some cases by halving the epochs or better, and provides some regularization, reducing generalization error.

4.4.7 FULLY CONNECTED LAYER

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different category by training.

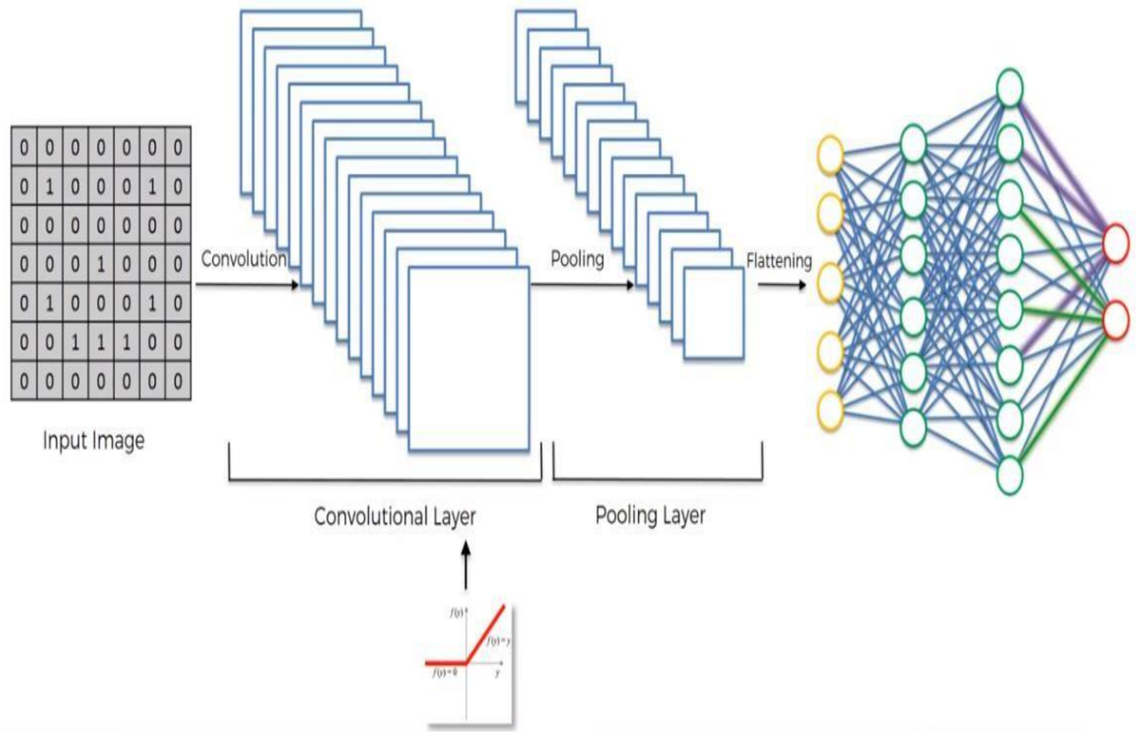


Fig4.6 Fully connected layer in CNN

4.4.8 SOFTMAX LAYER

Softmax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification. SoftMax function coverts real values into probabilities. It is only used as output layer of neural network. The higher probability is considered as actual output.

$$f_j(z) = \frac{e^{z}j}{\sum_k e^{z}k}$$

4.5 ALEXNET

Alex Net was the first convolutional network which used GPU to boost performance. Alex Net architecture consists of 5 convolutional layers, 3 max-pooling layers, 2 normalization layers, 2 fully connected layers, and 1 soft max layer.One thing to note here, since Alex net is **a deep architecture**, the authors introduced padding to prevent the size of the feature maps from reducing drastically. The input to this model is the images of size 227X227X3.The activation function used in all layers is ReLu. The activation function used in the output layer is SoftMax. Alex Net allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU. Not only does this mean that a bigger model can be trained, but it also cuts down on the training time. Overlapping Pooling.

4.5.2 BLOCK DIAGRAM OF ALEXNET

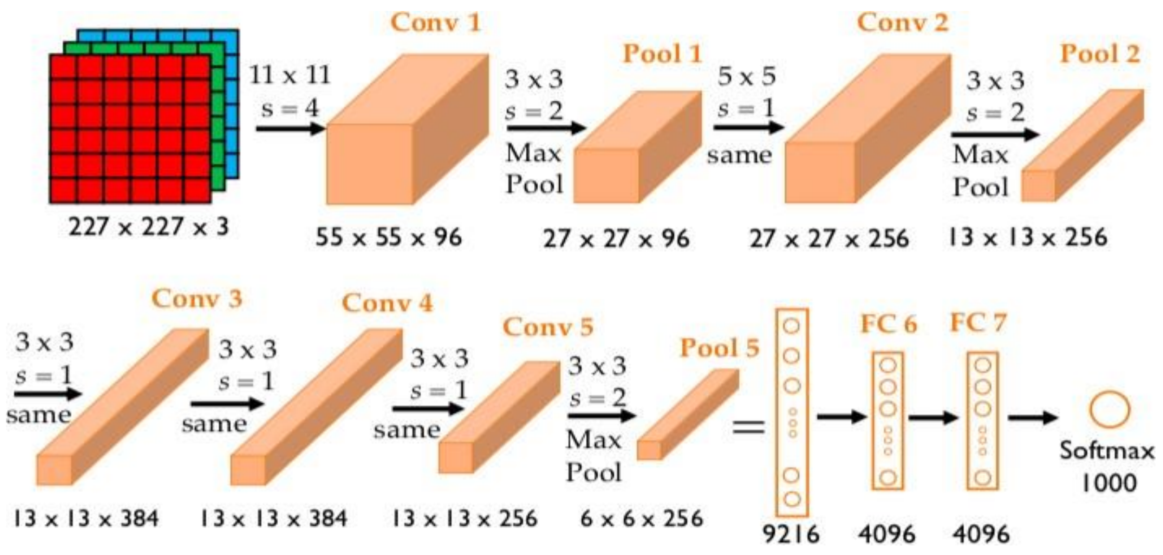


Fig4.7 Block Diagram of ALEXNET

Table 4.1 Convolution and Maxpooling layers in Alexnet

LAYER	#FILTERS/ NEURONS	FILTER SIZE	STRIDE	PADDING	SIZE OF FEATURE MAP	ACTIVATION FUNCTION
Input	-	-	-	-	227×227×3	-
Conv 1	96	11×11	4	-	56×56×96	ReLU
Max pool 1	-	3×3	2	-	27×27×96	-
Conv 2	256	5×5	1	2	27×17×256	ReLU
Max pool 2	-	3×3	2	-	13×13×256	-
Conv 3	384	3×3	1	1	13×13×384	ReLU
Conv 4	384	3×3	1	1	13×13×384	ReLU
Conv 5	256	3×3	1	1	13×13×256	ReLU
Max pool 3	-	3×3	2	-	6×6×256	-
Dropout 1	Rate=0.5	-	-	-	6×6×256	-

Table 4.2 Fully connected and dropout layers in Alexnet

LAYER	#FILTERS/ NEURONS	FILTER SIZE	STRIDE	PADDING	SIZE OF FEATURE MAP	ACTIVATION FUNCTION
-	-	-	-	-	-	-
-	-	-	-	-	-	-
-	-	-	-	-	-	-
Dropout 1	Rate=0.5	-	-	-	6×6×256	-
Fully Connected 1	-	-	-	-	4096	ReLU
Dropout 2	Rate=0.5	-	-	-	4096	-
Fully Connected 2	-	-	-	-	4096	ReLU
Fully Connected 3	-	-	-	-	1000	Softmax

4.6 RESIDUAL NEURAL NETWORK (ResNet -18)

Residual neural networks or commonly known as ResNets are the type of neural network that applies identity mapping. The input to some layer is passed directly or as a shortcut to some other layer. Skip connection is basically the identity mapping where the input from previous layer is added directly to the output of the other layer. There are 18 layers present in its architecture. It is very useful and efficient in image classification and can classify images into 1000 object categories. The network has an image input size of 224x224.

4.6.2 ResNet -18 ARCHITECTURE

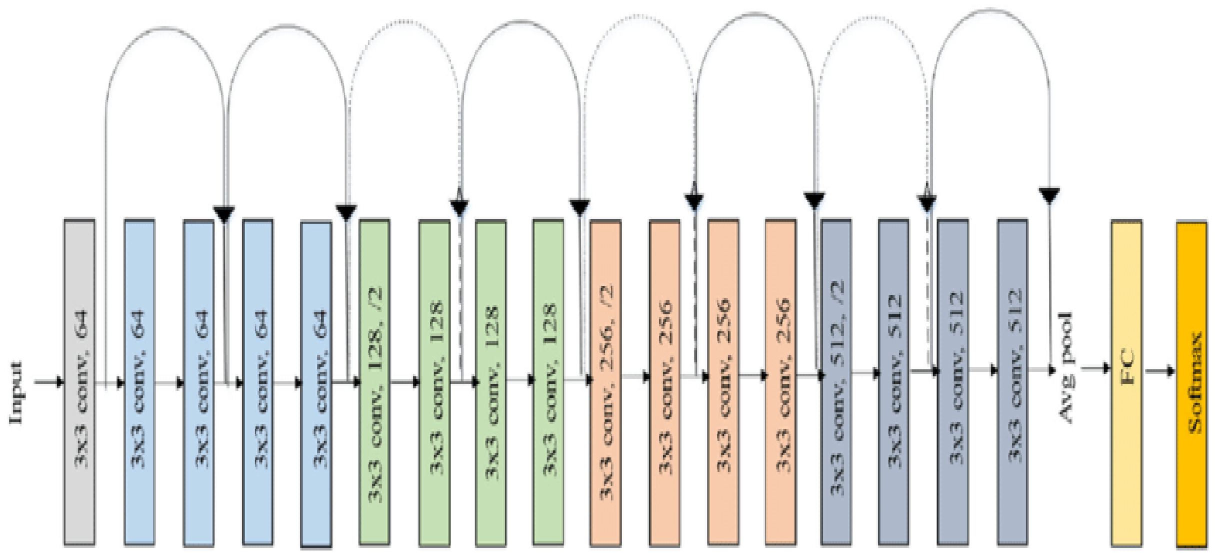
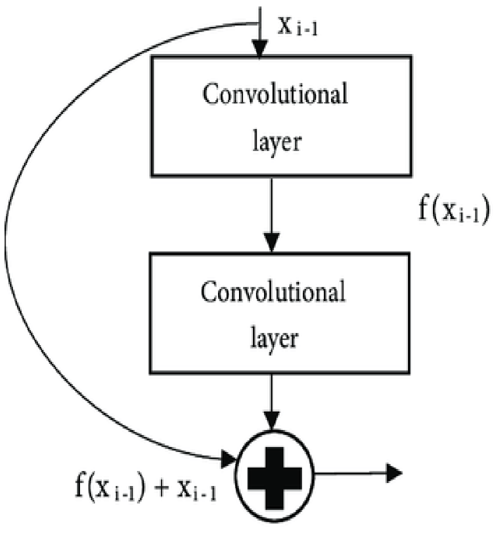
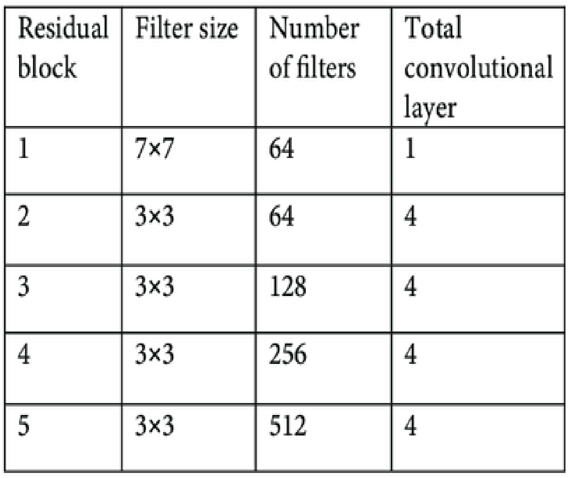


Fig 4.8 ResNet –18 Architecture

4.6.3 ResNet-18 BASIC BLOCK DIAGRAM AND FILTER INFORMATION



(a)



Residual block	Filter size	Number of filters	Total convolutional layer
1	7×7	64	1
2	3×3	64	4
3	3×3	128	4
4	3×3	256	4
5	3×3	512	4

(b)

Fig 4.9 ResNet -18 Basic Block Diagram and Filter Information

Table 4.3 ResNet-18 layer configuration architecture

LAYER NAME	OUTPUT SIZE	ResNet-18
Conv1	112×112×64	7×7,64,stride 2
Conv2_x	56×56×64	3×3maxpool,stride 2 $\begin{bmatrix} 3\times 3,64 \\ 3\times 3,64 \end{bmatrix} \times 2$
Conv3_x	28×28×128	$\begin{bmatrix} 3\times 3,128 \\ 3\times 3,128 \end{bmatrix} \times 2$
Conv4_x	14×14×256	$\begin{bmatrix} 3\times 3,256 \\ 3\times 3,256 \end{bmatrix} \times 2$
Conv5_x	7×7×512	$\begin{bmatrix} 3\times 3,256 \\ 3\times 3,256 \end{bmatrix} \times 2$
Average pool	1×1×512	$\begin{bmatrix} 3\times 3,512 \\ 3\times 3,512 \end{bmatrix} \times 2$
Fully connected	1000	7 ×7 average pool
Softmax	1000	512 ×1000 fully connections

4.7 CONFUSION MATRIX

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an **error matrix**.

The four metrics in the confusion matrix are

- **True Negative (TN):** Model has given prediction No, and the real or actual value was also No.
- **True Positive (TP):** The model has predicted yes, and the actual value was also true.
- **False Negative (FN):** The model has predicted no, but the actual value was Yes, it is also called as **Type-II error**.
- **False Positive (FP):** The model has predicted Yes, but the actual value was No. It is also called a **Type-I error.**

4.7.2 Need for Confusion Matrix in Machine learning

- It evaluates the performance of the classification models, when they make predictions on test data, and tells how good our classification model is.
- It not only tells the error made by the classifiers but also the type of errors such as it is either type-I or type-II error.
- With the help of the confusion matrix, we can calculate the different parameters for the model, such as accuracy, precision, etc.

4.7.3 Calculations using Confusion Matrix

Classification Accuracy: It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of predictions made by the classifiers.

The formula is given below:

Accuracy = (TP+TN)/(TP+TN+FP+FN)

Misclassification rate (Error rate): It is also termed as Error rate, and it defines how often the model gives the wrong predictions. The value of error rate can be calculated as the number of incorrect predictions to all number of the predictions made by the classifier. The formula is given below:

Error Rate = (FP+FN)/(TP+TN+FP+FN)

Precision: It can be defined as the number of correct outputs provided bythe model or out of all positive classes that have predicted correctly by themodel, how many of them were actually true. It can be calculated using the below formula:

Precision = TP/(TP+FP)

Recall: It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

Recall = (TP)/(TP+FN)

F-measure: If two models have low precision and high recall or vice versa,it is difficult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision.It can be calculated using the below formula:

F-Measure = 2* (Recall*Precision)/(Recall+Precision)

4.7.4 ALEXNET F-MEASURE (F1 SCORE) CALCULATION

F-MEASURE (F1SCORE) =2*(Recall*Precision)/(Recall+Precision)

TEST CASE	**CALCULATION**
EFFUSION	Recall=92.3% Precision=80% F1 Score=2*(92.3*80)/ (92.3+80) =2*(7384)/ (172.3) =2*42.85 =85.7%
NORMAL	Recall=85.7% Precision=80% F1 Score=2*(85.7*80)/ (85.7+80) =2*(6856)/ (165.7) =2*41.37 =84.5%
TUBE	Recall=76.9% Precision=100% F1 Score=2*(76.9*100)/ (76.9+100) =2*(7690)/ (176.9) =2*43.47 =87%

F-MEASURE (F1SCORE) = (85.7+84.5+87)/3**= 85.7%**

4.8 EVALUATION PARAMETERS

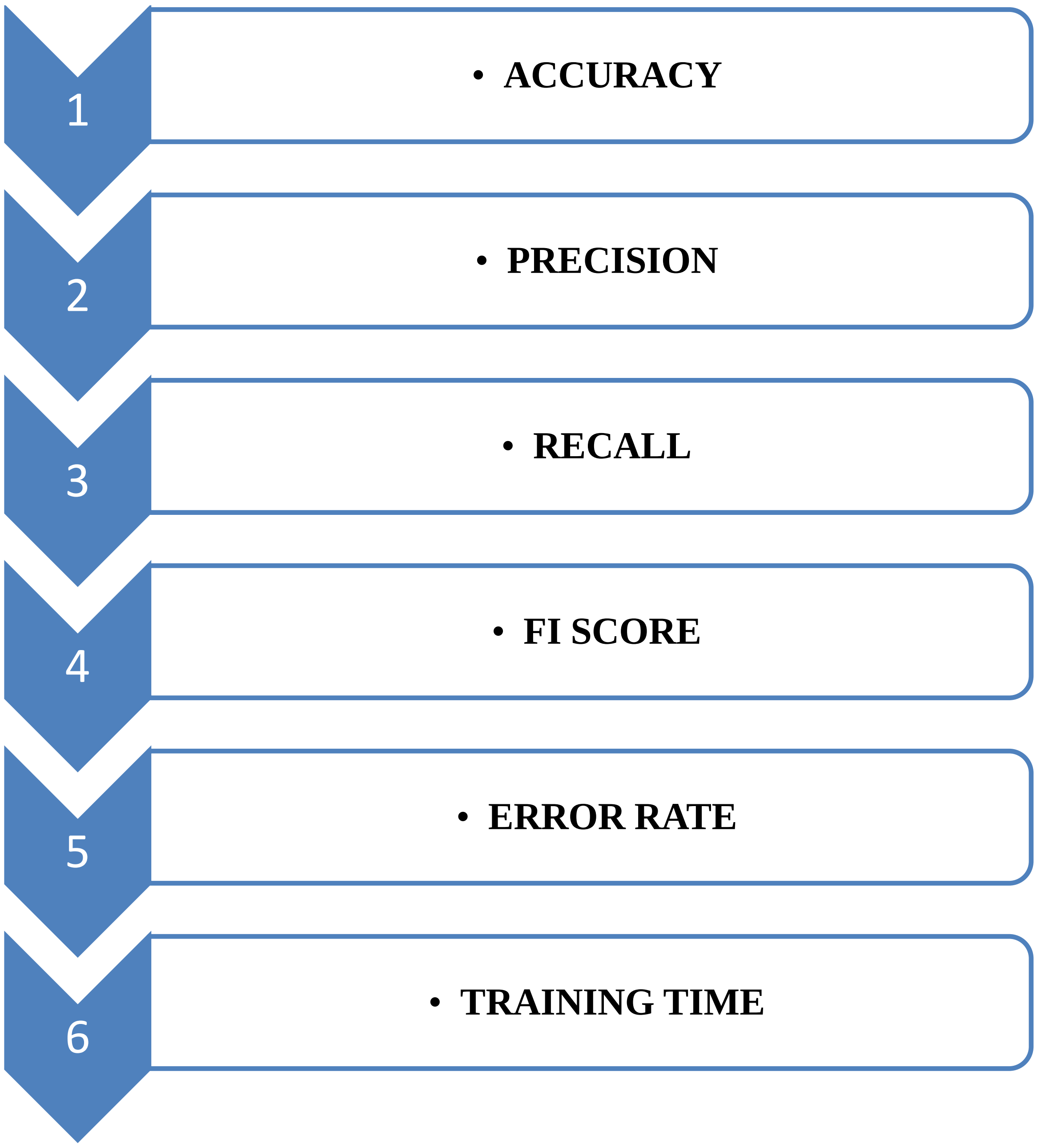


Fig 4.10 Evaluation Parameters

CHAPTER 5
SOFTWARE SPECIFICATIONS

5.1SOFTWARE REQUIREMENTS

- MATLAB 8.6 Version R2018a

5.1.1 MATLAB

MATLAB (**MAT**rix **LAB**oratory) is a numerical computing environment and fourth-generation programming language. Developed by Math Works, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user_interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

5.1.2 FEATURES OF MATLAB

- High-level language for technical computing.
- Development environment for managing code, files, and data.
- Interactive tools for iterative exploration, design, and problem solving.
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration.
- 2-D and 3-D graphics functions for visualizing data.
- Tools for building custom graphical user interfaces.
- Functions for integrating MATLAB based algorithms with external applications and languages, such as COM, and Microsoft Excel.

5.2 MATLAB 18a HIGHLIGHTS

- Expanded support for the Image Labeler app
- Improvements to the neural network training process.

- Updated pre-processing capabilities for tables and timetables
- The new Predictive Maintenance Toolbox™, which helps you get started with predictive maintenance, monitor machine health, estimate time to failure, and more
- More functions that natively support tall arrays so can work with big data
- The new Vehicle Dynamics Blockset, enabling you to evaluate a vehicle’s ride and handling characteristics, among other capabilities
- New support for designing automated driving systems, including the new Driving Scenario Designer
- Enhancements to vehicle connectively with CAN FD and XCP-over-Ethernet support

5.3 NEW FEATURES IN MATLAB 18a

- Simulation Analysis and Performance
- Simulink Editor
- Component -Based Modelling
- Project and File Management
- Data Management
- Block Enhancements
- Connection to Hardware
- MATLAB Function Blocks
- S-Functions

5.4 ADAVANTAGES OF MATLAB OVER OTHER SOFTWARES

- A very large (and growing) database of built-in algorithms for image processing and computer vision applications
- MATLAB allows you to test algorithms immediately without recompilation. You can type something at the command line or execute a section in the editor and immediately see the results, greatly facilitating algorithm development.

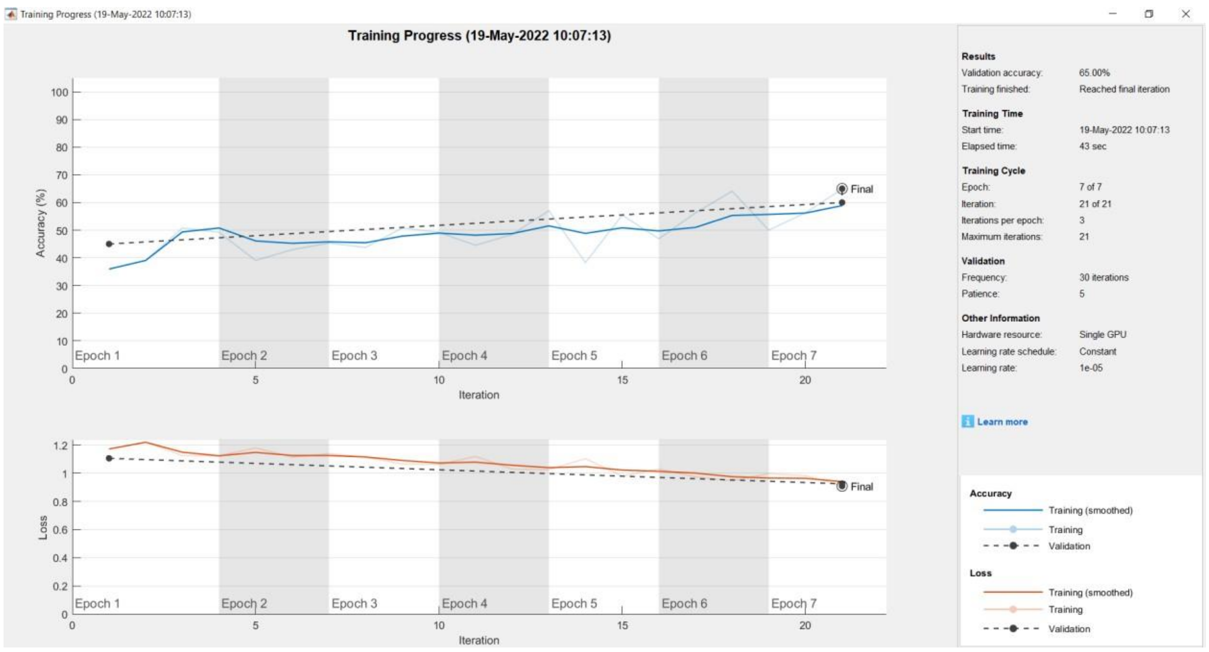
- The MATLAB Desktop environment, which allows you to work interactively with your data, helps you to keep track of files and variables, and simplifies common programming/debugging tasks
- The ability to read in a wide variety of both common and domain-specific image formats.
- The ability to call external libraries, such as OpenCV
- Clearly written documentation with many examples, as well as online resources such as web seminars ("webinars").
- Bi-annual updates with new algorithms, features, and performance enhancements
- If you are already using MATLAB for other purposes, such as simulation, optimation, statistics, or data analysis, then there is a very quick learning curve for using it in image processing.
- The ability to process both still images and video.
- Technical support from a well-staffed, professional organization (assuming your maintenance is up-to-date)
- A large user community with lots of free code and knowledge sharing
- The ability to auto-generate C code, using MATLAB Coder, for a large (and growing) subset of image processing and mathematical functions, which you could then use in other environments, such as embedded systems or as a component in other software.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 CNN RESULT

6.1.1 CNN TRAINING PROCESS



Results
Validation accuracy: 65.00%
Training finished: Reached final iteration

Training Time
Start time: 19-May-2022 10:07:13
Elapsed time: 43 sec

Training Cycle
Epoch: 7 of 7
Iteration: 21 of 21
Iterations per epoch: 3
Maximum iterations: 21

Validation
Frequency: 30 iterations
Patience: 5

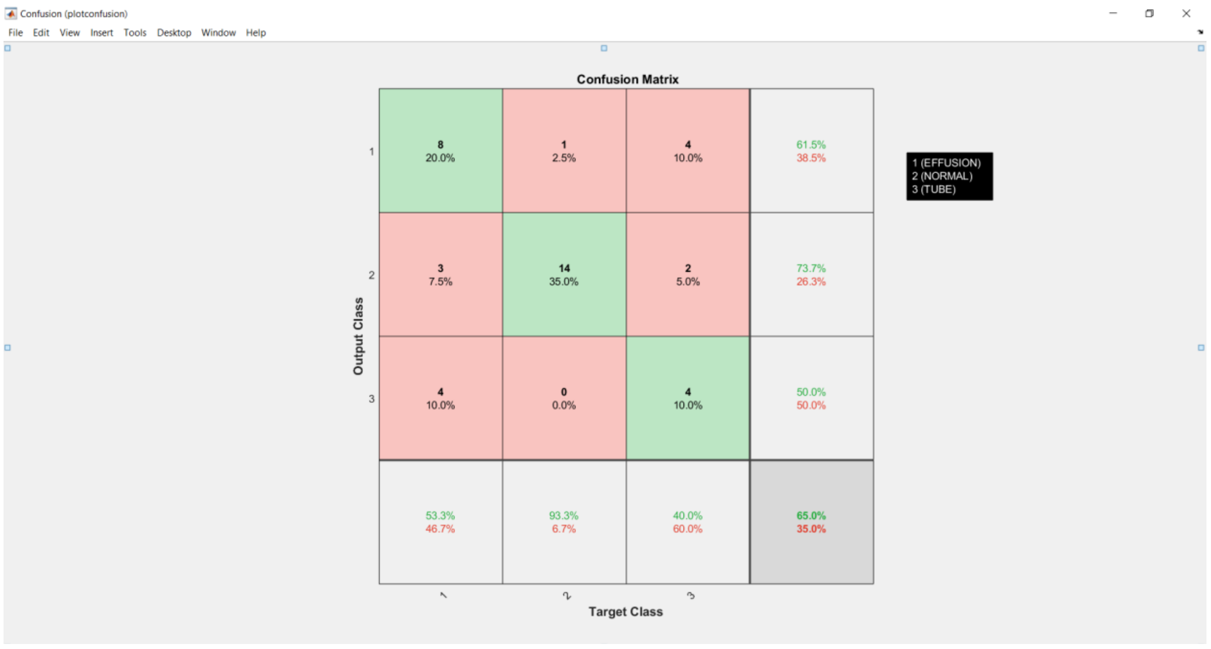
Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 1e-05

Accuracy
Training (smoothed)
Training
Validation

Loss
Training (smoothed)
Training
Validation

Fig 6.1 CNN training process

6.1.2 CONFUSION MATRIX FOR CNN



	1	2	3	
1	8 20.0%	1 2.5%	4 10.0%	61.5% 38.5%
2	3 7.5%	14 35.0%	2 5.0%	73.7% 26.3%
3	4 10.0%	0 0.0%	4 10.0%	50.0% 50.0%
	53.3% 46.7%	93.3% 6.7%	40.0% 60.0%	65.0% 35.0%

Fig 6.2 Confusion matrix for CNN

6.1.3 CNN TESTING PROCESS

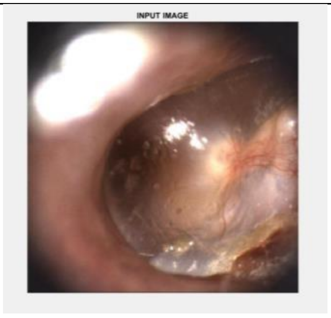
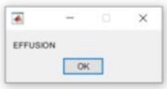
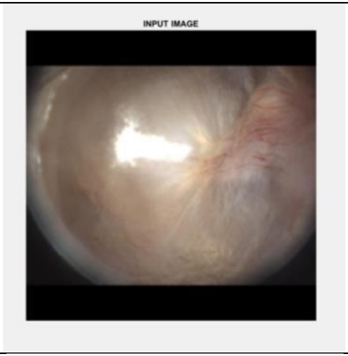
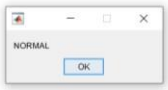
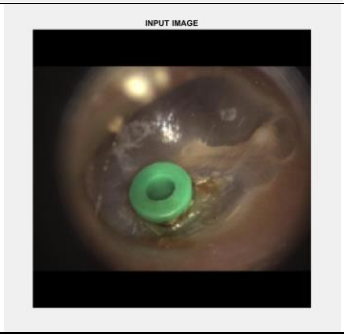
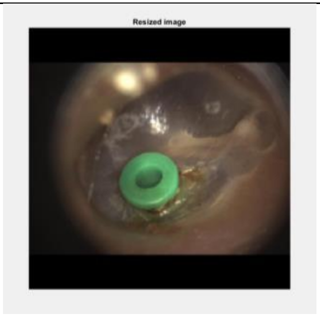
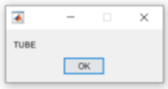
INPUT IMAGE	RESIZED IMAGE	OUTPUT
		 EFFUSION **EFFUSION**
		 NORMAL **NORMAL**
		 TUBE **TUBE**

Fig 6.3 CNN testing process

6.1.4 PREDICTION AND VALIDATION DATA ANALYSIS OF CNN ALGORITHM

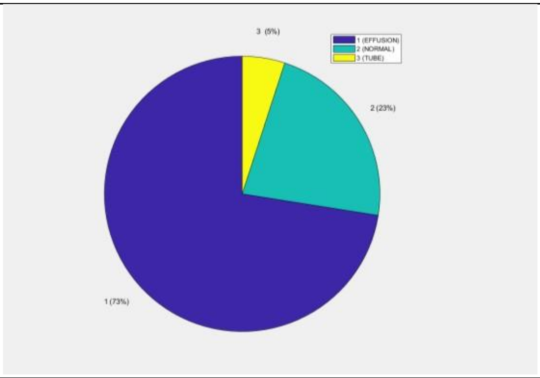
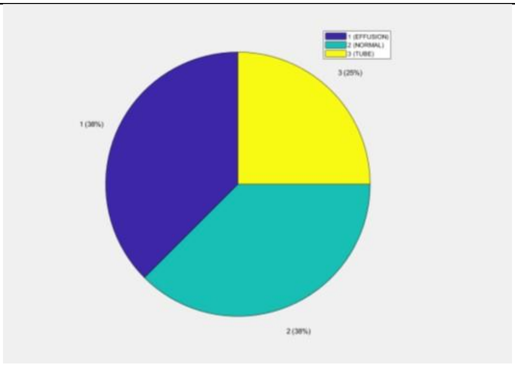
PREDICTION	VALIDATION
	

Fig 6.4 Prediction and validation data analysis of CNN algorithm

Table 6.1 Metrics in confusion matrix of CNN

METRICS IN CONFUSION MATRIX	EFFUSION	NORMAL	TUBE
True positive	20%	35%	10%
False positive	17.5%	2.5%	15%
False negative	12.5%	12.5%	10%
True negative	22.5%	22.5%	25%

Table 6.2 Calculations from confusion matrix

ACCURACY		PRECISION		RECALL		F-MEASURE	
Effusion	58.6%	Effusion	53.3%	Effusion	61.5%	Effusion	57.1%
Normal	79.3%	Normal	93.3%	Normal	73.7%	Normal	82.3%
Tube	59.1%	Tube	40%	Tube	50%	Tube	44.4%
Accuracy=65.6%		**Precision=62.2%**		**Recall=61.8%**		**F-measure=61.3%**	

6.1.5 ANALYSIS OF CNN ALGORITHM

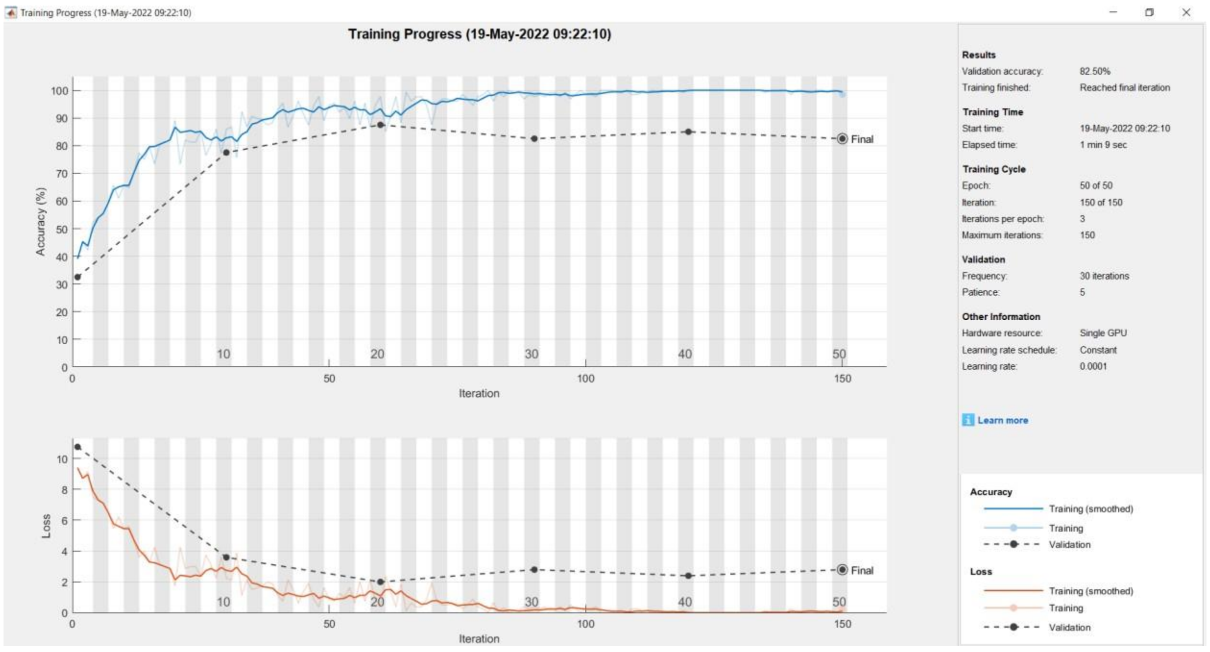
Table 6.3 Analysis of CNN Algorithm

PARAMETER	VALUE
Accuracy	65.6%
Precision	62.2%
Recall	61.8%
F-measure	61.3%
Elapsed time	43 seconds
Validation frequency	30 iterations
Learning rate	$1e^{-05}$
Error rate	34.5%

- ✓ CNN Algorithm produces an accuracy ranging from 45%-70%.
- ✓ The F1 score produced by CNN Algorithm is 0.61.
- ✓ Since CNN produces less accuracy and F1 score with a greater error rate of almost 34%, we consider another algorithm under CNN which is the ResNet -18 Algorithm.

6.2 RESNET -18 RESULT

6.2.1 RESNET -18 TRAINING PROCESS



Results
Validation accuracy: 82.50%
Training finished: Reached final iteration

Training Time
Start time: 19-May-2022 09:22:10
Elapsed time: 1 min 9 sec

Training Cycle
Epoch: 50 of 50
Iteration: 150 of 150
Iterations per epoch: 3
Maximum iterations: 150

Validation
Frequency: 30 iterations
Patience: 5

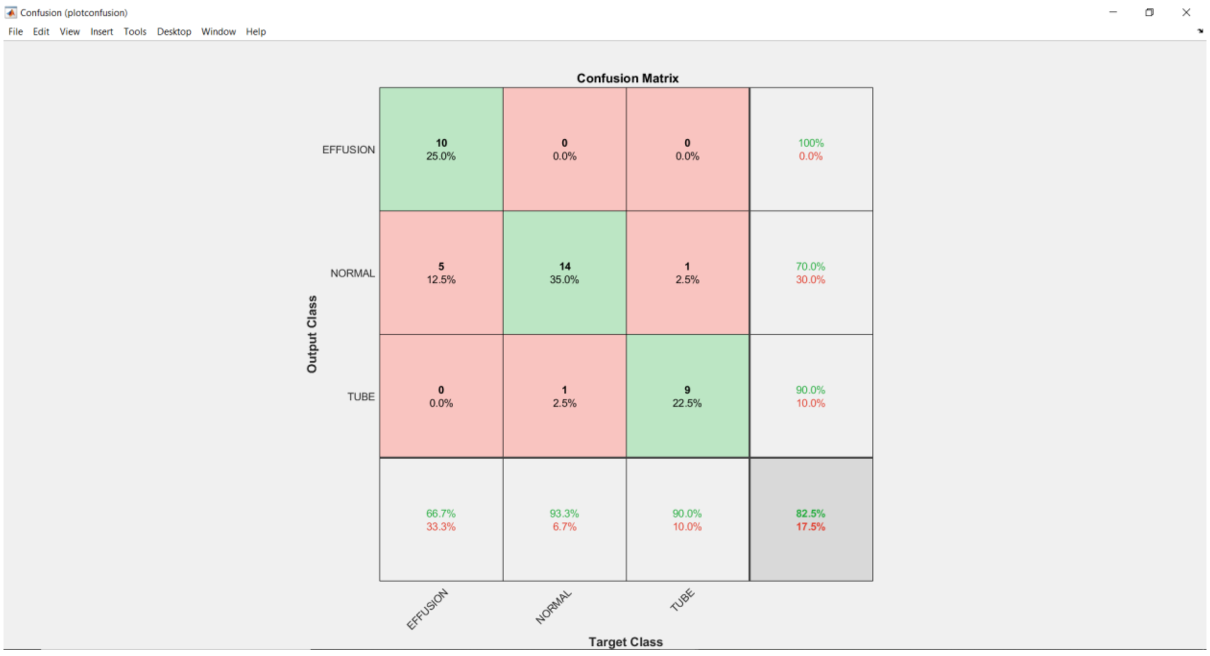
Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.0001

Accuracy
Training (smoothed)
Training
Validation

Loss
Training (smoothed)
Training
Validation

Fig 6.5 RESNET -18 training process

6.2.2 CONFUSION MATRIX FOR RESNET-18



Confusion Matrix

Output Class \ Target Class	EFFUSION	NORMAL	TUBE	
EFFUSION	10 25.0%	0 0.0%	0 0.0%	100% 0.0%
NORMAL	5 12.5%	14 35.0%	1 2.5%	70.0% 30.0%
TUBE	0 0.0%	1 2.5%	9 22.5%	90.0% 10.0%
	66.7% 33.3%	93.3% 6.7%	90.0% 10.0%	82.5% 17.5%

Fig 6.6 Confusion matrix for RESNET -18

6.2.3 RESNET -18 TESTING PROCESS

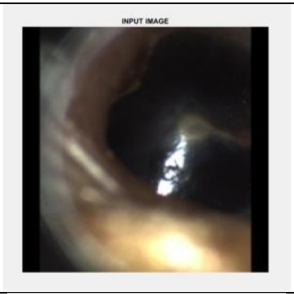
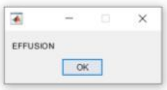
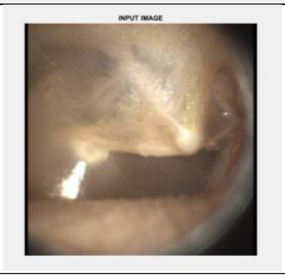
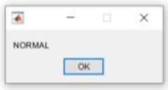
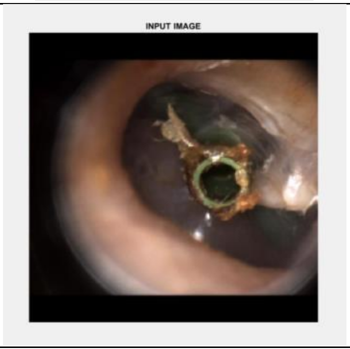
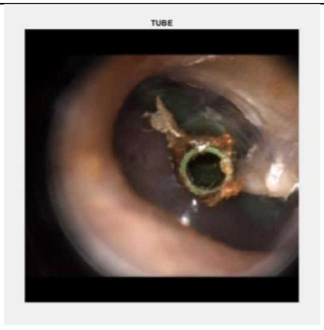
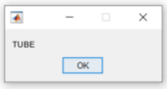
INPUT IMAGE	RESIZED IMAGE	OUTPUT
		 EFFUSION **EFFUSION**
		 NORMAL **NORMAL**
		 TUBE **TUBE**

Fig 6.7 RESNET -18 testing process

6.2.4 PREDICTION AND VALIDATION DATA ANALYSIS OF RESNET 18 ALGORITHM

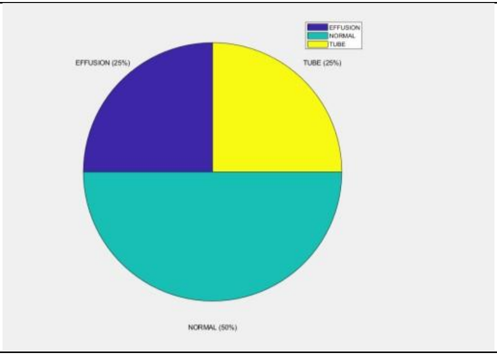
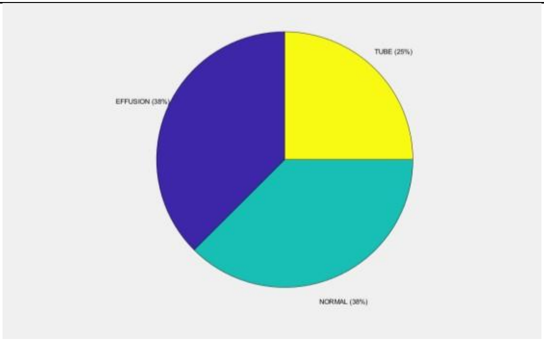
PREDICTION	VALIDATION
	

Fig 6.8 Prediction and validation data analysis of RESNET -18 algorithm

Table 6.4 Metrics in confusion matrix of RESNET -18

METRICS IN CONFUSION MATRIX	EFFUSION	NORMAL	TUBE
True positive	25%	35%	22.5%
False positive	12.5%	2.5%	2.5%
False negative	0%	15%	2.5%
True negative	17.5%	2.5%	15%

Table 6.5 Calculations from confusion matrix

ACCURACY		PRECISION		RECALL		F-MEASURE	
Effusion	77.2%	Effusion	66.7%	Effusion	100%	Effusion	80%
Normal	68.1%	Normal	93.3%	Normal	70%	Normal	80%
Tube	88.2%	Tube	90%	Tube	90%	Tube	90%
Accuracy=81%		**Precision=83.3%**		**Recall=86.7%**		**F-measure=83.3%**	

6.2.5 ANALYSIS OF RESNET-18 ALGORITHM

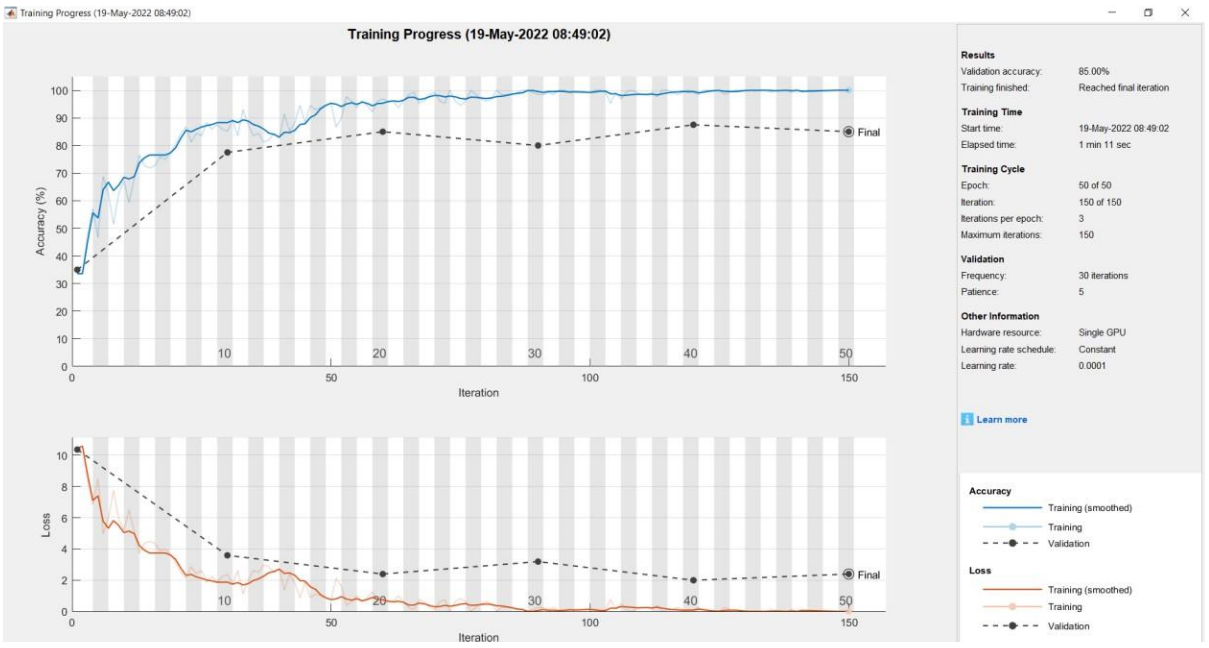
Table 6.4 Analysis of ResNet -18 Algorithm

PARAMETER	VALUE
Accuracy	81%
Precision	83.3%
Recall	86.7%
F-measure	83.3%
Elapsed time	1minute 9seconds
Validation frequency	30 iterations
Learning rate	0.0001
Error rate	17.5%

- ✓ ResNet -18 Algorithm produces an accuracy ranging from 75%-93%.
- ✓ The F1 score produced by ResNet -18 Algorithm is 0.83.
- ✓ ResNet -18 produces better accuracy and F1 score with a reduced error rate of almost 18% than the previous basic CNN Algorithm.
- ✓ But ResNet -18 Algorithm takes more training time, hence another algorithm is considered which is the AlexNet Algorithm.

6.3 ALEXNET RESULT

6.3.1 ALEXNET TRAINING PROCESS



Training Progress (19-May-2022 08:49:02)
Training Progress (19-May-2022 08:49:02)
Accuracy (%)
Final
Iteration
Loss
Final
Iteration
Learn more

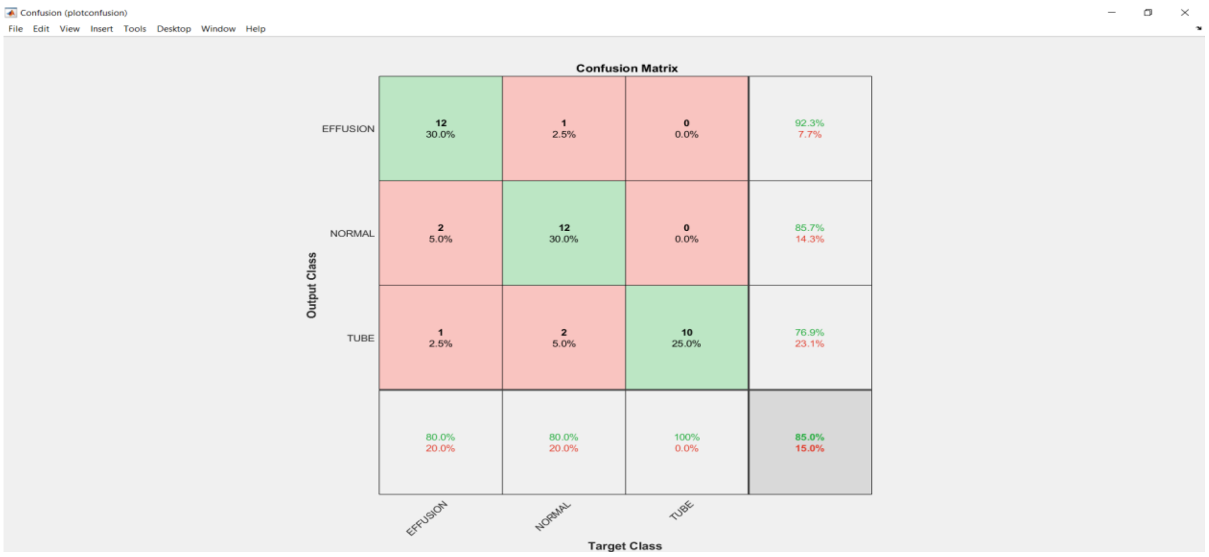
Results	
Validation accuracy:	85.00%
Training finished:	Reached final iteration
Training Time	
Start time:	19-May-2022 08:49:02
Elapsed time:	1 min 11 sec
Training Cycle	
Epoch:	50 of 50
Iteration:	150 of 150
Iterations per epoch:	3
Maximum iterations:	150
Validation	
Frequency:	30 iterations
Patience:	5
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Accuracy
- Training (smoothed)
- Training
- Validation

Loss
- Training (smoothed)
- Training
- Validation

Fig 6.9 ALEXNET training process

6.3.2 CONFUSION MATRIX FOR ALEXNET



	EFFUSION	NORMAL	TUBE	
EFFUSION	12 30.0%	1 2.5%	0 0.0%	92.3% 7.7%
NORMAL	2 5.0%	12 30.0%	0 0.0%	85.7% 14.3%
TUBE	1 2.5%	2 5.0%	10 25.0%	76.9% 23.1%
	80.0% 20.0%	80.0% 20.0%	100% 0.0%	85.0% 15.0%

Fig 6.10 Confusion matrix for ALEXNET

6.3.3 ALEXNET TESTING PROCESS

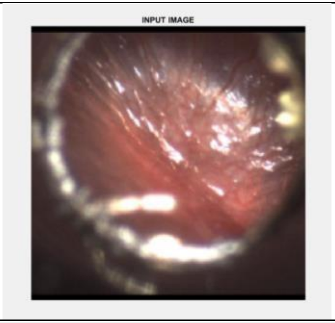
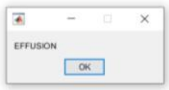
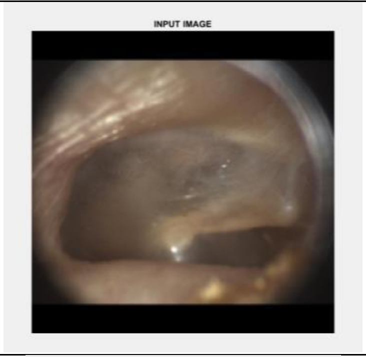
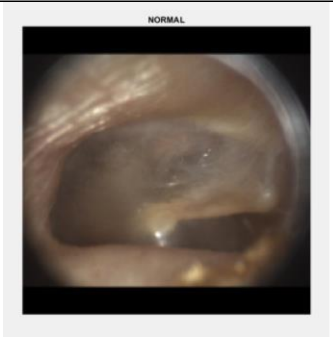
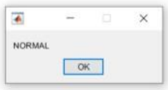
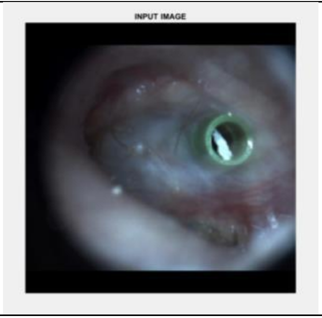
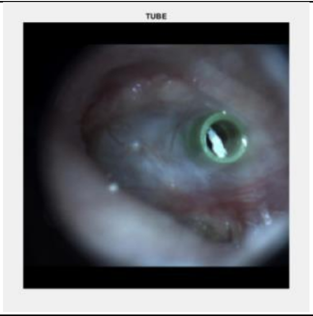
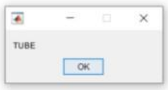
INPUT IMAGE	RESIZED IMAGE	OUTPUT
		 EFFUSION **EFFUSION**
		 NORMAL **NORMAL**
		 TUBE **TUBE**

Fig 6.11 ALEXNET testing process

6.3.4 PREDICTION AND VALIDATION DATA ANALYSIS OF ALEXNET ALGORITHM

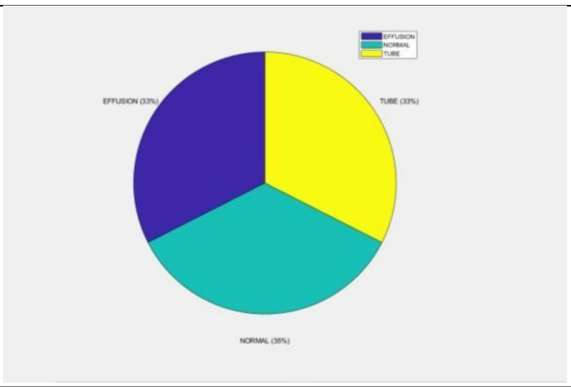
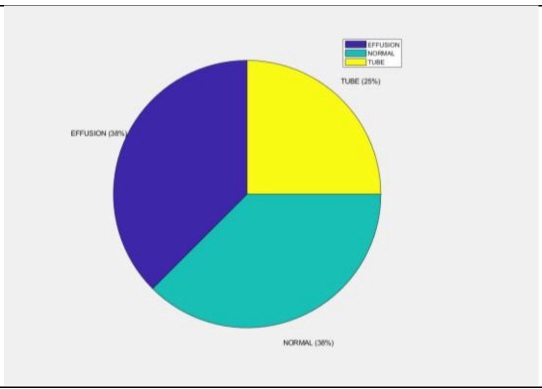
PREDICTION	VALIDATION
	

Fig 6.12 Prediction and validation data analysis of ALEXNET algorithm

Table 6.1 Metrics in confusion matrix of ALEXNET

METRICS IN CONFUSION MATRIX	EFFUSION	NORMAL	TUBE
True positive	30%	30%	25%
False positive	7.5%	7.5%	0%
False negative	2.5%	5%	7.5%
True negative	12.5%	10%	7.5%

Table 6.2 Calculations from confusion matrix

ACCURACY		PRECISION		RECALL		F-MEASURE	
Effusion	81%	Effusion	80%	Effusion	92.3%	Effusion	85.7%
Normal	76.2%	Normal	80%	Normal	85.7%	Normal	84.5%
Tube	81.2%	Tube	100%	Tube	76.9%	Tube	87%
Accuracy=84.9%		**Precision=86.6%**		**Recall=85%**		**F-measure=85.7%**	

6.3.5 ANALYSIS OF ALEXNET ALGORITHM

Table 6.3 Analysis of ALEXNET Algorithm

PARAMETER	VALUE
Accuracy	84.9%
Precision	86.6%
Recall	85%
F-measure	85.7%
Elapsed time	1 minute 11 seconds
Validation frequency	30 iterations
Learning rate	0.0001
Error rate	15%

- ✓ AlexNet Algorithm produces an accuracy ranging from 75%-90%.
- ✓ The F1 score produced by AlexNet Algorithm is 0.85.
- ✓ Although the accuracy range for AlexNet Algorithm is little less than the ResNet -18 Algorithm, AlexNet Algorithm takes less training time than ResNet -18 Algorithm.
- ✓ AlexNet produces a F1 score of 0.85 and very low error rate of 15% than the other two algorithms.
- ✓ Hence AlexNet Algorithm is chosen as the better algorithm for our proposed system.

6.4 ANALYSIS OF CNN, RESNET-18, ALEXNET

- We have tested three architectures: one classic CNN, ResNet-18 and AlexNet. Ignoring the large fluctuations in testing accuracy, however, AlexNet appears to have performed better than the other two. Further examination of each architecture and its constraints are necessary to make conclusions about the superiority of a network.
- Training a ResNet-18 requires a lot of computations (about 10 times more than that of AlexNet) which means more training time and energy required.
- Training an AlexNet takes about the same time as training ResNet -18. The memory requirements are 10 times less with improved accuracy (about 9%).
- AlexNet is a deeper architecture with 8 layers which means that is better able to extract features when compared to LeNet and ResNet. It works well for the time with color images.
- The ReLu activation function used in this network has 2 advantages.
    - (i) It does not limit the output unlike other activation functions. This means there isn't too much loss of features.
    - (ii) using the ReLU function over other activation functions is that it does not activate all the neurons at the same time.
- AlexNet improves model training speed since not all perceptron are active.

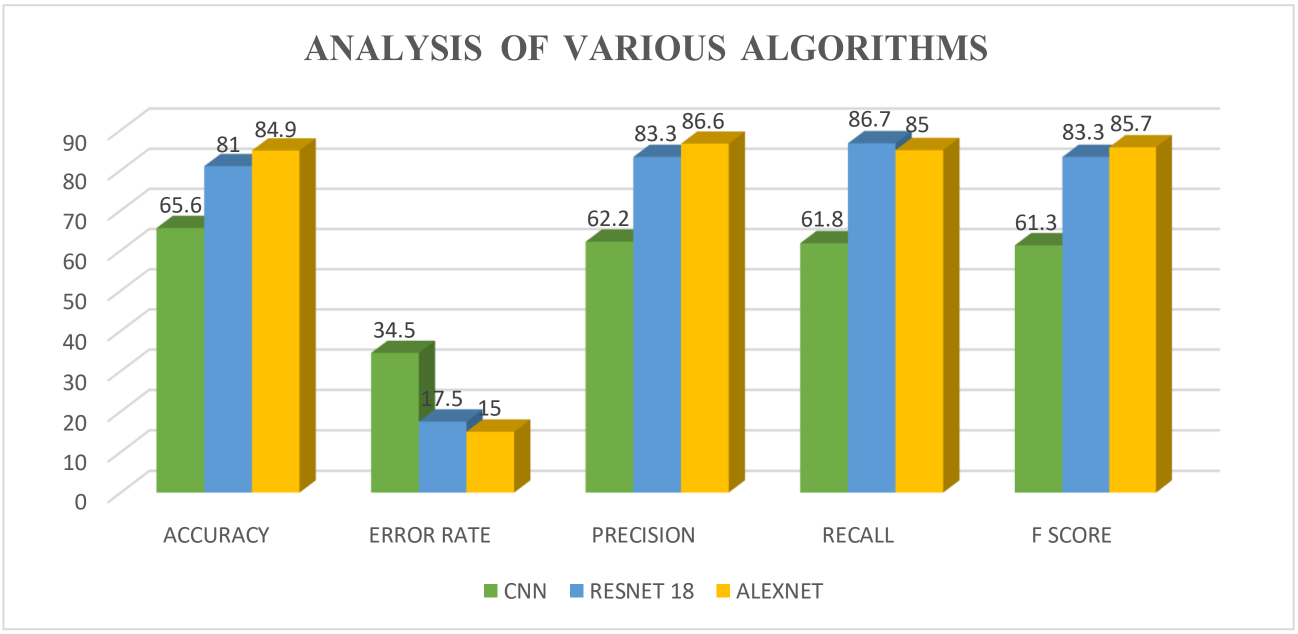


Fig 6.13 Analysis of various algorithm

6.4.1 Accuracy

Pro: Easy to interpret. If we say that a model is 90% accurate, we know that it correctly classified 90% of observations.

Con: Does not take into account how the data is distributed. For example, suppose 90% of all players do not get drafted into the NBA. If we have a model that simply predicts every player to not get drafted, the model would correctly predict the outcome for 90% of the players. This value seems high, but the model is actually unable to correctly predict any player who gets drafted.

6.4.2 F1 Score

Pro: Takes into account how the data is distributed. For example, if the data is highly imbalanced (e.g., 90% of all players do not get drafted and 10% do get drafted) then F1 score will provide a better assessment of model performance.

Con: Harder to interpret. The F1 score is a blend of the precision and recall of themodel, which makes it a bit harder to interpret.

6.5 ANALYSIS OF ACCURACY VS F1 SCORE

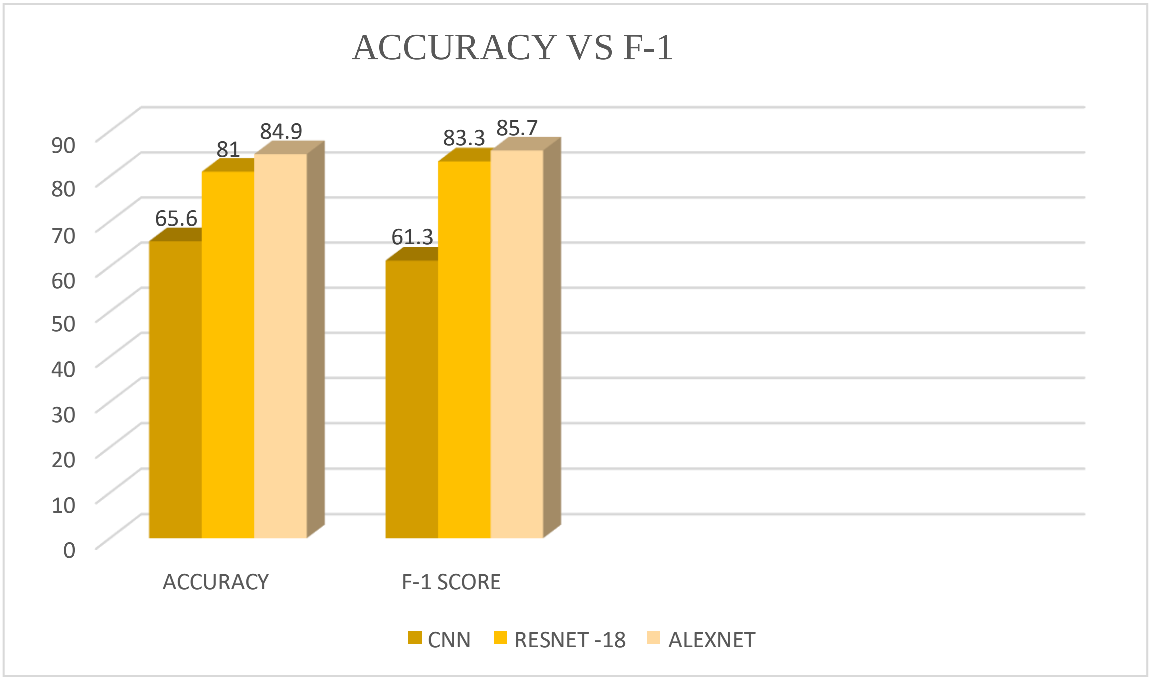


Fig 6.14 Accuracy vs F-1 score

Since our proposed system is a real-life classification problem, F1- score can be considered as the best metric for evaluation. From the above graph we shall conclude that both F-1 score and accuracyfor ALEXNET algorithm is the best when compared to other two algorithms like basic CNN and ResNet -18. Hence ALEXNET algorithm stands best for our proposed system.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

Accuracy is used when the True Positives and True negatives are more important while F1-score is used when the False Negatives and False Positives are crucial. Accuracy can be used when the class distribution is similar while F1-scoreis a better metric when there are imbalanced classes as in our proposed system. In most real-life classification problems, imbalanced class distribution exists and thus F1-score is a better metric to evaluate a system. Since our proposed system is a real-life classification problem, F1-score can be considered as the best metric for evaluation.

We shall conclude that both F-1 score and accuracyfor ALEXNET algorithm is the best when compared to other two algorithms like basic CNN and ResNet -18. Hence ALEXNET algorithm stands best for our proposed system.

7.1.1 APPLICATION OF PROPOSED SYSTEM

- To support early detection.

7.2 FUTURE ENHANCEMENTS

In future, with more time and with more comprehensive architecture models researches can be done in regard to the proposed system which could make the classification more accurate. Furthermore, a website or an app for doctors and nurses might be established for a more user-friendly experience

APPENDIX

SOURCE CODE FOR CNN ARCHITECTURE TRAINING PROCESS

```
%%%%%%%%%%%% EAR DRUM DISEASE DETECTION
%%%%%%%%%%%%%%%

clc; clear;

close all;

warning off



% %%% TRAIN THE DATASET IMAGES %%%%%
%
% matlabroot ='F:\IMAGE PROCESSING\EAR DRUM';
% data1 = fullfile(matlabroot,'TRAINING IMAGES');
%
Data=imageDatastore(data1,'IncludeSubfolders',true,'Label Source','foldernames');
%
% validationPath = fullfile(matlabroot,'TESTING IMAGES');
% imdsValidation = imageDatastore(validationPath, ...
% 'IncludeSubfolders',true,'LabelSource','foldernames');
%
% %%%%%%%CREATE CONVOLUTIONAL NEURAL NETWORK LAYERS
%%%%%%
%
%
%   layers=[imageInputLayer([227 227 3])
%
%          convolution2dLayer(3,8,'Padding','same')
%          batchNormalizationLayer
%          reluLayer
%
%          maxPooling2dLayer(2,'Stride',2)
%
%          convolution2dLayer(3,16,'Padding','same')
%          batchNormalizationLayer
%          reluLayer
%
%          maxPooling2dLayer(2,'Stride',2)
%
%          convolution2dLayer(3,32,'Padding','same')
%          batchNormalizationLayer
%          reluLayer
%
%          maxPooling2dLayer(2,'Stride',2)
%
%          convolution2dLayer(3,64,'Padding','same')
%          batchNormalizationLayer
%          reluLayer
%
%          maxPooling2dLayer(2,'Stride',2)
%
%          convolution2dLayer(3,128,'Padding','same')
%          batchNormalizationLayer
%          reluLayer
%
%          maxPooling2dLayer(2,'Stride',2)
%
%          convolution2dLayer(3,256,'Padding','same')
%          batchNormalizationLayer
%          reluLayer
%
%          maxPooling2dLayer(2,'Stride',2)
%
%          fullyConnectedLayer(3)
%          softmaxLayer
%          classificationLayer];
%
%
options=trainingOptions('sgdm','MaxEpochs',7,'InitialLear
nRate',0.00001,'Shuffle','every-epoch', ...
%          'ValidationData',imdsValidation, ...
%          'ValidationFrequency',30,...
%          'Verbose',false, ...
%          'Plots','training-progress');
%
%   convnet=trainNetwork(Data,layers,options);
%
%   save convnet.mat convnet
%
%
%
% % CLASSIFY VALIDATION IMAGES AND COMPUTE ACCURACY % % %
% %
```

```
%
% YPred = classify(convnet,imdsValidation);
%
% YValidation = imdsValidation.Labels;
%
% accuracy = sum(YPred ==
YValidation)/numel(YValidation);
```

SOURCE CODE FOR CNN ARCHITECTURE TESTING PROCESS

```
% %   %%%%%%%%%%READ THE IMAGE FROM THE DATASET
%%%%%%%%%%%%
% % % % % % %
  load convnet.mat
[filename,pathname]=uigetfile('*.*');

im1=imread([pathname,filename]); figure,imshow(im1),title('INPUT IMAGE');
%
%
% %%%%%%% RESIZE THE IMAGE %%%%%%%%%%
%
%
im=imresize(im1,[227 227]); figure,imshow(im),title('Resized image');

% % % %%%%%%%%%%%% CONVERT THE DATA TYPE INTO
UNSIGNEDINTEGER %%%%%%%%%%%
  re=im2uint8(im);
% %
%
% %%%% TO   CLASSIFY THE OUTPUT %%%%%%%
%
output=classify(convnet,re);tf1=[];

for ii=1:3
        st=int2str(ii) tf=ismember(output,st);
        tf1=[tf1 tf];
end

output1=find(tf1==1);if

output1==1

        msgbox('EFFUSION')

elseif output1==2

        msgbox('NORMAL')

elseif output1==3

        msgbox('TUBE')


end plotconfusion(YValidation,YPred)
```

SOURCE CODE FOR ResNet -18 ARCHITECTURE TRAINING PROCESS

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % %
% % % % % EAR DRUM DISEASE DETECTION USING ALEXNET DEEP
LEARNING % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % %
clc;
clear all; close all;

warning off;


% % % % % % % % % % % % % % % % % % % % %
% % % % %TRAIN THE IMAGES % % % % % % %
% % % % % % % % % % % % % % % % % % % % %

% % % % % SET TRAINING DATASET FOLDER % % % % % % %
```

```
matlabpath = 'C:\Users\RESHMA.V\Downloads\EAR DRUM\EARDRUM';

data = fullfile(matlabpath,'TRAINING IMAGES');train =
imageDatastore(data,'IncludeSubfolders',true,'LabelSource
','foldernames');

count   = train.countEachLabel;


% % % % % % % % SET VALIDATION DATASET FOLDER% % % % % %
% %

validationPath = fullfile(matlabpath,'TESTING IMAGES');imdsValidation =

imageDatastore(validationPath, ...

'IncludeSubfolders',true,'LabelSource','foldernames');

% % % % % LOAD THE NETWORK % % % % % % %

net = resnet18();layers =
[
        imageInputLayer([227 227 3]);net(2:end-

        3); fullyConnectedLayer(3); softmaxLayer;

        classificationLayer();

        ];


% % % % % %TRAINING   % % % % %

options=trainingOptions('sgdm','MaxEpochs',50,'InitialLea
rnRate',0.0001,'Shuffle','every-epoch', ...
        'ValidationData',imdsValidation, ...
        'ValidationFrequency',30,... 'Verbose',false, ...
        'Plots','training-progress');
```

```
% % % % TRAIN NETWORK USING TRAINING DATA % % % % %

training = trainNetwork(train,layers,options);

% % % % CLASSIFY VALIDATION IMAGES AND COMPUTE ACCURACY %
% % % %

YPred = classify(training,imdsValidation);YValidation =

imdsValidation.Labels;

accuracy = sum(YPred == YValidation)/numel(YValidation);


save training.mat training;load
training.mat;
```

SOURCE CODE FOR ResNet -18 ARCHITECTURE TESTING PROCESS

```
% % % % % % % % % % % % % % % % % % % % % %
% % % % % %TEST THE IMAGES % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % %
% %


[filename,pathname] = uigetfile('*.png*');im =

imread([pathname,filename]); figure,imshow(im),title('INPUT

IMAGE');

% % % % % % % TO RESIZE THE INPUT IMAGE % % %

I = imresize(im,[227, 227]);

% % % % % % % OUTPUT CLASSIFICATION% % % % % % % %

output=classify(training,I);

figure, imshow(I); title(string(output));

msgbox(string(output));
plotconfusion(YValidation,YPred)
```

SOURCE CODE FOR ALEXNET ARCHITECTURE TRAINING PROCESS

```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % %
% % % % % EAR DRUM DISEASE DETECTION USING ALEXNET DEEP
LEARNING % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % %
clc;
clear all; close all;

warning off;


% % % % % % % % % % % % % % % % % % % % %
% % % % %TRAIN THE IMAGES % % % % % % %
% % % % % % % % % % % % % % % % % % % % %

% % % % % SET TRAINING DATASET FOLDER % % % % % % %
matlabpath = 'C:\Users\RESHMA.V\Downloads\EAR DRUM\EARDRUM';

data = fullfile(matlabpath,'TRAINING IMAGES');train =
imageDatastore(data,'IncludeSubfolders',true,'LabelSource
','foldernames');

count   = train.countEachLabel;


% % % % % % % % SET VALIDATION DATASET FOLDER% % % % % %
% %

validationPath = fullfile(matlabpath,'TESTING IMAGES');imdsValidation =

imageDatastore(validationPath, ...

'IncludeSubfolders',true,'LabelSource','foldernames');

% % % % % LOAD THE NETWORK % % % % % % %

net = alexnet;
```

```
layers = [
        imageInputLayer([227 227 3]);net(2:end-

        3); fullyConnectedLayer(3); softmaxLayer;

        classificationLayer();

        ];


% % % % % %TRAINING   % % % % %

options=trainingOptions('sgdm','MaxEpochs',50,'InitialLea
rnRate',0.0001,'Shuffle','every-epoch', ...
        'ValidationData',imdsValidation, ...
        'ValidationFrequency',30,... 'Verbose',false, ...
        'Plots','training-progress');


% % % % TRAIN NETWORK USING TRAINING DATA % % % % %

training = trainNetwork(train,layers,options);

% % % % CLASSIFY VALIDATION IMAGES AND COMPUTE ACCURACY %
% % % %

YPred = classify(training,imdsValidation);YValidation =

imdsValidation.Labels;

accuracy = sum(YPred == YValidation)/numel(YValidation);

analyzeNetwork(training);

save training.mat training;load
training.mat;
```

SOURCE CODE FOR ALEXNET ARCHITECTURE TESTING PROCESS

```
% % % % % % % % % % % % % % % % % % % % % %
% % % % % %TEST THE IMAGES % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % %
% %

[filename,pathname] = uigetfile('*.png*');im =

imread([pathname,filename]); figure,imshow(im),title('INPUT

IMAGE');

% % % % % % % TO RESIZE THE INPUT IMAGE % % %

I = imresize(im,[227, 227]);

% % % % % % % OUTPUT CLASSIFICATION% % % % % % % % %

output=classify(training,I);

figure, imshow(I); title(string(output));

msgbox(string(output));

% % % % % % % % % % % % % % % % % % % % % % % % % % % % %
plotconfusion(YValidation,YPred)
```

REFERENCES

BASE PAPER

OtoMatch: Content-based eardrum image retrieval using deep learning (plos.org)

OtoMatch: Content-based eardrum image retrieval using deep learning - PubMed (nih.gov)

DATASET

OtoMatch: Content-based Eardrum Image Retrieval using Deep Learning | Zenodo

REFERENCES

[1] F. Bray, J. Ferlay, I. Soerjomataram, R. L. Siegel, L. A. Torre, and A. Jemal, ‘‘Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries,’’ CA: A Cancer J. Clinicians, vol. 68, no. 6, pp. 394–424, Nov. 2018.

[2] H. Gelband, P. Jha, R. Sankaranarayanan, and S. Horton, Disease Control Priorities: Cancer, vol. 3. Washington, DC, USA: World Bank, 2015.

[3] J. Rimal, A. Shrestha, I. K. Maharjan, S. Shrestha, and P. Shah, ‘‘Risk assessment of smokeless tobacco among oral precancer and cancer patients in eastern developmental region of nepal,’’ Asian Pacific J. Cancer Prevention, vol. 20, no. 2, pp. 411–415, Feb. 2019.

[4] J. G. Doss, W. M. Thomson, B. K. Drummond, and R. J. R. Latifah, ‘‘Validity of the FACT-H&N (v 4.0) among Malaysian oral cancer patients,’’ Oral Oncol., vol. 47, no. 7, pp. 648–652, 2011.

[5] H. Amarasinghe, R. D. Jayasinghe, D. Dharmagunawardene, M. Attygalla, P. A. Scuffham, N. Johnson, and S. Kularatna, ‘‘Economic burden of managing

oral cancer patients in sri lanka: A cross-sectional hospital - based costing study,'' BMJ Open, vol. 9, no. 7, Jul. 2019, Art. no. e027661.

[6] R. D. Jayasinghe, L. P. G. Sherminie, H. Amarasinghe, and M. A. Sitheeque, ``Level of awareness of oral cancer and oral potentially malignant disorders among medical and dental undergraduates,'' Ceylon Med. J., vol. 61, no. 2, p. 77, Jun. 2016.

[7] O. Kujan, A.-M. Glenny, R. Oliver, N. Thakker, and P. Sloan, ``Screening programmes for the early detection and prevention of oral cancer,'' Austral. Dental J., vol. 54, no. 2, pp. 170–172, Jun. 2009.

[8] N. Haron, R. B. Zain, W. M. Nabillah, A. Saleh, T. G. Kallarakkal, A. Ramanathan, S. H. M. Sinon, I. A. Razak, and S. C. Cheong, ``Mobile phone imaging in low resource settings for early detection of oral cancer and concordance with clinical oral examination,'' Telemed. e-Health, vol. 23, no. 3, pp. 192–199, Mar. 2017.

[9] M. M. R. Krishnan, V. Venkatraghavan, U. R. Acharya, M. Pal, R. R. Paul, L. C. Min, A. K. Ray, J. Chatterjee, and C. Chakraborty, ``Automated oral cancer identification using histopathological images: A hybrid feature extraction paradigm,'' Micron, vol. 43, nos. 2–3, pp. 352–364, Feb. 2012.

[10] M. Aubreville, C. Knipfer, N. Oetter, C. Jaremenko, E. Rodner, J. Denzler, C. Bohr, H. Neumann, F. Stelzle, and A. Maier, ``Automatic classification of cancerous tissue in laserendomicroscopy images of the oral cavity using deep learning,'' Sci. Rep., vol. 7, no. 1, p. 11979, Dec. 2017.