

ACCURATE DETECTION OF DIFFERENT TYPES OF EAR DISEASES IN INFANTS FOR EARLY TREATMENT

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology degree in Biomedical Engineering.

By

FAHIRA PARVEEN M.S (37240023)

LUMEN CHRISTY.V (37240048)



**DEPARTMENT OF BIOMEDICAL ENGINEERING
SCHOOL OF BIO AND CHEMICAL ENGINEERING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

MARCH -2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC | 12B Status by UGC | Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF BIOMEDICAL ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **FAHIRA PARVEEN M. S (37240023)**, **LUMEN CHRISTY. V (37240048)** who have done the project work as a team who carried out the project entitled “**Accurate detection of different types of ear diseases in infants for early treatment**” under my supervision from **October 2020 to March 2021.**

Ms. SINDU DIVAKARAN, M.E., (Ph.D).,

Guide and supervisor

Dr. T. SUDHAKAR, M.Sc., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on 09.04.2021

Internal Examiner

External Examiner

DECLARATION

We, **FAHIRA PARVEEN M. S (37240023)**, **LUMEN CHRISTY. V (37240048)** hereby declare that the Project Report entitled “**Accurate detection of different types of ear diseases in infants for early treatment**” done by us under the guidance of **Ms. Sindu Divakaran, Department of Biomedical Engineering** is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Biomedical Engineering.

DATE: 09.04.2021

PLACE: CHENNAI

1. *M.S. Fahira Parveen*

2. *Lumen*

SIGNATURE OF CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to the **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. J. Premkumar** M.Sc., Ph.D., and **Dr. T. Sudhakar** M.Sc., Ph.D., **Head of the Department, Dept. of Biomedical Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Ms. Sindu Divakaran**, M.E., (Ph.D)., **Department of Biomedical Engineering** for her valuable guidance, suggestions and constant encouragement paving the way for the successful completion of our project work.

We wish to express our thanks to all **Teaching and Non-teaching** staff members of the **Department of Biomedical Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The loss or inability of auditory sense is the most common sensory organ deficiency. Hearing deficiency has a major effect on one's social, cultural, and economic well-being, though it has a low mortality rate. Out of every 1000 new-borns, 5-6 are deaf or hard of hearing. They will not be detected until they are two or three years old, after which permanent trauma would have happened. One of the most prominent reasons for deafness in infants is due to ear diseases. Early detection of ear diseases is the top priority to prevent deafness in infants. Typically, visual examination and manual techniques of CT, MRI, or otoscopic scan images are used for these types of ear disease diagnosis. This method of manually decoding medical images takes a long time and is vulnerable to errors. Thus, in this project, we have developed and applied modified deep learning architecture such as AlexNet, mini GoogLeNet, and LeNet and compared the training by the three architecture. Of the three, AlexNet architecture detected and identified the presence of five different ear diseases such as acute otitis media, glue ear, safe CSOM, otomycosis, other infections and to also check whether the ear is normal, with high accuracy. Using AlexNet architecture at the back end, a web application has also been developed in which an input scanned image will give the output of the type of ear disease saving a lot of time and money invested by the patients. This leads to the earlier prediction of the presence of the ear disease and allows us to take prior actions immediately to avoid further consequences in an effective and cheap manner avoiding human error rate.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	i
	LIST OF ABBREVIATIONS	iv
	LIST OF FIGURES	v
	LIST OF TABLES	vii
1	INTRODUCTION	1
	1.1 OUTLINE OF THE PROJECT	1
	1.2 EAR	2
	1.3 EAR DISEASES	2
	1.3.1 OTOMYCOSIS	3
	1.3.2 SAFE CSOM	3
	1.3.3 GLUE EAR	3
	1.3.4 ACUTE OTITIS MEDIA	3
	1.4 IMAGE PROCESSING	4
	1.4.1 DEEP LEARNING	4
	1.5 WEB DEVELOPMENT	5
2	LITERATURE REVIEW	6
3	AIM AND SCOPE	12
	3.1 AIM	12
	3.2 EXISTING SYSTEM	12
	3.3 PROPOSED SYSTEM	12
	3.4 SCOPE	13
4	MATERIALS AND METHODS	14
	4.1 PYTHON	14
	4.2 COLLECTION OF EAR DISEASE DATASETS	15
	4.3 IMAGE PRE-PROCESSING	15
	4.4 IMAGE AUGMENTATION	16
	4.4.1 IMAGE DATA GENERATOR	17
	4.4.2 AUGMENTATION TECHNIQUES	17
	4.5 CONVOLUTIONAL NEURAL NETWORKS	18
	4.5.1 ALEXNET	19

	4.5.2	MINI GOOGLNET	23
	4.5.3	LENET	26
	4.6	COMPARISON OF ARCHITECTURE	29
	4.6.1	ACCURACY	29
	4.6.2	PRECISION	29
	4.6.3	RECALL	29
	4.6.4	F1-SCORE	29
	4.6.5	WEIGHTED AVERAGE	29
	4.6.6	MACRO-AVERAGE	29
	4.7	MICROSOFT VISUAL STUDIO	30
	4.8	FRONT-END DEVELOPMENT	30
	4.8.1	JAVASCRIPT	31
	4.8.2	HTML	31
	4.8.3	REACTJS	32
	4.9	APPLICATION PROGRAMMING INTERFACE	33
5		RESULTS AND DISCUSSION	34
	5.1	RESULTS	34
	5.1.1	COLLECTION OF DATASETS	34
	5.1.2	PRE-PROCESSING OUTPUT	34
	5.1.3	AUGMENTATION OUTPUT	36
	5.1.4	TRAINING OUTPUTS	37
	5.1.5	COMPARISON OF ARCHITECTURE	44
	5.1.6	WEB APPLICATION	45
	5.2	DISCUSSION	49
6		SUMMARY AND CONCLUSION	51
	6.1	SUMMARY	51
	6.2	CONCLUSION	51
		REFERENCES	52
		APPENDICES	55

LIST OF ABBREVIATIONS

API	APPLICATION PROGRAMMING INTERFACE
CBIR	CONTENT-BASED IMAGE RETRIEVAL
CMS	CONTENT MANAGEMENT SYSTEMS
CNN	CONVOLUTIONAL NEURAL NETWORKS
CSOM	CHRONIC SUPPURATIVE OTITIS MEDIA
DOCALN	DEEP OPTIMAL CONVOLUTIONAL ALEXNET
DOM	DOCUMENT OBJECT MODEL
GPU	GRAPHICAL PROCESSING UNIT
GUI	GRAPHICAL USER INTERFACE
HTML	HYPertext MARK-UP LANGUAGE
IDE	INTEGRATED DEVELOPMENT ENVIRONMENT
JSX	JAVASCRIPT XML
RELU	RECTIFIED LINEAR UNIT
XML	EXTENSIBLE MARK-UP LANGUAGE
XSLT	EXTENSIBLE STYLESHEET LANGUAGE

LIST OF FIGURES

FIG NO.	TITLE	PAGE NO.
1.1	ANATOMY OF EAR	2
4.1	WORKFLOW	15
4.2	CONVOLUTIONAL NEURAL NETWORK	19
4.3	MODIFIED ALEXNET ARCHITECTURE	20
4.4	RELU ACTIVATION FUNCTION	22
4.5	MINI GOOGLENET ARCHITECTURE	24
4.6	INCEPTION MODULE	25
4.7	LENET ARCHITECTURE	28
4.8	REACTJS ARCHITECTURE	32
5.1	COLLECTED DATASET	34
5.2	IMAGE BEFORE PRE-PROCESSING	35
5.3	IMAGE AFTER PRE-PROCESSING	35
5.4	IMAGE BEFORE AUGMENTATION	36
5.5	IMAGE AFTER AUGMENTATION	36
5.6	TRAINING BY ALEXNET	37
5.7	ALEXNET TRAINING GRAPH	38
5.8	INCREASED ACCURACY AFTER FINE TUNING	38
5.9	ALEXNET TRAINING GRAPH AFTER FINE TUNING	39
5.10	ALEXNET LOSS/ACCURACY GRAPH	40
5.11	TRAINING BY MINI GOOGLENET	40
5.12	MINI GOOGLENET TRAINING GRAPH	41
5.13	MINI GOOGLENET LOSS/ACCURACY GRAPH	42
5.14	TRAINING BY LENET	42
5.15	LENET TRAINING GRAPH	43
5.16	LENET LOSS/ACCURACY GRAPH	44
5.17	COMPARISON OF ARCHITECTURE	45
5.18	LOGIN PAGE OF WEB APPLICATION	45
5.19	HOME PAGE OF WEB APPLICATION	46
5.20	GLUE EAR DIAGNOSIS	46
5.21	OTOMYCOSIS DIAGNOSIS	47
5.22	SAFE CSOM DIAGNOSIS	47

5.23	ACUTE OTITIS DIAGNOSIS	48
5.24	OTHER INFECTION DIAGNOSIS	48
5.25	NORMAL EAR DIAGNOSIS	49

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
5.1	TRAINING BY ALEXNET	37
5.2	INCREASED ACCURACY AFTER FINE TUNING	39
5.3	TRAINING BY MINI GOOGLNET	41
5.4	TRAINING BY LENET	43
5.5	COMPARISON OF ARCHITECTURE	44

CHAPTER 1

INTRODUCTION

1.1 OUTLINE OF THE PROJECT

Human perception of the environment is mediated by sensory perceptions. Hearing is the sense that allows people to form relationships, engage in everyday activities, be alerted to risk, and encounter life events. Around 360 million people, or 5% of the global population, have hearing damage that is considered disabling; about 32 million of these are children. The vast majority of people live in low- and middle-income countries around the world. Hearing damage in children can be caused by a variety of ear disorders. This include impacted cerumen (excessive ear wax) and glue ear (non-suppurative otitis media), which is caused by fluid deposition within the ear. According to the World Health Organization (WHO), preventative interventions could prevent about 60% of childhood hearing loss. In order to prevent the deafness caused by ear diseases, it is necessary to diagnose these diseases at an early stage. There are many ways to scan an ear but there is no proper way to detect the type of disease in the ear. The main objective of this project is to determine the occurrence of ear disease in an infant, to reduce the time required to detect the presence of ear diseases in infants and to determine the cheap way to predict the presence of ear diseases. In order to meet these goals, a literature review was conducted to analyse the different types of ear disease detection techniques and methods that are available. Dataset of five different types of ear diseases was collected namely acute otitis media, safe chronic suppurative otitis media, glue ear, otomycosis, other types of ear infections along with normal ear images. This dataset was pre-processed and augmented. Then, the modified dataset was trained using three deep learning neural network architectures that includes AlexNet, mini GoogLeNet and LeNet. The output of the training done by these three architectures were compared to find out which architecture trained the images the best. Using the architecture with high accuracy, a web application was developed in the front end to give the input image and to display the type of ear disease as predicted by the trained architecture. The results are analysed, and conclusions are established.

1.2 EAR

Hearing and equilibrium are also regulated by the ear in mammals. The outer ear, middle ear, and inner ear are the three components of the ear in mammals. The pinna and the ear canal comprise the outer ear. The tympanic cavity and three ossicles comprise the middle ear. The inner ear, which is located inside the bony labyrinth, contains structures that are essential to many senses: the semicircular canals, which facilitate equilibrium and eye monitoring when moving; the utricle and saccule, which help balance when stationary; and the cochlea, which allows hearing, are all found in the bony labyrinth of the inner ear. The ears of vertebrates are symmetrically located on either side of the head, which helps sound localization. The ear is formed by the first pharyngeal pouch and six small swellings called otic placodes that form in the early embryo and are made of ectoderm.

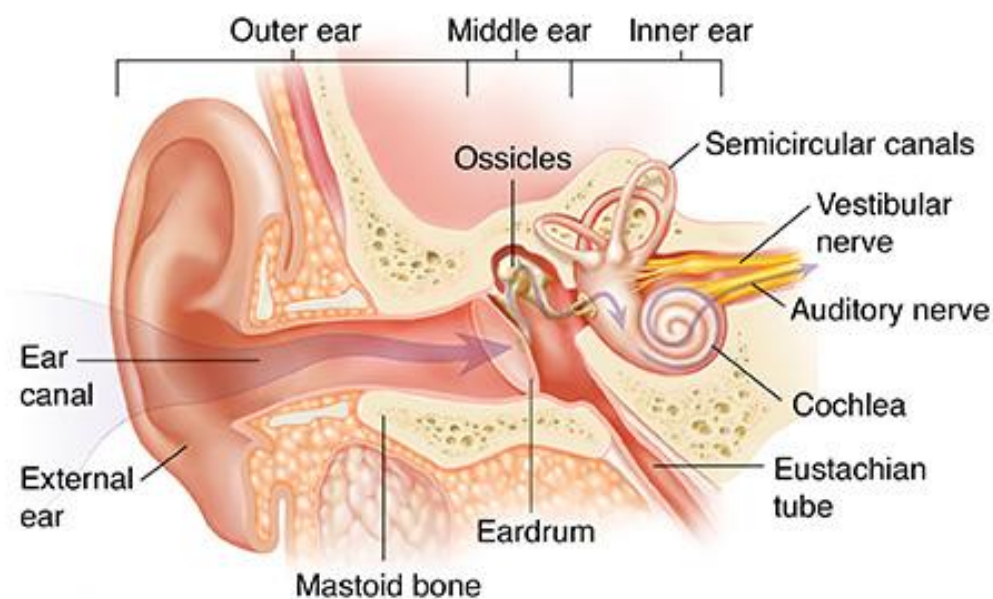


Fig: 1.1 Anatomy of ear

1.3 EAR DISEASES

Disease, cancer, and traumatic injury to the ear will also impair it. Hearing disturbance, tinnitus, and coordination disturbances like vertigo can also be caused by ear diseases. An ear infection is a middle ear inflammation caused by a virus or bacteria that causes fluid to build up behind the eardrum. An ear infection can affect anyone at any age, but it is more common in children than in adults.

since a child's immune system is less strong. 5 percent of children have congenital auricular deformities. [Rami Hallac, et al., 2019] Diagnosis by non-otolaryngologists using otoscopy or otoendoscopy is highly susceptible to misdiagnosis [Dongchul Cha, et al., 2019]

1.3.1 Otomycosis

It is a fungal ear infection that affects the outer ear canal and is caused by a superficial mycotic infection. In tropical countries, it is more popular. Malodorous discharge, inflammation, pruritus, scaling, and extreme pain are all symptoms of the infection, which may be subacute or acute. In the case of otomycosis, the fungus may range from dispersed saprophytic fungal colonies of different colors that cause no symptoms to closely packed fungal debris that fills the canal and involves the tympanic membrane (eardrum).

1.3.2 Safe CSOM

It is cholesteatoma-free CSOM. Based on whether or not the illness is present, it may be classified as active or inactive. Cholesteatoma is a complication of unsafe CSOM. Cholesteatoma is a benign but debilitating lesion of the base of the skull. If the otoscopic image of CSOM is taken it can be found that the tympanic membrane of the ear will be perforated.

1.3.3 Glue-ear

Adhesive otitis, also known as glue ear, is a condition in which the middle area of the ear is accumulated with fluid. Behind the eardrum is this section of the ear. The fluid will thicken and become sticky, similar to glue. Glue ear can progress to a middle ear infection over time. An amber-colored fluid deposit will be found in the case of glue ear.

1.3.4 Acute otitis media

It is an ear infection that causes discomfort. It happens when the middle ear, which is located behind the eardrum, gets inflamed and sick. Fits of fussiness and

excessive moaning are two actions that indicate AOM in infants. A tympanic membrane that is bulging, red, and immobile is strongly linked to acute otitis media.

1.4 IMAGE PROCESSING

An image may be represented as a two-dimensional equation, $f(x, y)$, where x and y are spatial (plane) coordinates. The intensity or grey level of the image at any pair of coordinates (x, y) is the amplitude of f at that point. The image is called a digital image because x , y , and the amplitude values of f are all finite, discrete quantities. Image processing is a technique for performing operations on an image in order to improve it or retrieve valuable material. It's a form of signal processing in which the input is an image and the output is either an image or the image's characteristics/features. [Mingyuan Xin, et al., 2019] Image processing is one of the fastest-growing innovations nowadays. It is also a subject of study in engineering and computer science.

1.4.1 Deep Learning

Not only can computers distinguish images automatically, but they can also identify the different elements in photos and write short sentences detailing each fragment. Any convolutional neural network (trained for classification) can be converted into an image retrieval model, and a pre-trained deep learning model can be converted into an image retrieval system. [Seda Camalan, et al., 2020] By converting fully connected layers into lookup tables, this method is achieved. Deep learning in medical imaging and discuss its potential for future research in the field. The Deep Learning network (CNN) accomplishes this by learning patterns that naturally appear in images. [Jonghoon Kim, et al., 2018] Deep learning, also known as neural networks, is a branch of deep learning that employs a computing paradigm that is heavily influenced by brain structure.

➤ Convolutional Neural Network

Yann LeCun, a postdoctoral computer science professor, first introduced convolutional neural networks (CNN), or ConvNets, in the 1980s. They are Deep Learning algorithms composed of neurons with learnable weights and biases that can differentiate images from one another. [Ahila Priyadharshini, et al.,

2020] As compared to other classification algorithms, the amount of pre-processing needed by a ConvNet is significantly less. [Darvamangala, et al., 2020] CNNs can be used to understand medical images. The architecture of a ConvNet is inspired by the organization of the Visual Cortex and is similar to the communication pattern of neurons in the human brain. Because of their high precision, CNNs are used for image detection and identification. A CNN is a tool that extracts features from images. Manual feature extraction is no longer needed.

1.5 WEB DEVELOPMENT

Web development is the process of creating an internet-based website (World Wide Web). It can range from creating a single static page of plain text to designing complex web apps, electronic companies, and social networking services. Web development has been a growing industry since the commercialization of the Internet. [Fernando Almeida, et al., 2017] Web pages to be rendered quickly and efficiently, providing a positive user experience on mobile, tablet, and desktop devices. The key non-design aspects of developing Web pages, such as writing mark-up and coding, are commonly referred to as web development. It can make use of content management systems (CMS) to make content changes easier and accessible to those with only rudimentary technical knowledge. Front-end development, back-end development, and full-stack development are the three types of Web development. Front-end developers are in charge of the actions and graphics in the user's browser, while back-end developers are in charge of the servers. It is done using ReactJS.

CHAPTER 2

LITERATURE REVIEW

For Dicom image recognition, Haripriya, et al., (2019) proposed a deep learning pre-trained architecture of AlexNet and GoogleNet. The classification efficiency of pre-trained DCNN architectures such as AlexNet and GoogleNet is investigated in this article. Pre-trained networks are used to conveniently adapt the model with one's data collection, as well as to offer cutting-edge functionality and accessibility. For AlexNet and GoogleNet, four separate important ratios of training and research dataset were used in this experiment: 50:50, 60:40, 70:30, and 80:20, respectively. As compared to other ratios and AlexNet output scores, the obtained result achieved the maximum classification accuracy of GoogleNet at 97.02 percent with an error rate of 0.01 in the 70:30.

Pinar Uluer, et al., (2020) developed a successful Robot Companion for the Audiology Rehabilitation process. The aim of this study is to create a useful robot companion for audiology recovery and to test the device with children who are deaf or hard of hearing. The multimodal "stress/emotion/motivation" awareness module for the robot to "understand" how the children feel, as well as the robot's behaviour and feedback module to show the children how the robot "feels," have both been created. This research focuses on the robot's behaviour and feedback module. The chosen affective/affirmative behaviours are measured using tablet games and used as an input tool on the robot during an audiology exam. To assess the children's perceptions of the robot and the behaviour collection, facial data is combined with surveys.

Cesare Alippi, et al., (2018) used a technique to transfer neural network convolution to an embedded device in the AlexNet and VGG-16 cases. This paper takes a step in that direction, proposing a technique for designing and porting CNNs to low-resource embedded systems, bridging the distance between CNNs and embedded systems. In order to accomplish this aim, this paper uses approximate computing methods to reduce the deep learning architecture's computational load and memory use by sacrificing precision for memory and computation. The suggested technique has been tested on two well-known CNNs, AlexNet and VGG-16, as well as an image-recognition program and two relevant off-the-shelf embedded platforms.

Satya Singh, et al., (2020) developed a 3D deep learning system for medical images. A brief mathematical overview of 3D CNN and the pre-processing steps needed for medical images before feeding them to 3D CNNs in this article, which traces the history of how the 3D CNN was built from its machine learning origins was included. Important studies in the field of 3D medical imaging analysis using 3D CNNs (and their variants) in a variety of medical fields, including recognition, segmentation, diagnosis, and localization has been reviewed. It was then followed by a discussion of the complexities of using 3D CNNs in medical imaging (and the use of deep learning models in general) as well as potential future developments in the area.

For medical applications, Sudhakar Sengan, et al., (2020) used the process of image resolution through optimal deep AlexNet architecture. MISR, also known as Deep Optimal Convolutional AlexNet (DOCALN), is a design proposed in the paper that finds the best learning rates for the ReLU activation function. Low Resolution (LR) medical images can be implemented using this optimal deep learning structure. The findings of our proposed model's experiments are compared to those of Convolution Neural Networks (CNN) variants in terms of image quality measurement, SR performance analysis, and execution time.

Dongchul Cha, et al., (2019) developed an algorithm for automatic ear condition diagnosis using ensemble deep learning and a large otoendoscopy image database. Using a vast database of otoendoscopic photographs obtained in the clinical setting, this thesis provides a machine learning model for automatically diagnosing ear disease. To distinguish eardrum and external auditory canal characteristics into six types of ear disorders, researchers used a total of 10,544 otoendoscopic images to train nine public convolution-based deep neural networks (Normal, Attic retraction, Tympanic perforation, Otitis external myringitis, Tumour). Following the evaluation of many optimization systems, two of the best-performing models were chosen to form an ensemble classifier, which combines the classification scores of each classifier.

Rami Hallac, et al., (2019) used a convolutional neural network to distinguish ear abnormalities from 2D images. CNN was used in this analysis to automatically detect ear deformity in 2D images. This study used a total of 671 profile photos,

including 457 photos of patients with ear deformities and 214 photos of patients with natural ears. Photographs were cropped to the ear border and arbitrarily split into three datasets: training (60%), validation (20%), and checking (20%). To create an ear deformity detector model in MATLAB, the paper changed the softmax classifier in the last layer of GoogLeNet, a deep CNN. All of the photographs were judged to be of good quality and suitable for teaching and research. The machine was trained in about 2 hours, and the training accuracy was nearly 100%. The precision of the test was 94.1 percent. It shows that deep learning has a lot of scope for detecting ear deformities. These machine learning approaches have the potential to be used to assess patient results in the future.

The system of otomatch content-based eardrum image retrieval using deep learning was used by Seda Camalan, et al., (2020). This study used eardrum images taken with a digital otoscope to develop a content-based image retrieval (CBIR) scheme (dubbed OtoMatch) for normal, middle ear effusion, and tympanostomy tube conditions. It presents a method for converting any classification-trained convolutional neural network into an image retrieval model. It turned a pre-trained deep learning model into an image retrieval system as a proof of concept. The machine was trained and tested using a database of 454 labeled eardrum images (179 standard, 179 effusion, and 96 tube cases). The suggested approach, when extracting the most comparable image from the database, had an average accuracy of 80.58 percent (SD 5.37 percent) and an overall F1 score of 0.90 on 10-fold cross-validation.

Alshazly, et al., (2020) proposed a system for unconstrained ear identification using a deep convolutional neural network. In a first computational study of ear recognition on the unconstrained EarVN1.0 dataset, this paper uses state-of-the-art Deep Convolutional Neural Networks (CNNs), such as AlexNet, VGGNet, Inception, ResNet, and ResNeXt. The researchers' experiments show that their networks, which are fine-tuned using custom-sized inputs calculated uniquely for each CNN architecture, achieve state-of-the-art recognition efficiency, with a single ResNeXt101 model achieving a rank-1 recognition accuracy of 93.45%. Furthermore, they use an ensemble of fine-tuned ResNeXt101 models to achieve the highest rank-1 recognition accuracy of 95.85%. They use the t-SNE algorithm

to investigate and visualize the learned features in order to clarify output variations between models to make our findings more understandable. The ear pictures of the various subjects are represented by well-separated clusters in feature visualizations.

The process of using AlexNet deep transfer learning for ear recognition was used by Almisreb, et al., (2018). The well-known AlexNet Convolution Neural Network (AlexNet CNN) was used in this paper to apply Transfer Learning to human recognition based on ear images. AlexNet CNN was embraced and fine-tuned for our issue domain in this article. To consider 10 groups instead of 1000, the last fully connected layer is replaced with another fully connected layer. To boost the network's non-linear problem-solving performance, another Rectified Linear Unit (ReLU) layer is introduced. 250 ear images from 10 participants for training and 50 ear images for evaluation and testing were used to train the fine-tuned network. In our application, the suggested fine-tuned network performs admirably, with 100% validation accuracy.

Yaman, et al., (2018) developed a system for determining age and gender from ear photographs. This paper presents a quantitative study of ear photos for extracting soft biometric traits such as age and gender. For ear representation, this research used both geometric and appearance-based elements. The thesis used AlexNet, VGG-16, GoogLeNet, and SqueezeNet, which are well-known convolutional neural network models. They were fine-tuned using a large-scale ear dataset generated from the Multi-PIE face dataset's profile and close-to-profile face pictures. A domain adaptation has been done in this direction. On the small-scale target ear dataset, which includes just about 270 ear images for testing, the revised models were fine-tuned once more. According to the findings of the experiments, methods based on appearance were equivalent to methods based on geometric characteristics. For gender classification, we achieved 94 percent accuracy, while for age classification, we achieved 52 percent accuracy. These findings suggest that ear pictures may be used to classify people based on their age and gender.

Convolutional neural networks in medical image perception have been studied by Sarvamangala, et al., (2021). Anomalies of the human body are captured using imaging methods. Health image interpretation is usually done by trained medical

personnel. Convolutional neural networks (CNNs) are powerful image-processing techniques. The aim of this paper was to provide a concise overview of CNN applications of medical image interpretation. The overarching goal is to encourage medical picture comprehension researchers to use CNN's widely in their study and diagnosis. The main medical image comprehension tasks have been added, including image recognition, segmentation, translation, and identification. The use of CNN in medical picture interpretation of brain, breast, liver, and other organ diseases has been extensively and comprehensively examined.

Fevziye Irem Eyiokur, et al., (2018) developed Domain Adaptation for Ear Recognition Using Deep Convolutional Neural Networks. When deep convolutional neural network models are used for ear recognition, the relevance of domain adaptation is first demonstrated. They mixed various deep convolutional neural network models to increase the results even further. The impact of ear image quality, such as lighting and aspect ratio, on classification efficiency was examined in detail in this article. Finally, in the field of ear identification, the issue of dataset bias was discussed. The accuracy was increased by 4% by combining various deep convolutional neural network models. Image consistency has also been shown to have an effect on the outcomes. Provided an ear picture, the tests were able to identify the dataset that it had come with 99.71 percent accuracy, indicating that there is a clear bias among the ear recognition datasets.

Unconstrained Ear Recognition by Domain Adaptive Deep Learning Models of Convolutional Neural Networks was developed by Marwin Alejo, et al., (2019). AlexNet, GoogLeNet, Inception-v3, Inception-ResNet-v2, ResNet-18, ResNet-50, SqueezeNet, ShuffleNet, and MobileNet-v2 are investigated and compared for use of unconstrained ear biometric recognition in this paper. 250 unconstrained ear images are captured and acquired from the internet via web crawlers, and preprocessed using simple image processing techniques, including the use of contrast limited adaptive histogram equalization to increase ear image accuracy. Each CNN architecture is structurally studied and fine-tuned to meet the ear recognition specifications. As attribute extractors, earlier layers of CNN architectures are used. The last 2-3 layers in each CNN architecture are fine-tuned, and layers of the same kind are replaced with layers of the same kind for ear recognition models, allowing them to distinguish 10 groups of ears rather than 1000.

The majority of the unconstrained ear images obtained are used for training, while the remaining 20% are used for research and validation. The test results are compared in terms of above-95 percent accuracy confidence, training time, training and validation outcomes such as acquired features and losses, and test results in terms of above-95 percent accuracy confidence. ResNet, AlexNet, and GoogleNet outperformed all other architectures with a 97-100 percent accuracy belief, making them ideal for unconstrained ear biometric identification.

Muhammad Shazam Kashar (2018) performed an automated feature extraction and image classification system for otitis media analysis. The key aim was to create Artificial Intelligence applications for analyzing and detecting otitis media contamination using photographs and open-source software focused on computer vision and deep learning. The aim was to provide these tools to assist and encourage medical practitioners in identifying and diagnosing otitis media at an early stage. Two computing systems were created using advanced computer vision, deep learning, and transfer learning approaches. The first digital framework assists professionals by adding feature engineering to photographs and removing those attributes that will assist them in manually classifying images. The second computing framework is a deep learning-based automatic image recognition software that can be used to render the final diagnosis. The deep learning-based image recognition program has performed admirably, with an accuracy of 82.2 percent using the InceptionV3 architecture and 80 percent using the MobileNet architecture.

CHAPTER 3

AIM AND SCOPE

3.1 AIM

In this project, we propose accurate prediction of different types of ear diseases in infants for early treatment. The acquired images are pre-processed, augmented and trained using modified deep learning architectures.

The main objective of this project is to determine the occurrence of ear disease in an infant, to reduce the time required to detect the presence of ear diseases in infants and to determine the cheap way to predict the presence of ear diseases.

3.2 EXISTING SYSTEM

The goal of this project is to create an effective robot companion for audiology rehabilitation and to test the system with children who are deaf or hard of hearing. The multimodal stress/emotion/motivation awareness module for the robot to understand how the children feel, as well as the robot's behaviour and feedback module to show the children how the robot feels, have both been created. This research focuses solely on the robot's behaviour and feedback module. The chosen affective/affirmative behaviours are measured using tablet games and used as an input tool on the robot during an audiology exam. To assess the children's perceptions of the robot and the behaviour collection, facial data is combined with surveys.

3.3 PROPOSED SYSTEM

In this project we develop and apply a modified deep learning architecture AlexNet GoogLeNet, LeNet to effectively detect and identify the presence of five different ear diseases such as acute otitis media, glue ear, safe CSOM, otomycosis, other infections and to also check whether the ear is normal without the need of several consultations from different doctors. This leads to earlier prediction of the presence of the ear disease and allows us to take prior actions immediately to avoid further consequences in an effective and cheap manner avoiding human error rate. A web application will also be developed in which an input scanned image will give the

output of the type of ear disease saving a lot of time and money invested by the patients.

3.4 SCOPE

The majority of ear infections do not result in long-term complications. However, recurrent ear infections can lead to severe complications such as impaired hearing, speech or developmental delays in toddlers, spread of infection and tearing of the ear drum. Early detection of hearing loss in infants, along with timely and effective therapies, will help children avoid developmental delays and improve communication, schooling, and social progress.

Deep learning is a form of machine software that simulates the brain's network of neurons. It is a branch of machine learning which is referred to as deep learning because it employs deep neural networks. Convolutional Neural Network (CNN) is a multi-layered neural network with a special architecture that determines the performance by extracting increasingly complex features from the data at each layer. CNNs are excellent at perceptual tasks.

By training the images with the CNN architectures, the ear diseases could be predicted at an early stage with high accuracy. This avoids several consultations from doctors and allows to take prior actions immediately to avoid further consequences in an effective manner avoiding human error rate. It also provides a cheap method for detecting the presence of ear disease and the type of ear disease as it saves time and money required for diagnosis. It can be also used as a hospital application. In this way deafness caused by ear diseases can be prevented to a certain level.

CHAPTER 4

MATERIALS AND METHODS

4.1 PYTHON

Python is an encoded high-level general-purpose programming language. It is consistently one of the most widely used programming languages. Python's programming philosophy prioritizes code readability, as shown by the extensive use of indentation. Its language constructs and object-oriented style were designed to assist programmers in writing simple, logical code for both small and large projects. Python is garbage-collected and dynamically typed. Multiple programming paradigms are supported, including hierarchical, object-oriented, and functional programming. Because of its extensive standard library, Python is sometimes referred to as batteries included language.

Python is a programming language that can be used in a variety of ways. Many of its features support functional programming and aspect-oriented programming, as well as object-oriented programming and formal programming. Many other paradigms, such as architecture by contract and logic programming, are provided by extensions. [Akshansh Sharma, et al., 2021] Python was named the fastest-growing programming language in recent years, based on research from around the world. Relevant knowledge and methods related to the Python programming language are also included in this article. Python manages memory with complex typing and a combination of reference counting and a cycle-detecting garbage collector. It also includes late binding (dynamic name resolution), which connects process and variable names during program execution.

Python was created with the intention of being extremely extensible. Python is designed to be a language that is simple to understand. Its formatting is clean and uncluttered, and it mostly uses English keywords instead of punctuation in other languages. Python's classes enable programmers to specify their own styles, which are commonly found in object-oriented programming.

In this work, python is used to develop the back-end processes that is for the modification of datasets, and training by architecture.

4.2 COLLECTION OF EAR DISEASE DATASETS

A data set is a set of information. Deep Learning has emerged as the preferred tool for tackling a wide range of difficult real-world problems. It is, without a doubt, the most effective approach for computer vision activities. Deep learning's dominance in computer vision is shown in the picture above. A deep network will segment and classify the key points of any individual in an image with enough preparation. These deep learning architectures, which have been performing admirably, need a lot of power, which is data.

This step included collecting otoscopic images of the five forms of ear diseases as well as images of the normal ear from different sources. The image dataset of the six forms of ear diseases was downloaded from multiple internet research sites. Most of the dataset may be found from third-party tools, as the dataset has been such a valuable asset in the deep learning age. The consistency improves as the dataset size grows. The dataset was divided into training and testing data subsets after the images were obtained. The convolutional neural networks were trained using the training dataset, and the accuracy of the trained neural network was validated using the testing dataset. Fig:4.1 briefs the processes from collection of datasets to web development.

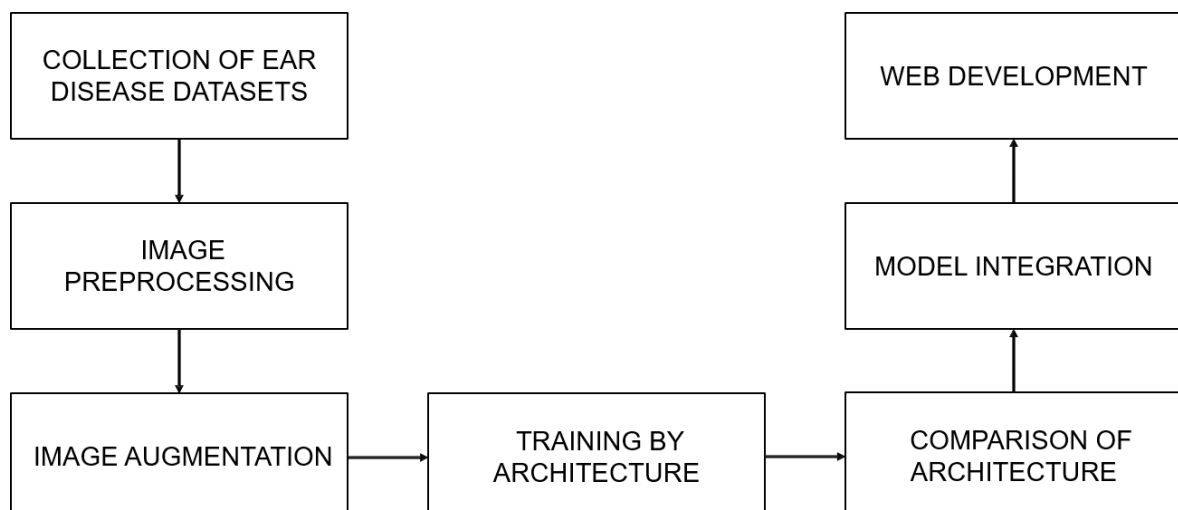


Fig: 4.1 Workflow

4.3 IMAGE PRE-PROCESSING

The expression image pre-processing refers to operations on images at the most basic level of complexity, where both the input and output are intensity images. If entropy is an information metric, these operations do not increase image information quality, but rather decrease it. The aim of pre-processing is to optimize image data by suppressing unwanted distortions or enhancing certain image features that are essential for subsequent processing and study. 3D CNN from its machine learning roots and provided a brief mathematical explanation of 3D CNN, and described the pre-processing steps for medical images before feeding them to 3D CNNs. [Satya Singh, et al., 2020] Image pre-processing allows the use of image redundancy. The brightness of neighbouring pixels referring to the same or identical real-world objects is the same or similar. If a skewed pixel can be extracted from the image with precision, it can be restored as the average value of nearby pixels. [Marwin Alejo, et al., 2019] Usually, the image dataset will not be clean in most cases. It means the data will contain non-uniform data formats, missing values, outliers, and features with very different ranges. The data cannot be used as training dataset for the model. For those reason, the data has been brought to a standardized format by effectively resizing it. This is done using the arguments AspectAwarePreprocessor and ImageToArrayPreprocessor. The aspect aware pre-processor helps to resize the images while the image to array pre-processor aligns the images to a standard height, width and depth.

4.4 IMAGE AUGMENTATION

Image data augmentation is a method for artificially increasing the size of a training dataset by modifying images in the dataset. Image augmentation algorithms includes geometric transformations, colour space augmentations, kernel filters, mixing images, random erasing, feature space augmentation, adversarial training, generative adversarial networks, neural style transfer, and meta-learning. [Shorten, et al., 2019] Image augmentation artificially produces training images using a variety of imaging techniques or a mixture of techniques, such as spontaneous rotation, shifts, shear, and flips, among others. Deep learning neural network models that are trained on more data become more skilled, and augmentation techniques may generate variations of the images that enhance the fit models' ability to generalise what they've learned to new images. For this purpose, class ImageDataGenerator is used.

4.4.1 ImageDataGenerator

Using real-time data augmentation, it creates minibatches of image data. At each epoch, the ImageDataGenerator class ensures that the model receives new variations of the images. However, it only restores the converted images and does not include them in the original corpus. If this is the case, the object would be exposed to the initial images several times, causing the model to overfit. Another benefit of ImageDataGenerator is that it consumes less memory. This is because if we don't use this class, we'll load all of the images at the same time. However, when we use it, we load the images in batches, which saves a significant amount of memory.

4.4.2 Augmentation techniques

➤ *Random Rotation*

Image rotation is a popular augmentation technique that causes the model to become invariant to the object's orientation. By passing an integer value in the rotation_range argument to the ImageDataGenerator class, images can be rotated arbitrarily between 0 and 360 degrees. Any pixels can shift beyond the image as it is rotated, leaving a vacant space that must be filled in. This can be filled in a variety of ways, including a constant value, closest pixel values, and so on. rotation_range=30 and fill_mode="nearest" has been used in this case.

➤ *Random Shifts*

It's possible that the object in the image isn't always in the middle. To solve this problem, the image's pixels can be shifted horizontally or vertically by applying a constant value to all of the pixels. For a vertical shift of an image, the height_shift_range parameter is used, and for a horizontal shift of an image, the width_shift_range parameter is used. If the value is a float integer, it indicates the percentage of the image's width or height that should be shifted. Otherwise, where the value is an integer, the width or height are merely changed by as many pixel numbers. width_shift_range=0.1, height_shift_range=0.1 is used in this case.

➤ *Random Flips*

For flipping around the vertical or horizontal axes, the ImageDataGenerator class has arguments `horizontal_flip` and `vertical_flip`. This technique, however, should be used in accordance with the object in the image. `horizontal_flip=True` is used in this case.

➤ *Random Zoom*

The zoom augmentation either zooms in or zooms out of the image at random. In the `zoom_range` parameter, the ImageDataGenerator class accepts a float value for zooming. The lower and upper limits in a list of two values can be stated. If a float value is not assigned, the zoom range would be `[1-zoom_range, 1+zoom_range]`. Any value less than 1 will enlarge the image. Any value greater than 1 causes the image to zoom out. `zoom_range=0.2` is used in this case.

4.5 CONVOLUTIONAL NEURAL NETWORKS

Humans actively analyse the environment in which they live. Assumptions are made on what is seen and they move on without even realizing it. When something is seen, it is labelled based on previous learning. Machine learning algorithms for medical image processing, stressing clinical aspects of the field concentrates on convolutional neural networks. [Justin Ker, et al., 2017] The primary sensory pathway, which connects the eyes to the brain, is the reason we can make sense of the world around us. Though vision begins with the eyes, it is the primary visual cortex in the brain that interprets what is seen. When an object is seen, the light receptors in the eyes transmit signals to the primary visual cortex, which processes the information. What the eye sees is decoded by the primary visual cortex.

People and computers view things differently. Just numbers exist in the computer universe. Pixels are two-dimensional arrays of numbers that can be used to represent any image. A special kind of Artificial Neural Network is used to train an algorithm on how to recognize objects in images: Convolutional Neural Network (CNN). Their name comes from one of the network's most significant operations: convolution. The brain is the source of inspiration for CNNs. It is intended for the processing of organized data arrays, such as images. The convolutional framework, which is made up of convolution and pooling layers, and a series of completely

linked layers on top that performs the classification operation, make up the CNN architecture. [Alshazly, et al., 2020]

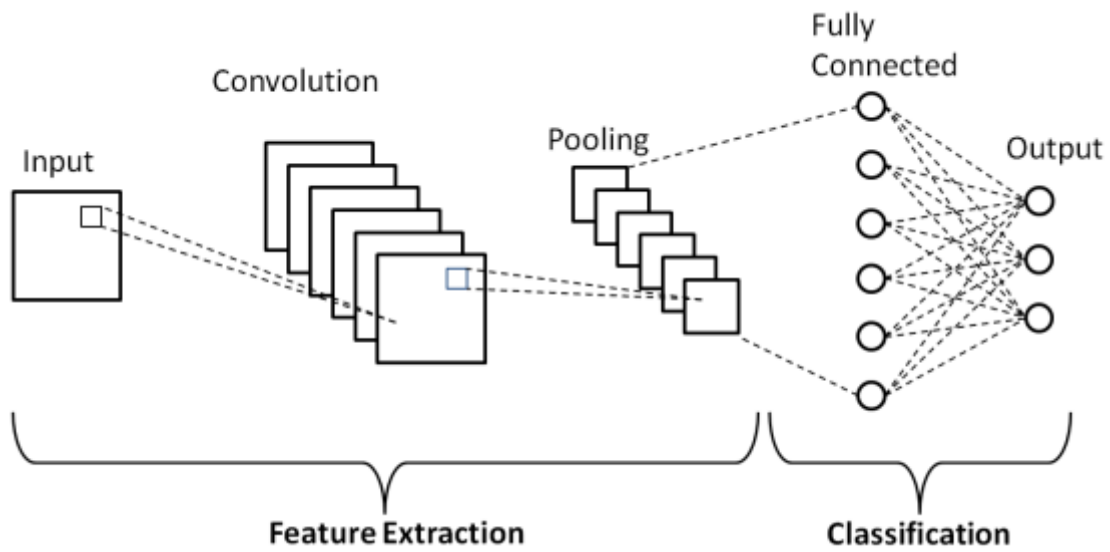


Fig: 4.2 Convolutional Neural Network

The architecture of convolutional neural networks differs from that of normal neural networks. An input is transformed by a sequence of hidden layers in a regular neural network. Each layer is made up of a group of neurons, each of which is completely connected to all neurons in the layer before it. Finally, the output layer, which represents the predictions, is the fully connected layer.

Fig: 4.2 summarises the working methodology of a CNN. CNNs are a different kind of neural network. The layers in CNN are divided into three dimensions: width, height, and depth. Furthermore, the neurons in one layer do not bind to any of the neurons in the following layer, but just a small portion of it. Patterns in the input signal, such as arcs, gradients, circles, or even eyes and faces, are easily detected by CNNs. It is because of this property that convolutional neural networks are so effective in computer vision. Convolutional neural networks, unlike previous computer vision algorithms, can run on a raw image without any pre-processing.

In this work, we employ three CNNs, AlexNet, MiniGoogLeNet and LeNet, to train the dataset

4.5.1 AlexNet

In this project, AlexNet architecture is utilized to effectively identify and detect five different types of ear diseases such as acute otitis media, glue ear, safe CSOM, Otomycosis and to check if the ear is normal and also for other ear infections. AlexNet is a convolutional neural network that has had a significant impact on machine learning, especially in the application of deep learning to machine vision. After each convolutional and fully-connected layer, it added ReLU activations. Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, Krizhevsky's Ph.D. advisor, collaborated on the concept.

Low Resolution (LR) medical images can be implemented using Deep Optimal Convolutional AlexNet (DOCALN), which calculates the ReLU activation function's optimal learning rates which is an optimal deep learning structure. [Sudhakar Sengan, et al., 2020] It solves the image classification problem by taking an image from one of 1000 different groups (e.g., cats, dogs, etc.) and producing a vector of 1000 numbers as the output. The probability that the input image belongs to the i^{th} class is represented by the i^{th} element of the output vector. As a result, the output vector's total element count is 1.

Though AlexNet was undeniably a breakthrough algorithm at the time, it requires the use of at least two GPUs to function properly. Many people may not have access to several GPUs. By placing half of the model's neurons on one GPU and the other half on another GPU, AlexNet allows for multi-GPU training. This not only allows for the preparation of a larger model but also reduces the training time.

➤ *Layers in AlexNet architecture*

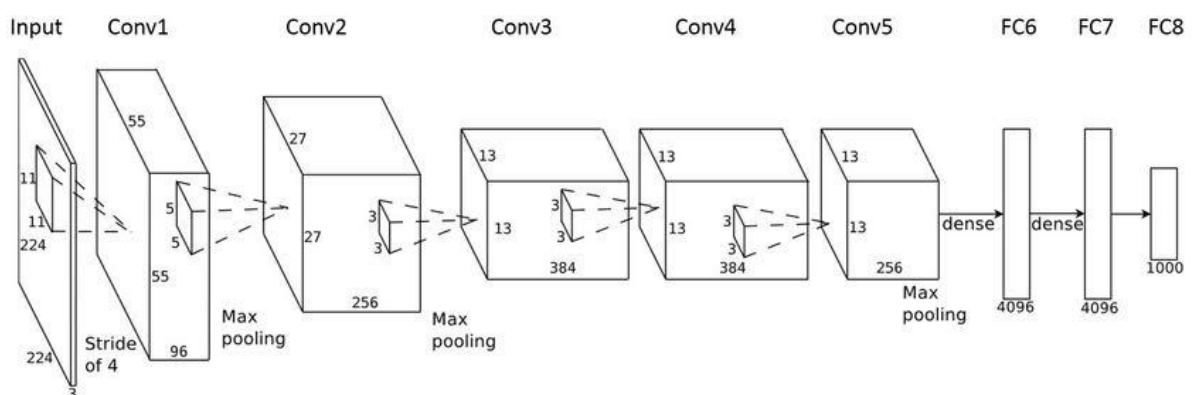


Fig: 4.3 Modified AlexNet architecture

The network can take the input image having height, width as multiples of 32 and 3 as channel width. It is to be of the input size as $224 \times 224 \times 3$. [Cesare Alippi, et al., 2020] Fig: 4.3 briefs the architecture of AlexNet used in this process. There are several techniques for various types of learning approaches, such as supervised, semi-supervised, and unsupervised learning. [Zahangir Alom, et al., 2019] The modified AlexNet architecture performs the initial convolution and max-pooling using 7×7 and 3×3 kernel sizes respectively. The network has five blocks in total. All the five block consists of layers of convolution, pooling, activation functions and dropout functions.

Convolution Layer: A CNN's first layer is often a Convolutional Layer. Convolutional layers perform a convolution on the input before forwarding the output to the next layer. The pixels in a convolution's receptive region are all converted to a single value.

The convolutional layer's final output is a vector. Multiple types of convolutions will be used depending on the form of issue to be solved and the type of functionality to be learned. In this work, 2D convolutional network is used. In a conv2D layer, a filter or kernel "slides" over the 2D input data, performing elementwise multiplication. As a result, the data will be summed into a single output pixel.

Pooling Layer: The function map's dimensions are reduced by using pooling layers. As a result, the number of parameters to learn and the amount of processing in the network are also reduced. The features present in a region of the feature map created by a convolution layer are summed up by the pooling layer. As a result, rather than precisely placed features created by the convolution layer, further operations are performed on summarised features. As a result, the model is more resistant to changes in the location of features in the input image.

In this work, Max Pooling is used. It is a type of pooling that selects the maximum factor from the area of the feature map covered by the filter. As a result, the contribution of the max-pooling layer will be a feature map with the most notable features from the previous feature map. Here, the pool size is 3 and the stride size is 2.

Activation function: The performance of a neural network is determined using activation functions, which are mathematical equations. Rectified Linear Unit or ReLU is used in this case.

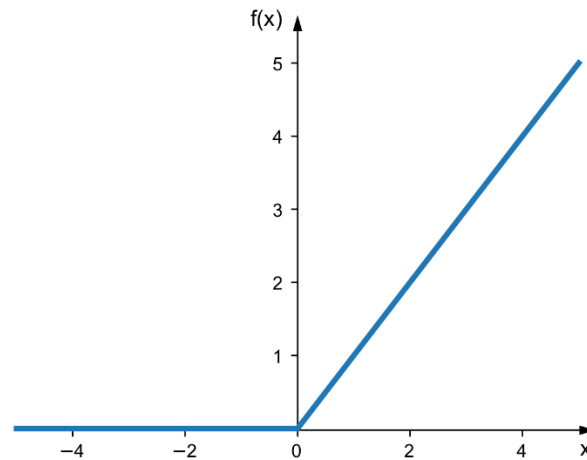


Fig: 4.4 ReLU activation function

Fig: 4.4 shows the graph of ReLU function. ReLU is a piecewise linear function that, if the input is positive, outputs the input directly; otherwise, it outputs zero. Since a model that uses it is easy to learn and therefore achieves improved results, it has become the default activation mechanism for many forms of neural networks. It's a deceptively simple formula: $\max(0, z)$. The below function describes the graph of the activation function.

$$R(z) = \begin{cases} z & z > 0 \\ 0 & z \leq 0 \end{cases}$$

Batch Normalization: Batch normalization is a strategy for training very deep neural networks that standardizes each mini-input batch to a layer. This stabilizes the learning process and significantly reduces the number of training epochs used to train deep networks. The training dataset has been split into 32 images per batch.

Dropout Layer: Dropout is a strategy for preventing overfitting in a model. At each modification of the training process, Dropout operates by setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0. In this work, a combination of fraction of 0.5 and 0.25 is used in different blocks.

Flatten layer: After obtaining the pooled featured map the next step is to flatten it. Flattening is the method of converting the whole pooled feature map matrix into a single column that is then fed into the neural network for processing.

Fully connected layer: Feed forward neural networks are what the Fully Connected Layer is all about. Fully Connected Layers are the network's final layers. The output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer, is the input to the fully connected layer.

Softmax classifier: The softmax classifier takes its name from the softmax algorithm, which is used to squash raw class scores into normalized positive values that add up to one, allowing the cross-entropy loss to be implemented.

In the first block and the second block, there is a 2D convolutional layer, max pooling layer of kernel size 3 and stride size 2, activation layer, batch normalisation and a drop out layer of fraction of 0.25. In the third block there is repetitive layers of 2D convolutional layer, batch normalisation and activation layer followed by a max pooling layer and a drop out layer of fraction 0.25.

The fourth and the fifth blocks is a set of fully connected layers that has a flattened layer, and neurons of size 4096, activation layer, batch normalization and a drop out layer of fraction 0.5. The last layer is the softmax classifier.

4.5.2 Mini GoogLeNet

The GoogLeNet convolutional neural network is built on the Inception architecture. It employs Inception modules, which enable the network to choose from a variety of convolutional filter sizes in each block. An Inception network piles these modules on top of each other, with periodic stride 2 max-pooling layers to halve the grid's resolution.

Google's GoogLeNet (a.k.a. Inception V1) took first place in the ILSVRC 2014 competition. It had a 6.67 percent top-five error score. This was very similar to human-level results. The network used a CNN inspired by LeNet but added a new feature called an inception module. Batch normalisation, image distortions, and RMSprop were all included. To greatly minimise the number of parameters, this module is built on many very small convolutions. GoogLeNet has 22 layers but has around a dozen times fewer parameters than AlexNet and is built on a new model

called inception. The input image is screened independently by various filters in the inception layers. [Yaman, et al., 2018] They were fine-tuned using a large-scale ear dataset generated from the Multi-PIE face dataset's profile and close-to-profile face pictures. The design used a 22-layer deep CNN, but the number of parameters was limited from 60 million to 4 million (AlexNet).

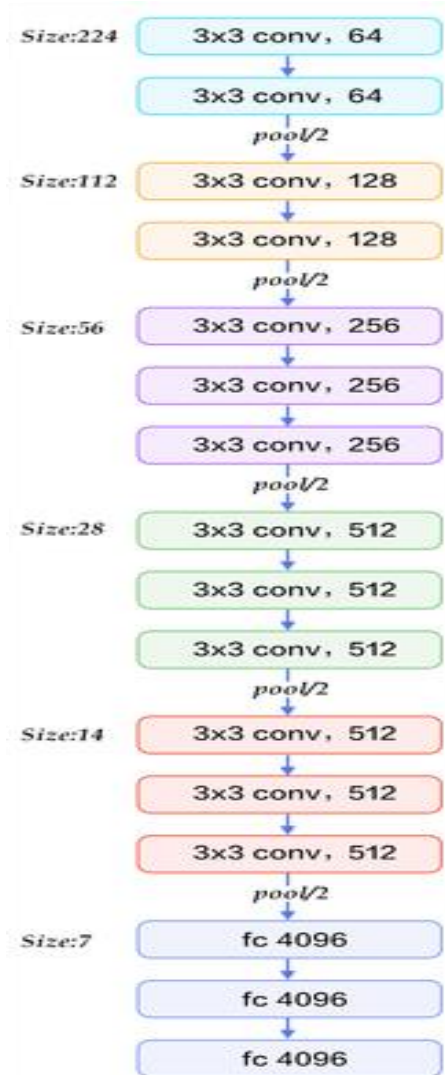


Fig: 4.5 Mini GoogLeNet Architecture

The increased use of computational power within the network is the key feature of this architecture. Instead of using the naive version of inception module, dimensionality reduction was used to make the architecture computationally efficient. [Farhana Sultana, et al., 2019] This was accomplished by a well-crafted architecture that allows the network's breadth and width to be increased while maintaining the computational budget stable. The architectural decisions were based on the

Hebbian theory and the wisdom of multi-scale production to maximize efficiency. A network that has been pre-trained using either the ImageNet or Places365 data sets will be loaded.

Since GoogleNet accepts 224*224 images as input by default, and since it is so deep and after a series of convolution and pooling layers, using 50*50 images (or 36*36 after crop) would result in a very small output after passing the input through some layers with kernel sizes smaller than the kernel size of the next layer. Using transfer learning, retrain a GoogLeNet network to perform a new task.

The most popular method for performing transfer learning is to use networks that have been pre-trained on the ImageNet data collection. If the new role is similar to scene classification, the network trained on Places-365 will provide better results. The architecture used in this paper is Mini GoogLeNet. Fig: 4.5 explains the architecture of the mini GoogLeNet employed.

➤ *Layers in the Mini GoogLeNet*

Convolutional module: This module consists of a 2D convolutional layer, batch normalization and an activation layer that is ReLU activation function.

Inception module: Convolutional Neural Networks use Inception Modules to allow for more effective computation and deeper networks by reducing dimensionality with stacked 11 convolutions. The modules were created to address problems such as computing cost and overfitting, among others. In a nutshell, the approach is to use different kernel filter sizes within the CNN and order them to run on the same level rather than piling them sequentially.

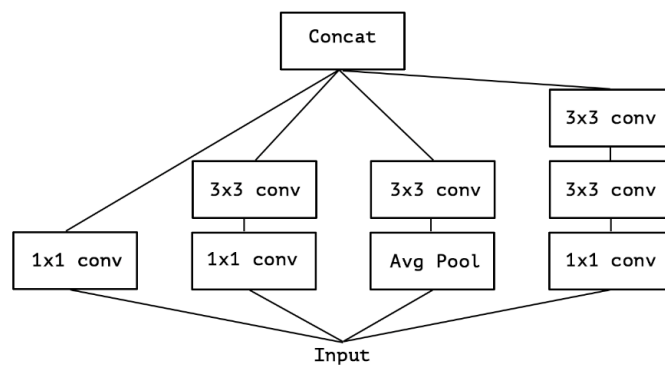


Fig: 4.6 Inception Module

Downsampling layer: The standard convolution (without stride) operation produces an output image that is the same size as the input image. Strided convolution, on the other hand, results in downsampling, or a reduction in the size of the input image.

Pooling layer: The function map's dimensions are reduced by using pooling layers. As a result, the number of parameters to learn and the amount of processing in the network are also reduced.

In the Mini GoogLeNet architecture average pooling and max pooling is used in different blocks. Average pooling differs from Max pooling in that it saves a lot of detail about a block's or pool's less significant components. Unlike Max Pooling, which actually discards them by selecting the highest value, Average Pooling mixes them in. This is useful in a number of contexts where such data is needed.

Dropout Layer: Dropout is a strategy for preventing overfitting in a model. At each modification of the training process, Dropout operates by setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0. In this work, a fraction of 0.5 is used.

Softmax classifier: The softmax classifier takes its name from the softmax algorithm, which is used to squash raw class scores into normalized positive values that add up to one, allowing the cross-entropy loss to be implemented.

In this work after the convolutional block two inception modules followed by a downsampling module is used. In the next step, four inception modules followed by a downsampling module is used. In the next step two inception modules followed by an average pooling layer, dropout layer and a softmax classifier is utilised.

4.5.3 LeNet

The LeNet is a concept that defines the core components of CNN and corresponds to its inception. Other algorithms, such as SVM, can produce comparable or even better results than the LeNet due to a lack of hardware, especially GPUs (Graphics Processing Units, a specialized electronic circuit designed to rapidly manipulate and alter memory to facilitate the creation of images in a frame).

Deep neural networks are now the most advanced machine learning models in a broad range of fields, from image recognition to natural language processing, and are commonly used in academia and industry. [Alshazly Selvikvg Lundervold, et al.,

2019] Convolutional neural network (CNN) LeNet automatically assessing a person's emotional state based on their speech. CNN models differ from LeNet models in several ways, but they are all built on LeNet. [Iskhakova, et al., 2020] LeNet contains the basic components of a convolutional neural network, such as the convolutional layer, pooling layer, and fully connected layer, laying the groundwork for potential convolutional neural network growth.

There are seven levels of LeNet. In addition to input, every other layer will train parameters. C_x represents the convolution layer, S_x represents the subsampling layer, F_x represents the fully connected layer, and x represents the layer index. LeNet refers to LeNet-5, a simple convolutional neural network that aided deep learning development. Convolutional neural networks are a form of feed-forward neural network that uses artificial neurons to respond to a subset of the surrounding cells in the coverage range and process large quantities of data.

In 1989, Bell Labs' Yann LeCun became the first to extend the backpropagation algorithm to functional implementations, assuming that providing restrictions from the task's domain would increase the ability to learn network generalization dramatically. He read handwritten numbers using a backpropagation-trained convolutional neural network and successfully applied it to identifying handwritten zip code numbers.

This was the first version of LeNet, as it was later dubbed. In the same year, LeCun published another paper in which he introduced a small handwriting digit recognition problem and showed that single-layer networks had poor generalization capacities despite the problem being linearly separable.

These DNNs are tweaked and added to the description of acoustic scenes. To empirically compare these approaches, a series of studies were performed on the TUT Acoustic Scenes. When using shift-invariant function detectors on a multi-layered, constrained network, the model could perform well. [Duppada, et al., 2018] These results, he said, showed that reducing the number of free parameters in a neural network would increase the generalization ability of the neural network.

Convolution	28x28
Average Pooling	14x14
Convolution	10x10
Average Pooling	5x5
Convolution	1x1
FC	84
FC	10

Fig: 4.7 LeNet architecture

➤ *Layers in LeNet Architecture*

Convolutional Layer: A CNN's first layer is often a Convolutional Layer. Convolutional layers perform a convolution on the input before forwarding the output to the next layer. The pixels in a convolution's receptive region are all converted to a single value. In this architecture, 2D convolutional network is used. In a conv2D layer, a filter or kernel “slides” over the 2D input data, performing elementwise multiplication. As a result, the data will be summed into a single output pixel.

Pooling Layer: The function map's dimensions are reduced by using pooling layers. As a result, the number of parameters to learn and the amount of processing in the network are also reduced. In this architecture, Max Pooling is used. It is a type of pooling that selects the maximum factor from the area of the feature map covered by the filter. The pool size and the stride size are 2.

Activation function: The performance of a neural network is determined using activation functions, which are mathematical equations. Rectified Linear Unit or ReLU is used in this case.

Softmax classifier: The softmax classifier takes its name from the softmax algorithm, which is used to squash raw class scores into normalized positive values that add up to one, allowing the cross-entropy loss to be implemented.

In this architecture, 2D convolution and Max pooling is used in the first block and the second block which is followed by a fully connected layer and a softmax classifier.

4.6 COMPARISON OF ARCHITECTURE

After training the dataset using three different architecture comparison has to be made among them to find out the architecture with the best training accuracy. This comparison was done based on the certain parameters.

4.6.1 Accuracy

The percentage of precise predictions for the test results is known as accuracy. It's easy to figure out by calculating the number of valid predictions by the total number of predictions.

$$accuracy = \frac{\text{correct predictions}}{\text{all predictions}}$$

4.6.2 Precision

Precision is classified as the percentage of relevant examples (true positives) among all the examples expected to belong to a certain class.

$$precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

4.6.3 Recall

The proportion of examples predicted to belong to a class compared to all of the examples that actually belong in the class is known as recall.

$$recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

4.6.4 F1-Score

Simply put, the harmonic mean of a class's accuracy and recall. $(2 \times \text{Recall} \times \text{Precision}) / (\text{Recall} + \text{Precision})$ would be the F1 score.

4.6.5 Weighted average

The weighted average is the total of all class scores after multiplying their respective class proportions.

4.6.6 Macro-average

A macro-average is the plain average of all class scores. It calculates the metric separately for each class and then averages the results.

4.7 MICROSOFT VISUAL STUDIO

Microsoft Visual Studio is used as the IDE in this project. Microsoft Visual Studio is the company's integrated development environment (IDE). It's used to make blogs, online applications, digital utilities, and smartphone applications, among other things. Windows API, Windows Shapes, Windows Presentation Foundation, Windows Store, and Microsoft Silverlight are some of the Microsoft applications creation tools used by Visual Studio. It has the ability to generate both native and controlled code.

Visual Studio comes with a code editor that supports IntelliSense (code completion) and code refactoring. The integrated debugger can be used as both a source-level and a machine-level debugger. Client-side JavaScript study over the last decade are divided into six categories: static analysis, dynamic analysis, formalization and reasoning, type safety and JIT optimization, security for web applications, and empirical studies. [Kwangwon Sun, et al., 2017] A code profiler, a designer for creating GUI programs, a web designer, a class designer, and a database schema designer are among the other built-in tools. It embraces plug-ins that improve flexibility at almost any level, such as incorporating support for source management frameworks (such as Subversion and Git) and new toolsets for domain-specific languages, such as editors and graphic designers, or toolsets for other facets of the software development lifecycle.

Visual Studio supports 36 separate programming languages, and the code editor and debugger can support almost any programming language (to varying degrees). C, C++, C++/CLI, Visual Basic.NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML, and CSS are among the built-in languages. Plug-ins have support for other languages such as Python, Ruby, Node.js, and M, among others. In the past, Java (and J#) were supported.

The Community version of Visual Studio is the most basic and is available for free. The version of Visual Studio that is officially supported is 2019. The front-end development is done in the Microsoft Visual Studio using JavaScript in ReactJS.

4.8 FRONT-END DEVELOPMENT

The process of translating data to a graphical interface using HTML, CSS, and JavaScript so that users can display and communicate with the data is known as

front-end web creation. The objectives for front-end creation are speedy development, usability, and efficiency. In this work, JavaScript along with HTML is used that is done in ReactJS library.

4.8.1 JavaScript

JavaScript is a high-level programming language that is one of the Web's main technologies. It was created as a website scripting language, but it quickly gained popularity as a general-purpose programming language, and it is now the most commonly used programming language. JavaScript is a text-based programming language that helps you to make interactive web pages on both the client and server sides. JavaScript adds interactive elements to web pages, allowing users to communicate with them.

JavaScript supports event-driven, functional, and imperative programming styles as a multi-paradigm language. It has APIs for working with text, dates, regular expressions, standard data structures, and the Document Object Model, among other things (DOM).

4.8.2 HTML

HTML (Hypertext Markup Language) is that the standard terminology for documents that are intended to be viewed during a browser. It can be aided by technologies like Cascading Style Sheets (CSS) and scripting languages like JavaScript, which affect web page behaviour and material. CSS determines the appearance and layout of text. Various HTML tags are must for developing HTML tags. [Ankush Sharma, et al., 2018] Online browsers receive HTML documents from a web server or locally stored files and convert them to multimedia web pages. HTML originally provided clues for the document's appearance and defined the layout of a web page semantically.

HTML elements are the components that make up HTML pages. Images and other artefacts, such as interactive forms, may be inserted in the rendered page using HTML constructs. HTML allows you to construct organized documents by indicating structural semantics for text elements like headings, paragraphs, lists, links, quotes, and other elements. Tags, which are written with angle brackets, separate HTML elements.

4.8.3 ReactJS

React, also known as React.js or ReactJS, is a front-end JavaScript library for creating user interfaces or UI components that is open-source. ReactJS may be a JavaScript library that's used to build user interfaces for single-page applications. We can also make reusable UI components with React. Jordan Walke, a Facebook software developer, was the first to build React. In 2011, React was launched on Facebook's newsfeed, followed by Instagram.com in 2012.



Fig: 4.8 ReactJS architecture

React enables developers to build massive web applications that can alter data without requiring a page reload. ReactJS is a JavaScript library which is deployed to develop reusable user interface (UI) components. [Sanchit Aggarwal, et al., 2018] React's main goal is to be fast, scalable, and straightforward. It only applies to application user interfaces. View in the MVC template corresponds to this. It can be used with other JavaScript libraries or frameworks in MVC applications, such as Angular JS. HTML is not needed to write React code. However, NPM and Node are needed for using React in production. React "elements" are made with JSX. HTML can be written in React using JSX. Data can be calculated, manipulated, and validated using JavaScript. Without using any createElement() methods, JSX helps us to write HTML elements in JavaScript and position them in the DOM. HTML tags are converted to react elements using JSX. While JSX is not needed, it does make writing React applications easier.

➤ JSX

JavaScript XML is abbreviated as JSX. HTML can be written in React using JSX. In React, JSX makes it easy to write and add HTML.

It is faster than standard JavaScript since it optimizes when converting to standard JavaScript. It facilitates the development of models for us. Rather than splitting the markup and logic in different formats, React makes use of components.

➤ *Components*

Components are self-contained, interchangeable code bits. They perform the same functions as JavaScript functions, but they operate independently and return HTML with a render function.

Class components and Function components are the two kinds of components; in this tutorial, we'll focus on Class components.

➤ *Unidirectional data flow and Flux*

One-way data flow is implemented in React, making it simple to the reason for the app. Flux is a pattern that aids in the unidirectional flow of data.

4.9 APPLICATION PROGRAMMING INTERFACE

A collection of routines, functions, protocols, and rules make up an application programming interface (API). [Heryandi A 2018] It lays out how various program modules or packages can work with one another. APIs (Application Programming Interfaces) are structures used in programming languages that make it easier for developers to build complex features. An application programming interface (API) for the Web is referred to as a Web API. They abstract more complex coding away from you and replace it with a simpler syntax. After taking the image from the user in the front-end the API communicates the input to the back-end where it is processed and an output is generated which is then displayed in the front-end.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 RESULTS

5.1.1 Collection of datasets

In the first step otoscopic images of different ear diseases including acute otitis media, safe CSOM, glue ear, otomycosis, other types of ear infections and images of normal ear. Fig: 5.1 is a sample of the dataset collected. 300 images have been collected in total, 50 for each type of disease from different websites such as Kaggle and dataworld.



Fig: 5.1 Collected Dataset

After collection of datasets, the datasets are separated into different categories of diseases.

5.1.2 Pre-processing output

The dataset collected undergo pre-processing to optimize image data by suppressing unwanted distortions or enhancing certain image features that are essential for subsequent processing. Fig: 5.2 shows an image before pre-

processing. If the images are of varied dimensions the learning time required by the CNN increases therefore it is necessary to process the image to uniform dimensions.



Fig: 5.2 Image before pre-processing

The image before pre-processing was of varied height and width but after pre-processing, the output image was of uniform height and width. Before pre-processing the image was 1000 x 990 in dimension.

Fig: 5.3 shows the image after pre-processing has been done. After pre-processing the dimensions of the image was changed to 224 x 224.



Fig: 5.3 Image after pre-processing

5.1.3 Augmentation output

The next step is data augmentation, in this phase the images collected are set to standard resolution format. It is done to artificially increase the size of a training dataset by modifying images in the dataset. The collected images have been augmented to get modified copies of the images. Fig: 5.4 show the image before augmentation.



Fig: 5.4 Image before augmentation

Image augmentation artificially produces training images using a variety of imaging techniques or a mixture of techniques, such as spontaneous rotation, shifts, shear, and flips, among others. Fig: 5.5 shows the images after augmentation. It is clearly shown that giving one input image has generated 10 output images that are uniform in dimensions.



Fig: 5.5 Image after augmentation

5.1.4 Training Outputs

From the images obtained, 75% have been used for the training and 25% have been used for testing.

➤ AlexNet Output

Fig: 5.6 shows the training done by the AlexNet architecture.

```
6/6 [=====] - 15s 2s/step - los
[INFO] evaluating after initialization...
      precision    recall  f1-score   support

   glue-ear      0.71      0.38      0.50        13
    normal      1.00      0.47      0.64        15
     other      1.00      0.90      0.95        10
    otitis      0.85      0.85      0.85        13
 otomycosis      0.44      1.00      0.62        12
  safe CSOM      1.00      1.00      1.00        12

 accuracy              0.75              75
 macro avg      0.83      0.77      0.76        75
 weighted avg      0.83      0.75      0.75        75

[INFO] re-compiling model...
[INFO] fine-tuning model...
```

Fig: 5.6 Training by AlexNet

The obtained output shows how accurately the algorithm has trained the dataset.

Table 5.1 summarises the values obtained from the training output of AlexNet architecture. The precision, recall and f-score of safe CSOM has been 100%. The values of other diseases are less than 100. Low values have been obtained in the case of glue-ear and otomycosis.

Table 5.1 Training by AlexNet

	Precision	Recall	F1-Score	Support
Glue Ear	0.71	0.38	0.5	13
Normal	1.00	0.47	0.64	15
Other	1.00	0.9	0.95	10
Otitis	0.85	0.85	0.85	13
Otomycosis	0.44	1.00	0.62	12
Safe CSOM	1.00	1.00	1.00	12

Fig: 5.7 shows the graph before fine tuning the CNN. From the graph, it can be clearly seen that the network doesn't predict all the six categories of diseases with 100% accuracy. For certain diseases like glue-ear and otomycosis the values obtained are very less.

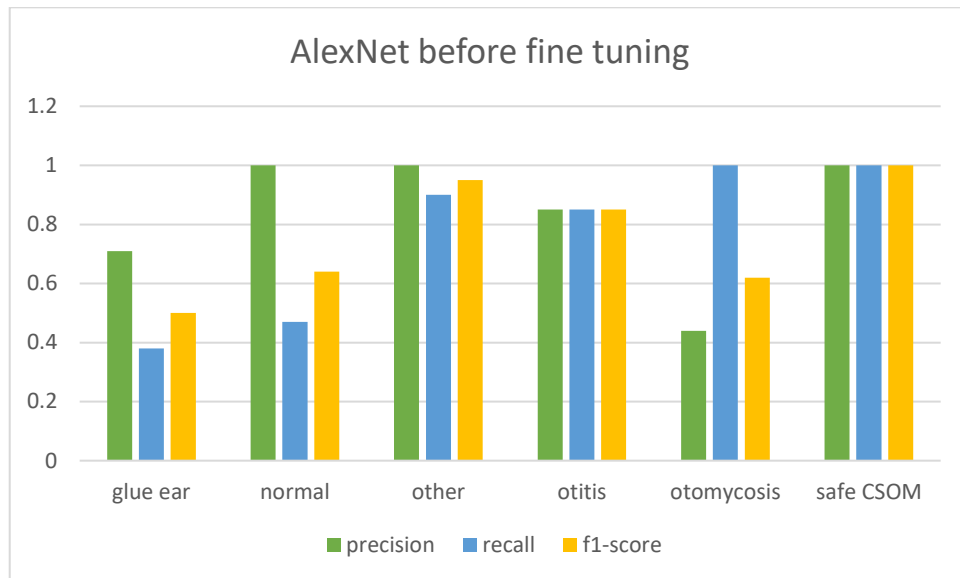


Fig: 5.7 AlexNet training graph

In order to increase the accuracy of the architecture, fine tuning was done where the datasets were trained for an increased number of epochs. Fig: 5.8 shows the output obtained after fine tuning.

```

loss
acc
val_loss
val_acc
[INFO] evaluating after fine-tuning...

```

	precision	recall	f1-score	support
glue-ear	1.00	1.00	1.00	13
normal	1.00	1.00	1.00	15
other	1.00	1.00	1.00	10
otitis	1.00	1.00	1.00	13
otomycosis	1.00	1.00	1.00	12
safe CSOM	1.00	1.00	1.00	12
accuracy			1.00	75
macro avg	1.00	1.00	1.00	75
weighted avg	1.00	1.00	1.00	75

Fig: 5.8 Increased accuracy after fine tuning

The accuracy has raised from 0.75 to 1.00 after fine tuning. That is the architecture has been trained to detect the diseases with 100% accuracy.

Table 5.2 summarises the values obtained from the training output of AlexNet architecture after fine tuning.

Table 5.2 Increased accuracy after fine tuning

	Precision	Recall	F1-Score	Support
Glue Ear	1.00	1.00	1.00	13
Normal	1.00	1.00	1.00	15
Other	1.00	1.00	1.00	10
Otitis	1.00	1.00	1.00	13
Otomycosis	1.00	1.00	1.00	12
Safe CSOM	1.00	1.00	1.00	12

The precision, recall and f-score of all the six categories of diseases have been 100%. From the values obtained, it can be concluded that the network has trained the architecture very effectively and it is very efficient in predicting the six categories of images.

Fig: 5.9 is the graphical representation of the values obtained after fine tuning.

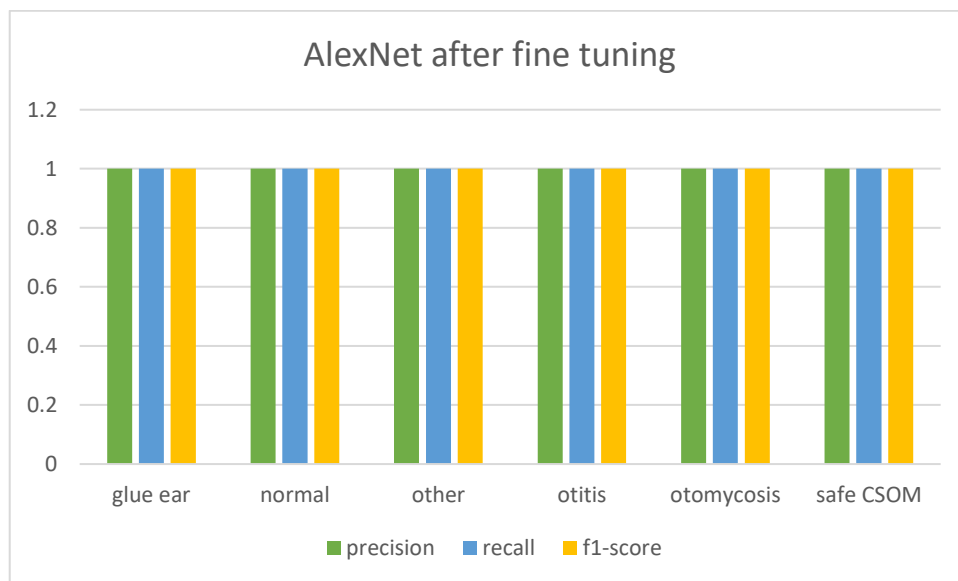


Fig: 5.9 AlexNet training graph after fine tuning

From the graph it can be inferred that all the six diseases have been predicted with complete accuracy.

Fig: 5.10 shows the graph that is obtained after the training. It is the training loss and accuracy graph.

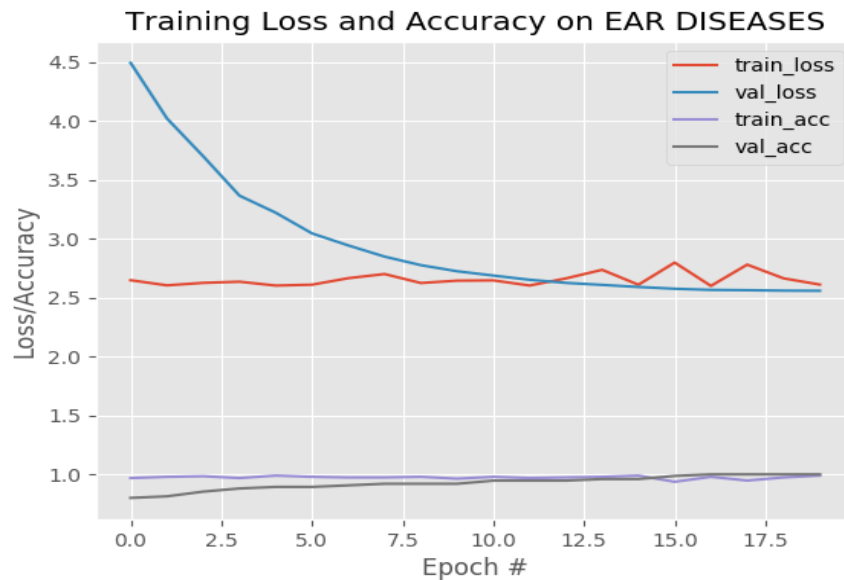


Fig: 5.10 AlexNet loss/accuracy Graph

The training loss has started around 2.6 and after fluctuating a little around 13-17 epochs it has returned to its original value. The validation loss has started at 4.5 and gradually decreased to 2.5. The training accuracy has been stable at 1.0 while the validation accuracy has started at 0.9 at gradually stabilised to 1.0.

➤ Mini GoogLeNet Output

Fig: 5.11 shows the training done by the Mini GoogLeNet architecture.

```

ned and being set to 0.0 in labels with no predicted samples. Use
parameter to control this behavior.
_warn_prf(average, modifier, msg_start, len(result))
precision    recall  f1-score   support

   glue-ear      0.00      0.00      0.00        13
    normal      0.37      1.00      0.54        15
     other      0.00      0.00      0.00        10
    otitis      0.00      0.00      0.00        13
 otomycosis      0.36      0.33      0.35        12
   safe CSOM      0.00      0.00      0.00        12

 accuracy              0.25        75
  macro avg           0.12      0.22      0.15        75
 weighted avg           0.13      0.25      0.16        75

```

Fig: 5.11 Training by Mini GoogLeNet

From the values obtained it can be seen that the precision and accuracy for the particular deep learning model is very less. The accuracy value is 0.25. Table: 5.3 shows the values obtained from the training done by the Mini GoogLeNet.

Table 5.3 Training by Mini GoogLeNet

	Precision	Recall	F1-Score	Support
Glue Ear	0.00	0.00	0.00	13
Normal	0.37	1.00	0.54	15
Other	0.00	0.00	0.00	10
Otitis	0.00	0.00	0.00	13
Otomycosis	0.36	0.33	0.35	12
Safe CSOM	0.00	0.00	0.00	12

The precision, recall and f-score of all the six categories of diseases have been very less. The precision, recall, and f1-score of almost all diseases is zero, with the values of normal ear images and otomycosis being slightly higher.

Fig: 5.12 is the graphical representation of the values obtained from the training of the dataset by Mini GoogLeNet. It can be seen that the bar graph shows value for normal and otomycosis only.

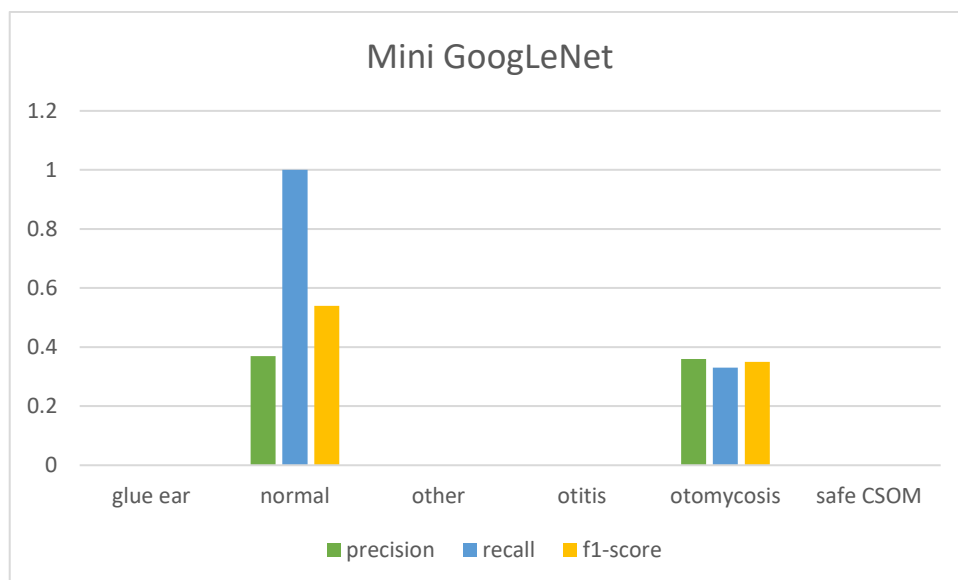


Fig: 5.12 Mini GoogLeNet training graph

Fig: 5.13 shows the training loss accuracy graph of GoogLeNet architecture.

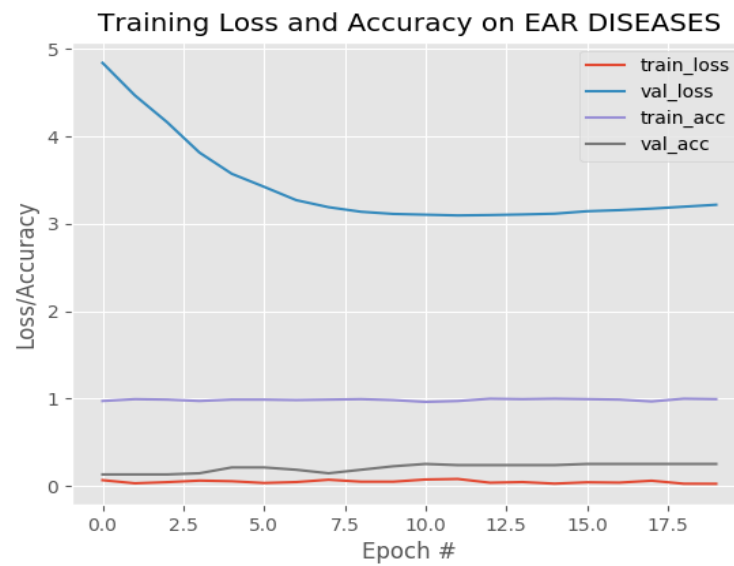


Fig: 5.13 Mini GoogLeNet loss/accuracy graph

The training loss has been stable at 0. The validation loss has started at 5 and gradually decreased to 3. The training accuracy has been stable at 1.0 but the validation accuracy has been around 0. The training by Mini GoogLeNet is very poor and not as efficient as AlexNet.

➤ LeNet Output

Fig: 5.14 shows the training by LeNet architecture.

```
[INFO] evaluating after fine-tuning...
```

	precision	recall	f1-score	support
glue-ear	1.00	0.54	0.70	13
normal	0.79	1.00	0.88	15
other	1.00	1.00	1.00	10
otitis	0.58	0.54	0.56	13
otomycosis	1.00	0.67	0.80	12
safe CSOM	0.58	0.92	0.71	12
accuracy			0.77	75
macro avg	0.83	0.78	0.78	75
weighted avg	0.82	0.77	0.77	75

Fig: 5.14 Training by LeNet

The obtained output shows how accurately the algorithm has trained the dataset.

Table 5.4 summarises the values obtained from the training output of LeNet architecture.

Table 5.4 Training by LeNet

	Precision	Recall	F1-Score	Support
Glue Ear	1.00	0.54	0.7	13
Normal	0.79	1.00	0.88	15
Other	1.00	1.00	1.00	10
Otitis	0.58	0.54	0.56	13
Otomycosis	1.00	0.67	0.8	12
Safe CSOM	0.58	0.92	0.71	12

The precision, recall and f-score of all the six categories has been found to be in between 0.5 and 1.0. Among all the diseases, the value of other infection is 1 and the lowest value has been obtained in case of otitis media, which means that the network identifies other infections very effectively but it identifies otitis media with less accuracy.

Fig: 5.15 is the graphical representation of the values obtained from the training of the dataset by LeNet. The values obtained clearly indicated that the training by LeNet architecture is much better compared to the training by Mini GoogLeNet.

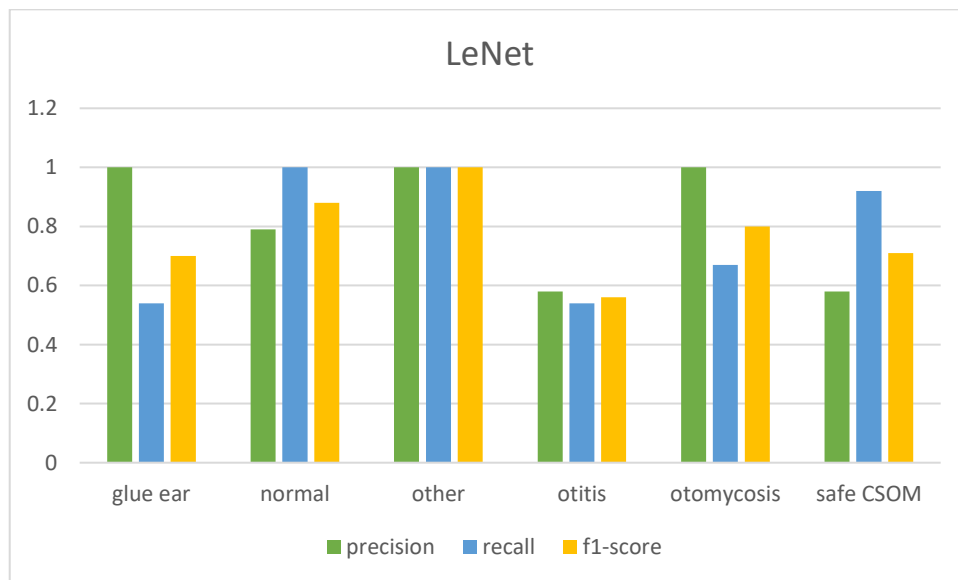


Fig: 5.15 LeNet training graph

Fig: 5.16 shows the training loss accuracy graph of LeNet. The training loss has started at 0.775 and it has been fluctuating throughout.

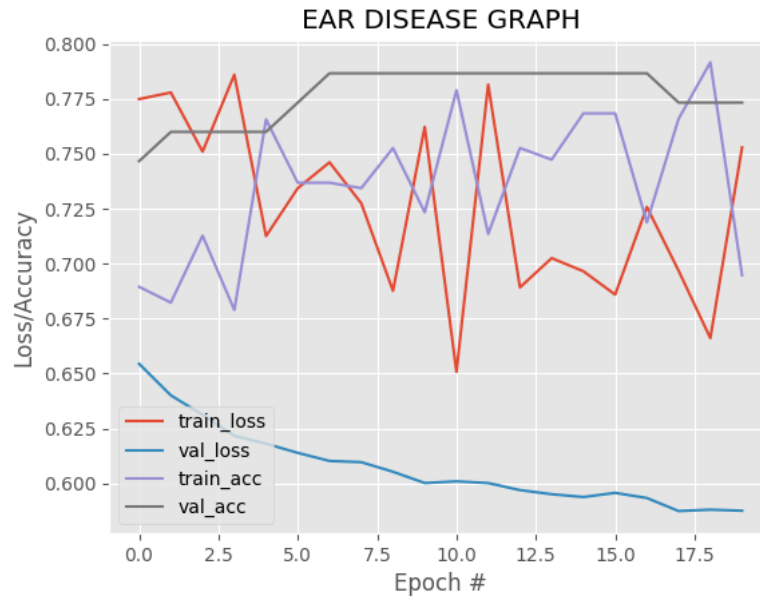


Fig: 5.16 LeNet loss/accuracy graph

The validation loss has started at 0.650 and gradually decreased to below 0.600. The training accuracy has started at 0.690 and has been fluctuating throughout. The validation accuracy has been started at 0.750 and at the end of 20 epochs, it is at 0.775. Here, the accuracy is higher than the Mini GoogLeNet but it is less compared to AlexNet. The accuracy value is 0.77.

5.1.5 Comparison of architecture

After training the dataset by the three architecture, using the training outputs generated, a comparison was made between the three architecture to find which architecture trained the dataset the best. Table 5.5 summarises the parameters that has been used to compare the architectures and the values that has been obtained.

Table 5.5 Comparison of Architecture

ARCHITECTURE	ALEXNET	MINI GOOGLNET	LENET
Accuracy	1.00	0.25	0.77
Macro Average	1.00	0.15	0.78
Weighted Average	1.00	0.16	0.77
F1 score	1.00	0.16	0.77
Precision	1.00	0.13	0.77
Recall	1.00	0.25	0.78

Fig: 5.17 shows the graphical representation of the accuracy, macro average, weighted average, f1-score, precision and recall of the three architectures.

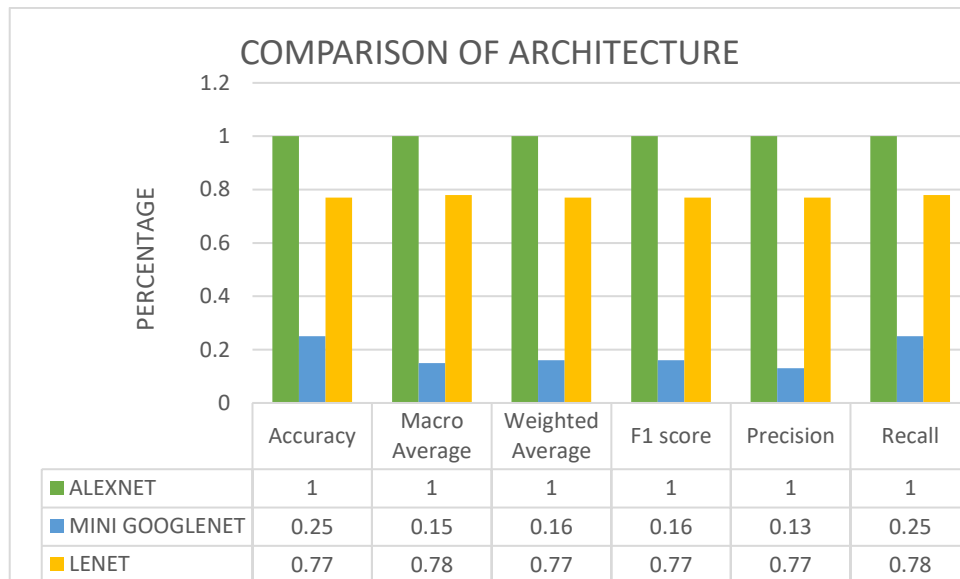


Fig: 5.17 Comparison of Architecture

From the above values, it was concluded that AlexNet architecture trains the images the best and it is used in the back-end.

5.1.6 Web application

A web application was created using ReactJS. Fig: 5.18 shows the login page of the web application developed. After logging in to the web page using the username and password which is “ear”, the home page will be loaded.

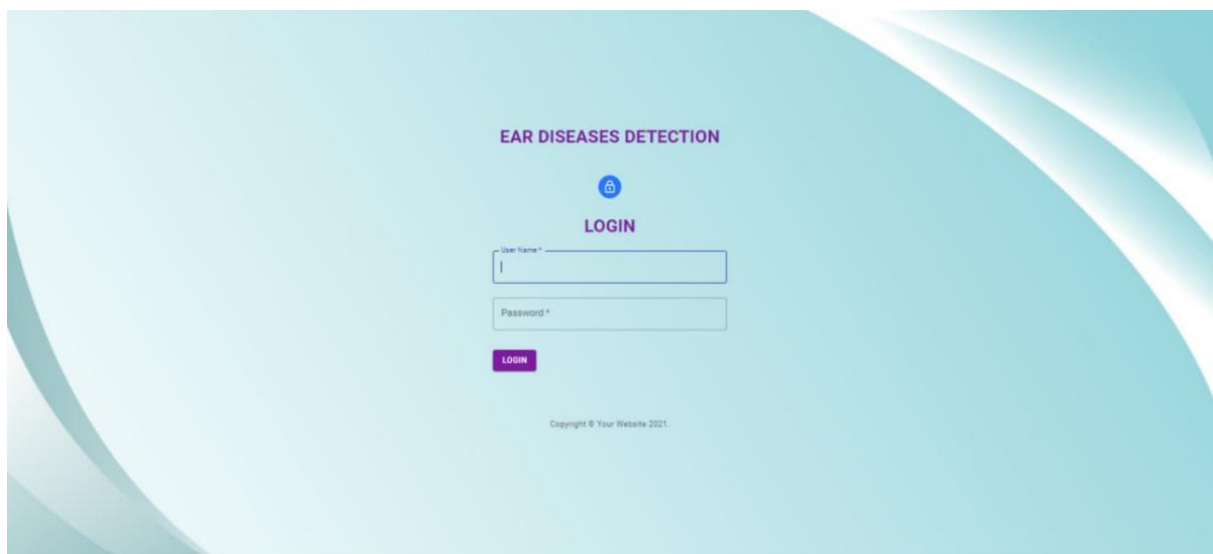


Fig: 5.18 Login page of web application

Fig: 5.19 shows the home page of the web application.

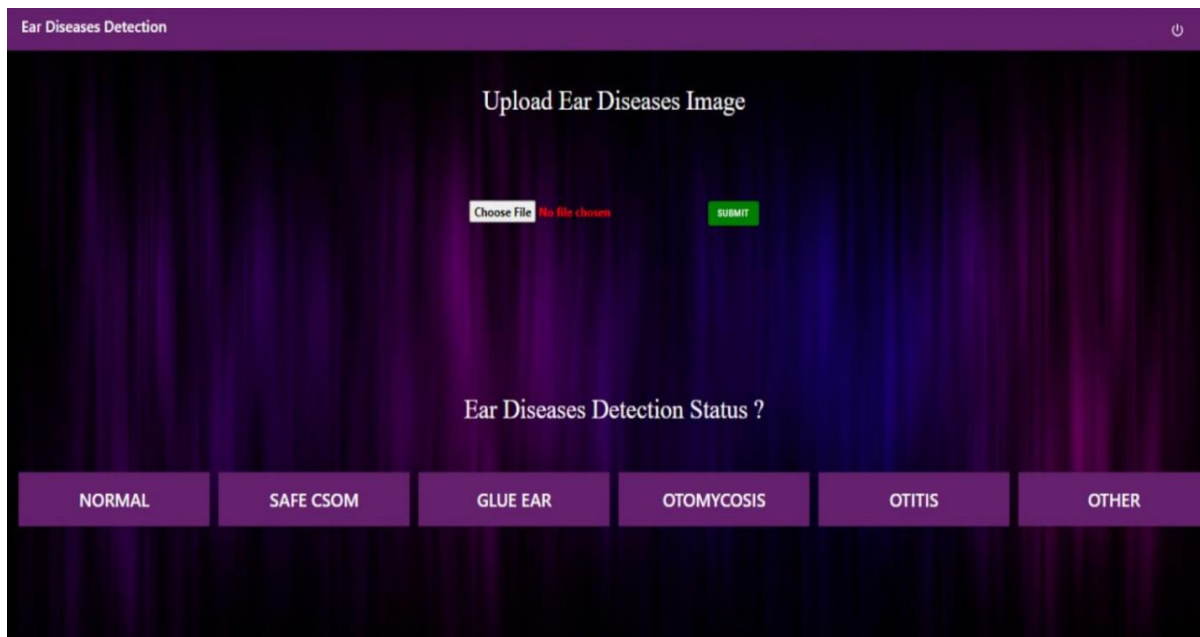


Fig: 5.19 Home page of web application

In the web page there is an option for choosing the image from the file and after selecting the image it can be uploaded. After uploading the image, it can be submitted. Below there are six options showing the six diseases that can be predicted.

On uploading an image of an ear infected from glue ear image the system effectively identifies it as shown in the Fig: 5.20.

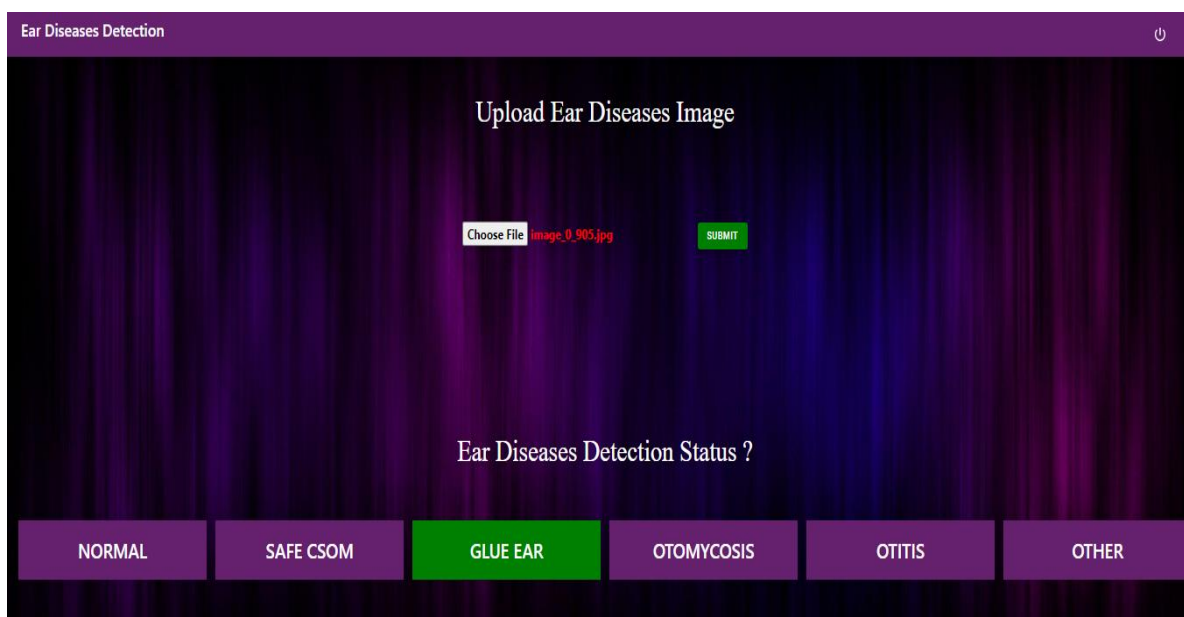


Fig: 5.20 Glue ear diagnosis

On uploading an image of an ear infected from otomycosis image the system effectively identifies it as shown in the Fig: 5.21.



Fig: 5.21 Otomycosis diagnosis

On uploading an image of an ear infected from safe CSOM image the system effectively identifies it as shown in the Fig: 5.22.

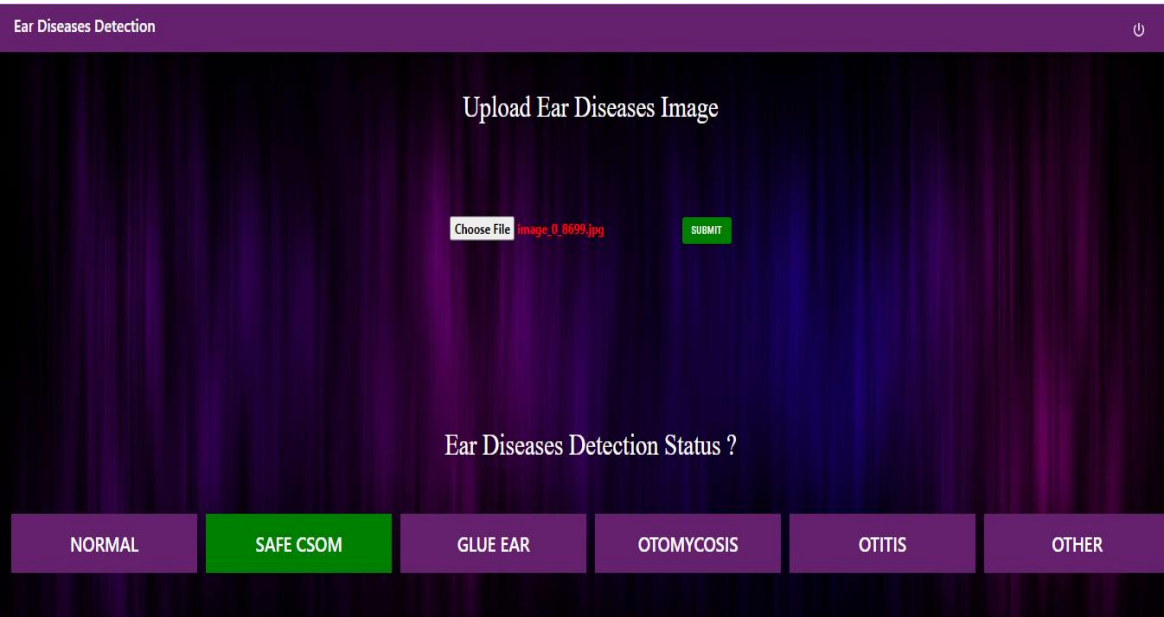


Fig: 5.22 Safe CSOM diagnosis

On uploading an image of an ear infected from acute otitis media image the system effectively identifies it as shown in the Fig: 5.23.

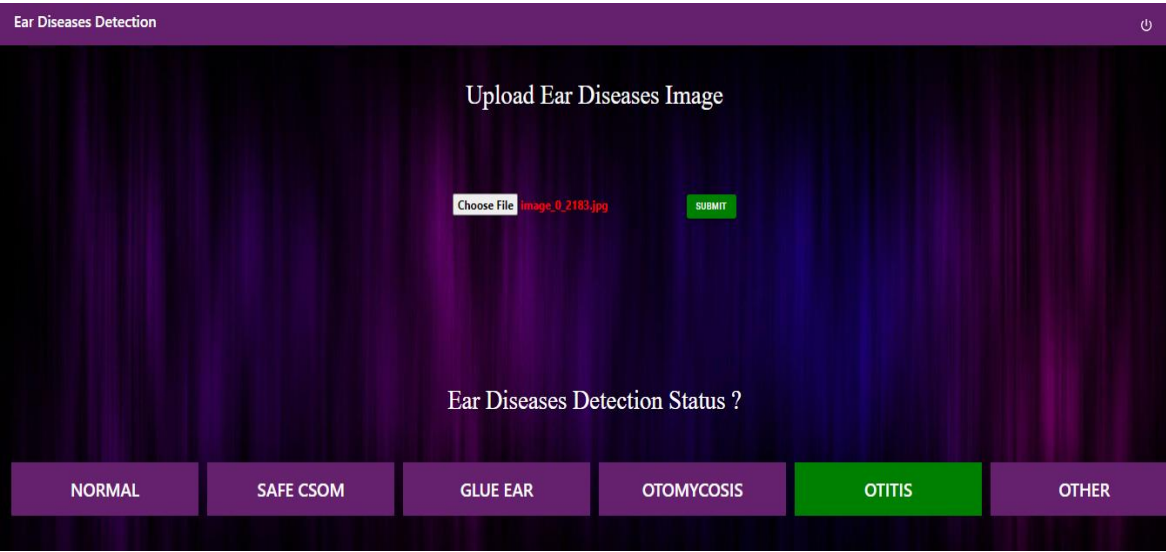


Fig: 5.23 Acute otitis diagnosis

On uploading an image of an ear infected from other types of ear infection image the system effectively identifies it as shown in the Fig: 5.24.



Fig: 5.24 Other infection diagnosis

On uploading an image of an ear infected from normal image the system effectively identifies it as shown in the Fig: 5.25.

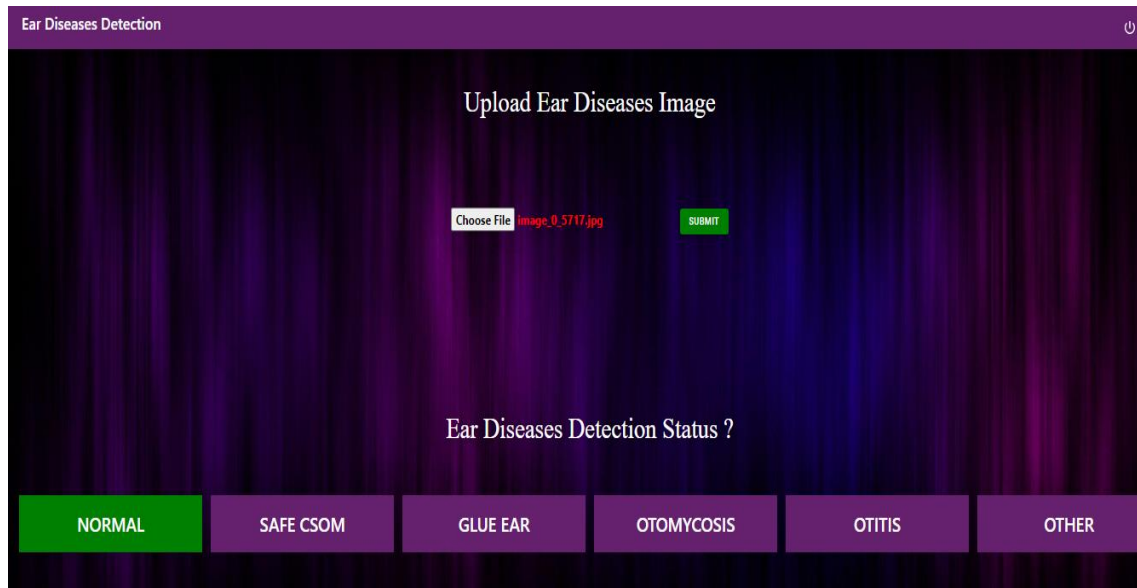


Fig: 5.25 Normal ear diagnosis

5.2 DISCUSSION

Pinar Uluer, et al., (2020) developed a successful Robot Companion for the Audiology Rehabilitation process. The aim of this study is to create a useful robot companion for audiology recovery and to test the device with children who are deaf or hard of hearing and facial data is combined with surveys. to assess the children's perceptions of the robot and the behaviour collection. Using ensemble deep learning and a massive otoendoscopy image database, Dongchul Cha, et al., (2019) created an algorithm for automatic ear condition diagnosis. This study presents a machine learning model for automatically diagnosing ear disease based on a large database of otoendoscopic photographs collected in a clinical environment. To classify six types of ear disorders based on the eardrum and external auditory canal characteristics. To differentiate ear abnormalities from 2D images, Rami R Hallac, et al., (2019) used a convolutional neural network. In this study, CNN was used to detect ear deformity in 2D images automatically. Photographs of patients with and without ear deformities were taken in the studio to assess care efficiency. Deep learning algorithm for otomatch content-based eardrum image retrieval. Content-based image retrieval (CBIR) scheme (OtoMatch) for normal, middle ear effusion, and tympanostomy tube conditions using eardrum images taken

with a digital otoscope. It explains how to transform any classification-trained convolutional neural network into an image retrieval model. Otoendoscopic images of ear deformities were taken, trained in deep learning, and the accuracy was less than 87.08 percent. We developed deep learning architecture by modifying AlexNet, mini GoogLeNet, and LeNet architecture to effectively detect and diagnose the existence of five different ear diseases, such as acute otitis media, glue ear, safe CSOM, otomycosis, other infection and to verify whether the ear is normal or not, without the need for multiple consultations from different doctors.

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 SUMMARY

In this project, we develop and apply three different modified deep learning architectural namely, AlexNet, mini GoogLeNet and LeNet to detect and identify the presence of different types ear disease. The first process is the collection of images for the different types ear disease that include acute otitis media, glue ear, safe CSOM, otomycosis, normal ear and other infection. After this we modify it by preprocessing and augmentation. This helps in easy identification of the images by the neural network. Then we trained the dataset using different architecture. After this we will be comparing how the three different architecture has trained the images. These images are stored in a file as prediction and thus the model is integrated, these are back-end process and is written using python. We create a web application in the front-end using JavaScript. The input images are given to the front end and is converted to array and then given to the back end. In the back end it is again converted into image and it will be compared with the predictions. The predicted output is displayed on the screen. The python library is used to create an application programming interface (API) between the front end and the back end. The trained architecture has effectively predicted the disease and using the web application, input scanned image will give the output of different types of ear diseases, saving lot of time and money invested by patients.

6.2 CONCLUSION

Using a web application built in ReactJS, this project is used to detect the prevalence of ear diseases and provide preventative steps. This further aids in ensuring effective care at a low cost, reducing the time taken to diagnose ear infections in the current state. This initiative decreases the time taken for manual classification and removes the human error rate. Among the three architectures compared, AlexNet emerged with higher accuracy. This initiative has a bright future ahead of it, where manual prediction can be easily transformed into computerized prediction at a low cost.

REFERENCES

1. Ahila Priyadharshini R, Arivazhagan S, Arun M, "A deep learning approach for person identification using ear biometrics", Appl Intell, 2020.
2. Akshansh Sharma, Firoj Khan, Deepak Sharma, Dr. Sunil Gupta, "Python programming, Real-time programming. Efficient and fast programming", International Journal of Innovative Research in Technology, Volume 6, Issue 12, 2021, Pp 115 - 118.
3. Alexander Selvikvåg Lundervold, Arvid Lundervold, "An overview of deep learning in medical imaging focusing on MRI", Zeitschrift für Medizinische Physik, Volume 29, Issue 2, 2019, Pages 102-127.
4. Alshazly H, Linse C, Barth E, Martinetz T, "Deep convolutional neural networks for unconstrained ear recognition", IEEE Access, Vol. 8, 2020, pp. 170295-170310.
5. Cesare Alippi, Simone Disabato, Manuel Roveri, "Moving convolutional neural networks to embedded systems: The AlexNet and VGG-16 Case", 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2018, pp. 212-223.
6. Darvamangala DR, Kulkarni RV, "Convolutional neural networks in medical image understanding: A Survey", Evol Intel, 2020.
7. Dogucan Yaman, Fevziye Irem Eyiokur, Nurdan Sezgin, Hazım Kemal Ekenel, "Age and gender classification from ear images", IEEE IWBF, 2018.
8. Dongchul Cha, ChongwonPae, Si-BaekSeong, Jae YoungChoi, Hae-JeongPark, "Automated diagnosis of ear disease using ensemble deep learning with a big otoendoscopy image database", Lancet, Volume 45, 2019, Pages 606-614.
9. Farhana Sultana A, Sufian, Paramartha Dutta, "Advancements in image classification using convolutional neural network", 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), 2019.
10. Fernando Almeida, José Monteiro, "The role of responsive design in web development", Webology, Volume 14, Number 2, December, 2017.

11. Hallac RR, Lee J, Pressler M, James R. Seaward, Alex A. Kane, "Identifying ear abnormality from 2D photographs using convolutional neural networks", Sci Rep 9, 18198, 2019.
12. Heryandi A, "Developing Application Programming Interface (API) for student academic activity monitoring using Firebase Cloud Messaging (FCM)," IOP Conf. Ser. Mater. Sci. Eng., Vol. 407, No. 1, 2018.
13. Iskhakova AO, Wolf DA, Galin RR, Mamchenko MV, "1-D Convolutional neural network based on the inner ear principle to automatically assess human's emotional state", E3S Web of Conferences 224, TPACEE-2020, 2020.
14. Ker Justin & Wang, Lipo & Rao, Jai & Lim, Tchoyoson, "Deep learning applications in medical image analysis", 2017, IEEE Access. PP. 1-1. 10.1109/ACCESS.2017.2788044.
15. Kwangwon Sun, Sukyoung Ryu, "Analysis of JavaScript programs: challenges and research trends", ACM Computing Surveys, Volume 50, Issue 4, article 59, 2017.
16. Marwin Alejo, Cris Paulo Hate, "Unconstrained ear recognition through domain adaptive deep learning models of convolutional neural network", International Journal of Recent Technology and Engineering (IJRTE), ISSN: 2277-3878, Volume-8, Issue-2, 2019.
17. Md Zahangir Alom, Tarek Taha, Chris Yakopcic, Stefan Westberg, "The history began from AlexNet: A comprehensive survey on deep learning approaches", (2019).
18. Sanchit Aggarwal, "Modern web-development using ReactJS", International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 5, Issue 1, 2018, pp. 133-137.
19. Satya P Singh, Lipo Wang, Sukrit Gupta, Haveesh Goli, Parasuraman Padmanabhan, Balázs Gulyás, "3D Deep learning on medical images: A Review", Sensors Journal, Volume 4, 5097, 2020.
20. Seda Camalan, Muhammad Khalid Khan Niazi, Aaron C. Moberly, Theodoros Teknos, Garth Essig, Charles Elmaraghi, Nazhat Taj-Schaal, Metin N. Gurcan, "Otomatch: Content-based eardrum image retrieval using deep learning", Plos One 15(5), Journal. pone. 0232776, 2020.

21. Sharma, Ankush & Sharma, Aakanksha. "Introduction to HTML (Hyper Text Markup Language) - A review paper", International Journal of Science and Research (IJSR), 2018, 1337-1339.
22. Shorten C., Khoshgoftaar T.M., "A survey on image data augmentation for deep learning", Journal Big Data 6, 60 (2019).
23. Sudhakar Sengan, Arokia Jesu Prabhu Lazar, Ramachandran V, Priya V, Logesh R, Subramaniaswamy V, "Images super-resolution by optimal deep AlexNet architecture for medical application: A Novel DOCALN", Journal of Intelligent and Fuzzy Systems, Vol. 39, no. 6, 2020, pp. 8259-8272.
24. Venkatesh Duppada, Sushant Hiray, "Ensemble of deep neural networks for acoustic scene classification", Detection and Classification of Acoustic Scenes and Events, 2017.
25. Xin M., Wang Y., "Research on image classification model based on deep convolution neural network", Journal Image Video Proc, 2019, 40 (2019).

APPENDICES

USAGE

```
# python train.py --dataset dataset1 --model pn.model --output output.png
```

```
# import the necessary packages
```

```
from sklearn.preprocessing import LabelBinarizer
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import classification_report
```

```
from headers.preprocessing import ImageToArrayPreprocessor
```

```
from headers.preprocessing import AspectAwarePreprocessor
```

```
from headers.datasets import SimpleDatasetLoader
```

```
from headers.nn.conv import FCHeadNet
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
from tensorflow.keras.optimizers import RMSprop
```

```
from tensorflow.keras.optimizers import SGD
```

```
from tensorflow.keras.optimizers import Adam
```

```
from tensorflow.keras.applications import VGG16
```

```
#from headers.nn.conv import AlexNet
```

```
from tensorflow.keras.layers import Input
```

```
from tensorflow.keras.models import Model
```

```
from imutils import paths
```

```
import numpy as np
```

```
import argparse
```

```
import os
```

```
import matplotlib.pyplot as plt
```

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import BatchNormalization
```

```
from tensorflow.keras.layers import Conv2D
```

```
from tensorflow.keras.layers import MaxPooling2D
```

```
from tensorflow.keras.layers import Activation
```

```
from tensorflow.keras.layers import Flatten
```

```
from tensorflow.keras.layers import Dropout
```

```
from tensorflow.keras.layers import Dense
```

```

from tensorflow.keras.regularizers import l2
from tensorflow.keras import backend as K
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True,
                help="path to input dataset")
ap.add_argument("-m", "--model", required=True,
                help="path to output model")
ap.add_argument("-o", "--output", required=True,
                help="path to output graph model")
args = vars(ap.parse_args())

# construct the image generator for data augmentation
aug = ImageDataGenerator(rotation_range=30, width_shift_range=0.1,
                        height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
                        horizontal_flip=True, fill_mode="nearest")

class AlexNet:
    @staticmethod
    def build(width, height, depth, classes, reg=0.0002):
        # initialize the model along with the input shape to be
        # "channels last" and the channels dimension itself
        model = Sequential()
        inputShape = (height, width, depth)
        chanDim = -1

        # if we are using "channels first", update the input shape
        # and channels dimension
        if K.image_data_format() == "channels_first":
            inputShape = (depth, height, width)
            chanDim = 1

        # Block #1: first CONV => RELU => POOL layer set
        model.add(Conv2D(96, (11, 11), strides=(4, 4),
                        input_shape=inputShape, padding="same",

```

```

        kernel_regularizer=l2(reg)))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
model.add(Dropout(0.25))

```

```

# Block #2: second CONV => RELU => POOL layer set
model.add(Conv2D(256, (5, 5), padding="same",
        kernel_regularizer=l2(reg)))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
model.add(Dropout(0.25))

```

RELU

```

# Block #3: CONV => RELU => CONV => RELU => CONV =>
model.add(Conv2D(384, (3, 3), padding="same",
        kernel_regularizer=l2(reg)))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(384, (3, 3), padding="same",
        kernel_regularizer=l2(reg)))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(256, (3, 3), padding="same",
        kernel_regularizer=l2(reg)))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
model.add(Dropout(0.25))

```

```

# Block #4: first set of FC => RELU layers
model.add(Flatten())
model.add(Dense(4096, kernel_regularizer=l2(reg)))

```

```

        model.add(Activation("relu"))
        model.add(BatchNormalization())
        model.add(Dropout(0.5))

        # Block #5: second set of FC => RELU layers
        model.add(Dense(4096, kernel_regularizer=l2(reg)))
        model.add(Activation("relu"))
        model.add(BatchNormalization())
        model.add(Dropout(0.5))

        # softmax classifier
        model.add(Dense(classes, kernel_regularizer=l2(reg)))
        model.add(Activation("softmax"))

        # return the constructed network architecture
        return model

# grab the list of images that we'll be describing, then extract
# the class label names from the image paths
print("[INFO] loading images...")
imagePaths = list(paths.list_images(args["dataset"]))
classNames = [pt.split(os.path.sep)[-2] for pt in imagePaths]
classNames = [str(x) for x in np.unique(classNames)]

# initialize the image preprocessors
aap = AspectAwarePreprocessor(256, 256)
iap = ImageToArrayPreprocessor()

# load the dataset from disk then scale the raw pixel intensities to
# the range [0, 1]
sdl = SimpleDatasetLoader(preprocessors=[aap, iap])
(data, labels) = sdl.load(imagePaths, verbose=100)
data = data.astype("float") / 255.0

# partition the data into training and testing splits using 75% of

```

```

# the data for training and the remaining 25% for testing
(trainX, testX, trainY, testY) = train_test_split(data, labels,
                                                  test_size=0.25, random_state=42)

# convert the labels from integers to vectors
trainY = LabelBinarizer().fit_transform(trainY)
testY = LabelBinarizer().fit_transform(testY)

model = AlexNet.build(256, 256, 3, 6)

print("[INFO] compiling model...")
opt = Adam(lr=0.001)
model.compile(loss="categorical_crossentropy", optimizer=opt,
              metrics=["accuracy"])

print("[INFO] training head...")
model.fit_generator(aug.flow(trainX, trainY, batch_size=32),
                    validation_data=(testX, testY), epochs=40,
                    steps_per_epoch=len(trainX) // 32, verbose=1)

# evaluate the network after initialization
print("[INFO] evaluating after initialization...")
predictions = model.predict(testX, batch_size=32)
print(classification_report(testY.argmax(axis=1),
                            predictions.argmax(axis=1), target_names=classNames))
print("[INFO] re-compiling model...")
opt = SGD(lr=0.0001)
model.compile(loss="categorical_crossentropy", optimizer=opt,
              metrics=["accuracy"])

```

```

print("[INFO] fine-tuning model...")
H = model.fit_generator(aug.flow(trainX, trainY, batch_size=32),
                        validation_data=(testX, testY), epochs=20,
                        steps_per_epoch=len(trainX) // 32, verbose=1)

for key in H.history.keys():
    print(key)
# evaluate the network on the fine-tuned model
print("[INFO] evaluating after fine-tuning...")
predictions = model.predict(testX, batch_size=32)
print(classification_report(testY.argmax(axis=1),
                            predictions.argmax(axis=1), target_names=classNames))

# save the model to disk
print("[INFO] serializing model...")
model.save(args["model"])

plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, 20), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, 20), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, 20), H.history["acc"], label="train_acc")
plt.plot(np.arange(0, 20), H.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy on EAR DISEASES")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.savefig(args["output"])

```