A
Project Report
On
"Biometric Ear Recognition"

(CE416 - Software Project Major)

Prepared by

Vignesh Patel (14CE110)

Under the Supervision of

Dr. Jitendra P Chaudhari

Submitted to

Charotar University of Science & Technology (CHARUSAT)
for the Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology (B.Tech.)
in Computer Engineering (CE)
for 8th semester B.Tech

Submitted at

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
Accredited with Grade A by NAAC
Accredited with Grade A by KCG

U & P U. PATEL DEPARTMENT OF COMPUTER ENGINEERING
(NBA Accredited)
Chandubhai S. Patel Institute of Technology (CSPIT)
Faculty of Technology & Engineering (FTE), CHARUSAT
At: Changa, Dist: Anand, Pin: 388421.

April, 2018

DECLARATION BY THE CANDIDATE

I hereby declare that the project report entitled "**Biometric Ear Recognition**" submitted by me to Chandubhai S. Patel Institute of Technology, Changa in partial fulfilment of the requirement for the award of the degree of **B.Tech** in Computer Engineering, from U & P U. Patel Department of Computer Engineering, CSPIT/FTE, is a record of bonafide CE416 Software Project Major (project work) carried out by me under the guidance of **Dr. Jitendra P Chaudhari**. I further declare that the work carried out and documented in this project report has not been submitted anywhere else either in part or in full and it is the original work, for the award of any other degree or diploma in this institute or any other institute or university.

Vignesh Patel (14CE110)

Dr. Jitendra P Chaudhari
Associate Professor
CHARUSAT space research and technology center,
V. T. Patel Department of Electronics & Communication Engineering,
CSPIT/FTE, CHARUSAT-Changa.

ABSTRACT

Ear recognition is gaining on popularity in recent years. The human ear is neither affected by expressions like faces are nor do need closer touching like finger-prints do. In this paper, a novel algorithm was proposed to do ear recognition using deep convolutional neural network and provide a visualization of the learned network.

Ear location on a face is relative stable, when toward the side of the face, ear is more useful. Unlike face, ear biometric has been utilized to recognize identical twins because of its discriminant characteristics. In addition, image segmentation of ear from facial region is less challenging than face because of having a predictable background. Deep convolutional neural network (CNN) has produced great advances in solving complex vision recognition problems during recent years, including notable successes in recognizing natural images.

There has been a lot of research on using deep CNN on a variety of challenging machine learning tasks. Also, as we know, deep CNN has already made a big progress in human face identification. All the mentioned achievements above demonstrated that well-constructed deep CNNs are powerful tools to tackle these challenges. Thus, a nature mind is that: How does deep CNN on the recognition of human ear. We are inspired by the powerful ability of deep learning. Not exactly the same as face, the only two changes in ear recognition are pose variation and partial occlusion, especially ear occlusion. So ear recognition may be a less challenging task due to reasons talked above.

ACKNOWLEDGEMENT

We are highly intended to the following personalities who have helped us throughout our project and without their support this project would never have been completed. We are extremely thankful to them. We are thankful to **Dr. Jitendra P Chaudhari and Prof. Chintan Bhatt,** who has provided us resources from the college. For their support and help we heartily thank them. This project would be incomplete without them.

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

1.1. PROJECT SUMMARY

An image from the directory or camera is given as an input to the system. It rectifies the image, performs pre-processing, features extraction, classification, training, and then recognizes the user. It uses convolutional neural networks to learn the unique patterns of earlobe. Backpropagation algorithm is used to learn the network by updating the weights of the nodes and calculating the gradient descent to determine the change required to learn. At the end, after passing through several convolutional and pooling layers, it will detect and recognize the correct person no matter how irrelevant the position of an ear is.

1.2. PURPOSE

- Biometrics is a method of human identification as opposed to identifying humans by their possession of keys or remembering passwords
- Preferred method of identification because ID's and cards can easily be stolen and passwords are likely to be forgotten or shared.
- Discourages fraud.
- Enhances security.

1.3. MOTIVATION

Among various physiological biometric traits, ear has gained much popularity in recent years as it has been found to be a reliable biometrics for human recognition. Use of ear for human recognition has been studied by Iannarelli in 1989. This study has suggested the use of features based on twelve manually measured distances of the ear. It has used 10; 000 ear images to demonstrate the uniqueness of ears and has concluded that ears are distinguishable based on limited number of characteristics. This has motivated researchers in the field of biometrics to look at the use of ear for human recognition. Analysis of the decidability index (which measures the separation between genuine and imposter scores for a biometric system) also

suggests the uniqueness of an individual ear. It has been found that the decidability index of the ear is in an order of magnitude greater than that of face, but not as large as vii that of iris. Below is a list of characteristics which make ear biometrics a popular choice for human recognition.

1. Ear is found to be very stable. Medical studies have shown that major changes in the ear shape happen only before the age of 8 years and after that of 70 years. Shape of the ear is found to be stable for rest of the life.
2. Ear is remarkably consistent and does not change its shape under expressions like face.
3. Color distribution of the ear is almost uniform.
4. Handling background in case of ear is easy as it is very much predictable. An ear always remains fixed at the middle of the profile face.
5. Ear is unaffected by cosmetics and eye glasses.
6. Ear is a good example of passive biometrics and does not need much cooperation from the subject. Ear data can be captured even without the knowledge of the subject from a distance.
7. Ear can be used in a standalone fashion for recognition or it can be integrated with the face for enhanced recognition.

Even though ear has so many rich characteristics as compared to other biometrics, the performance of 2D or 3D ear recognition techniques is found to be low and hence it has kept it away from being widely used.

1.4. OBJECTIVE

- To recognize the position of ear, inappropriate of the position.
- To augment the data i.e. to increase the dataset.
- Smart recognition.

1.5. SCOPE

- The system will be designed to store and retrieve multiple images in real time data.
- Ear images can be easily taken from a distance and without knowledge of the examined person.
- More specifically, this system is highly desired to both consumers and computational photography and computer vision applications.
- Ear is also smaller than face, which means that it is possible to work faster and more efficiently with the images with the lower resolution.

1.6. TECHNOLOGY AND LITERATURE REVIEW

The whole system is implemented on Python using Spyder Notebook. is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder is extensible with plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint and Rope. It is available cross-platform through Anaconda, on Windows.

Literature survey:

Table 1.1. Literature survey 1

1. Ear Recognition Based on Deep Convolutional Network[1]	
Author:	Liang Tian, Zhichun Mu
Published at:	2016 9th International Congress on Image and Signal Processing, Bio-Medical Engineering and Informatics
Link:	ieeexplore.ieee.org/document/7852751/

Summary:

We recognize ear identification using deep convolutional neural network. This is an end to end learning system. In the training process, we can obtain all the parameters by forward propagation and error back propagation. The system can get both low-level and high-level abstract feature of ears like a feature extractor. The learned feature is more generative than

traditional hand crafted feature. In this paper, what we need to do is to design a network and use proper methods to train it. In our work, a neural network with three convolutional layers, three pooling layers and two fully-connected layers is created. The data set is split into two parts. The first part is training set. And the other part is the test set. Then we take some methods to preprocess the two parts. After that, the final training set is used to train the parameters of the network and the final test set is used to examine the identify ability of the network. The last layer of the network is a soft-max classifier. The output values indicate the probability of each class. The class with the maximum value is considered as the right class.

Table 1.2. Literature Survey 2

2. A Review on Biometrics and Ear Recognition Techniques[2]	
Authors:	Sukhdeep Singh, Dr. Sunil Kumar Singla
Published at:	IJARCSSE Volume 3, Issue 6, June 2013
Link:	ijarcsse.com/Before_August_2017/docs/papers/Volume_3/6.../V3I6-0660.pdf

Summary:

The immediate background of the ear is very predictable it is always located on the side of the head, whereas facial recognition typically requires a controlled background for accurate capture a situation that is obviously not always present. Unlike iris, retina, or fingerprint capture which are contact biometrics, the ear does not require close proximity to achieve capture. Figure shows the common terminology of the external ear. Ears have played a significant role in forensic science. All identification or authentication technologies operate using the following four stages:

a) **Capture**: A sample is captured by the camera during Enrolment and also in identification or verification process, it is taken by any digital camera and easy to use.

b) **Extraction**: by this unique data is extracted from the sample by using different techniques and a template is created by using it on different platforms like matlab and Lab VIEW.

c) **Comparison**: the template is then compared with a sample.

d) **Match/non match**: ear recognition is very complex technology and is largely software based, the system decides if the features extracted from the new samples are a match or a non-match.

Table 1.3. Literature Survey 3

3. A Review Paper on Ear Recognition Techniques: Models, Algorithms and Methods[3]	
Authors:	Khamiss Masaoud, S. Algabary, Khairuddin Omar, Md. Jan Nordin, Siti Norul Huda Sheikh Abdullah
Published at:	Australian Journal of Basic and Applied Sciences
Link:	ajbasweb.com/old/ajbas/2013/January/411-421.pdf

Summary:

During the 1980s in the United States, Iannarelli was instrumental in developing a system for describing specific ear features as useful in the identification of individuals (Iannarelli 1989). An ear recognition system is similar to the typical face recognition system and "consists of five components: image acquisition, preprocessing, feature extraction, model training and template matching" Alvarez *et al*. (2005) used a modified active contour algorithm and Ovoid model for distinguishing the ear. Likewise, Saleh *et al*. (2006) tested a dataset of ear images using several image-based classifiers and feature-extraction methods. Their results indicated an accuracy rate of 76.5% to 94.1% based on their experiments. Furthermore, Islam *et al*. (2008) proposed an ear detection approach based on the AdaBoost (Adaptive Boosting) algorithm (developed by Freund & Schapire 1997), which is believed to be sensitive to noise and certain outliers in the data. The system developed by Islam *et al* (2007), was qualified with rectangular Haar-like attributes and used a dataset that included a variety of races, sexes, appearances, orientations and illuminations. As cited by Boodoo & Subramanian (2009), The data were collected by cropping and synthesizing from several face image databases. The approach is fully automatic, provides 100% detection while tested with 203 non-occluded images and also works well with some occluded and degraded images.

Table 1.4. Literature Survey 4

4. Training CNN with limited training data for ear recognition in the wild.[4]	
Authors:	Ziga Emersic, Dejan Stepec, Peter Peer
Published at:	2017 IEEE 12th International Conference on Automatic Face & Gesture Recognition
Link:	https://arxiv.org/abs/1711.09952

Summary:

We address the problem of training CNNs with limited data and strive to develop an effective CNN based model for ear recognition. Existing approaches to CNN training with small amounts of training data typically include:

1. Metric- learning approaches, where training is performed with image pairs instead of single images.
2. Data augmentation techniques that in addition to geometric and color perturbations of the existing training data also include the generation of synthetic data samples and
3. Using existing CNNs as so called back box feature extractors, on top of which additional classifiers are trained and used for recognition.

Table 1.5. Literature Survey 5

5. Smart Augmentation Learning an Optimal Data Augmentation Strategy.[5]	
Authors:	Joseph Lemley, Shabab Bazrafkan, Peter Corcoran
Published at:	arXiv:1703.08383v1 24 Mar 2017
Link:	https://arxiv.org/abs/1703.08383

Summary:

Many deep learning frameworks can generate augmented data. For example, Keras has a built in method to randomly flip, rotate, and scale images during training but not all of these methods will improve performance and should not be used “blindly”. Manual augmentation techniques such as rotating, flipping and of noise to the data samples, are described in depth in and which

attempt to measure the performance gain given by specific augmentation techniques. They also provide a list of recommended data augmentation methods. In 2014, Srivastava et al. introduced the dropout technique, aiming to reduce overfitting, especially in cases where there is not enough data. Dropout works by temporarily removing a unit (or artificial neuron) from the Artificial Network and any connections to or from that unit.

CHAPTER 2. PROJECT MANAGEMENT

2.1. PROJECT PLANNING

2.1.1. Project Development Approach and Justification

Agile model is used for the development of this project. We had a short period of time to develop the project. So to get flexibility and have ease in managing and developing the project we used agile model.

Phase 1 – Literature survey and analysis. We surveyed the requirements for the project and did analysis of various research papers and survey papers for the project.

Phase 2 - Simulation. We simulated the system based on the analysis and survey.

Phase 3 - Implementation. We implemented the various techniques, methods and algorithms for the system.

Phase 4 - Testing. We have tested all the modules of the project.

Phase 5 – Documentation. We have prepared required documents for the project.

2.1.2. Project Effort and Time, Cost Estimation

Estimation technique used: The project size is a measure of the problem complexity in terms of the effort and time required to develop the product. Currently, two metrics are popularly being used to estimate size: Line of code (LOC) and function point (FP).

Parameters	Count		Simple	Average	Complex		Total
No. of user Input	1	X	3	4	6	=	3
No. of user Output	3	X	4	5	7	=	12
No. of Inquires	0	X	3	4	6	=	0
No. of Files	1	X	7	10	15	=	15
External Interface	0	X	5	7	10	=	0

Complexity Weight Factor:

Sr. No.	Factors	Weights
1.	Does the system require reliable backup and recovery?	**0**
2.	Are data communication required?	**0**
3.	Are there distributed processing functions?	**5**
4.	Is performance critical?	**4**
5.	Will the system run in an existing, heavily utilized operational environment?	**4**
6.	Does the system require online data entry?	**0**
7.	Does the on-line data entry require the input transactions to be built over multiple screens or operation?	**0**
8.	Are the master file updated on-line	**0**
9.	Are the inputs, outputs, files, or inquiries complex?	**2**
10.	Is the internal processing complex?	**5**
11.	Is the code designed to be reusable?	**4**
12.	Are conversion and installation included in the design?	**3**
13.	Is the system designed for multiple installations in different organizations?	**4**
14.	Is the application designed to facilitate change and ease of use by the user?	**4**

FP Count:

FP = count total*[0.65+0.01 *$\sum$(Di)]

FP = 30* [0.65+0.01*35]

FP = 30

Function Point is: 30

Line of code (LOC) = FP*53 = 30*53 = 1590

KLOC = 1.590

Weight	Degree Of Influence
0	No Influence
1	Incidental
2	Moderate
3	Average
4	Significant
5	Essential

Software Project Type

Type	a_b	b_b	c_b	d_b
Organic	**2.4**	**1.05**	**2.5**	**0.38**
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Effort = a_b*(KLOC) ^ b_b

= 2.4*(1.590) ^1.05

= 3.90 PM

Tdev = c_b * (Effort) ^d_b

= 2.5*(3.90) ^0.38

= 4.2 Months

2.1.3. Roles and Responsibilities

Table 2.5. Roles and Responsibilities

Responsibilities	**Roles**
Literature survey and Analysis	Vignesh Patel
Simulation	Vignesh Patel
Implementation	Vignesh Patel
Testing	Vignesh Patel
Documentation	Vignesh Patel

2.2. PROJECT SCHEDULING (GANTT CHART/PERT/NETWORK CHART)

2.2.1 Task Network Representation

	Task Mode	Task Name	Duration	Start	Finish	Predeces	Resource Names
1		**Requirement gathering**	**13 days**	**Mon 01-01-18**	**Wed 17-01-18**		**Laptop,Internet,Vignesh**
2		Analysis	3 days	Mon 01-01-18	Wed 03-01-18		Vignesh
3		Study of existing system	0 days	Thu 04-01-18	Thu 04-01-18		Vignesh,Internet
4		**Planning**	**15 days**	**Thu 18-01-18**	**Wed 07-02-18**	**1**	**Laptop,Internet,Vignesh**
5		Literature survey	9 days	Thu 18-01-18	Tue 30-01-18		Vignesh
6		Resource allocation	1 day	Wed 31-01-18	Wed 31-01-18		Vignesh
7		Time	5 days	Thu 01-02-18	Wed 07-02-18		
8		Design	13 days	Thu 08-02-18	Mon 26-02-18	4	**Laptop,Internet,Vignesh**
9		Coding	25 days	Tue 27-02-18	Mon 02-04-18	8	Vignesh,Internet
10		Testing	10 days	Thu 22-03-18	Wed 04-04-18		Internet,Vignesh
11		Documentatiuon	5 days	Tue 03-04-18	Mon 09-04-18	9	Vignesh

Figure 2.1. Task Network Representation

2.2.2 Gantt Chart Representation

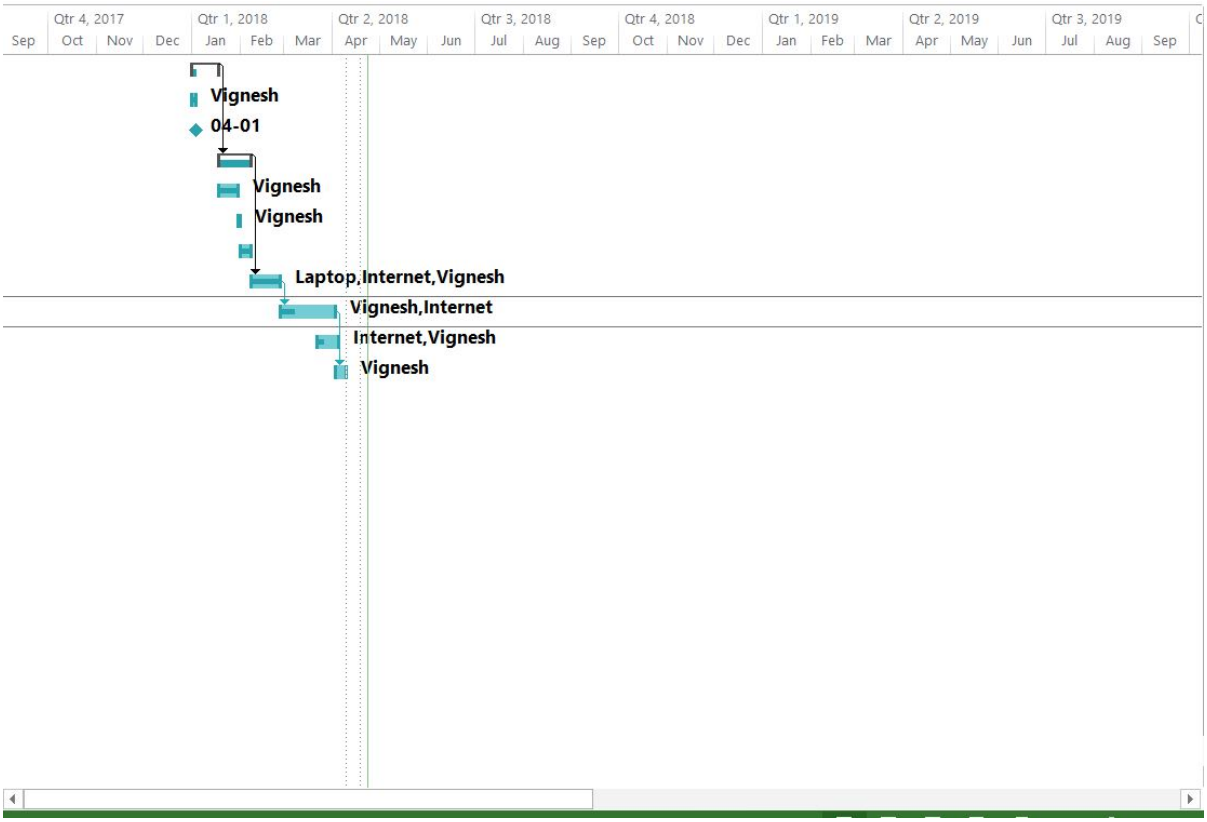

Figure 2.2. Gantt chart Representation

2.2.3 Pert Chart Representation

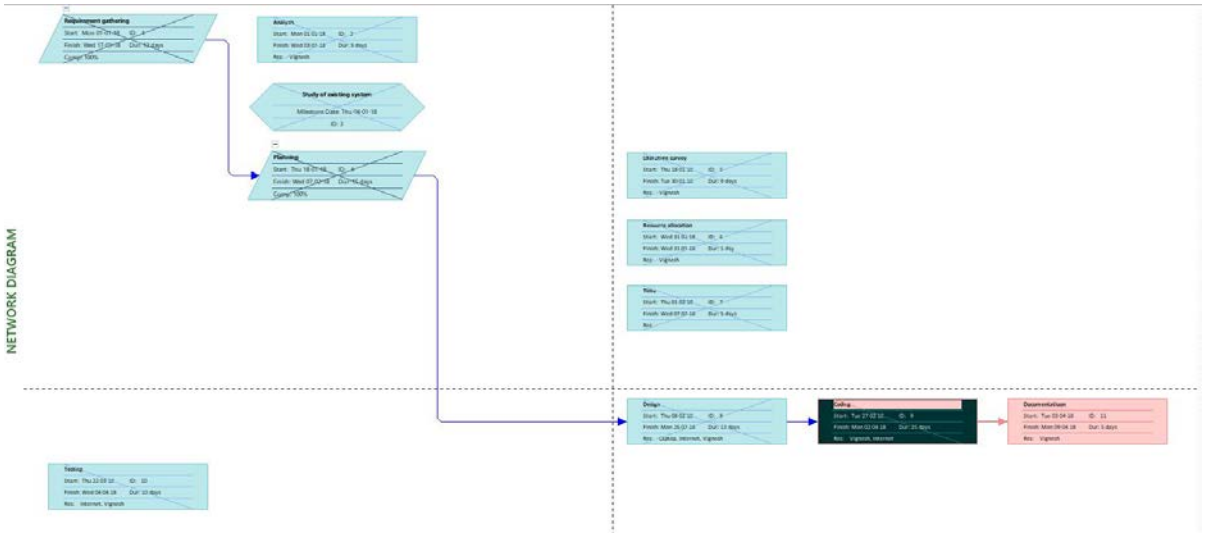

Figure 2.3. Pert chart Representation

CHAPTER 3. SYSTEM REQUIREMENTS STUDY

3.1. USER CHARACTERISTICS

The users have to put the images of their ear on the dataset. System automatically takes images from the dataset, process them and generates the output by showing the accuracy and relevant visualizations.

3.2. HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements

- Minimum 4GB RAM
- Disk Space Required: Minimum 30GB HDD
- 64-bit CPU architecture

Software Requirements

- Anaconda Navigator (x)64bit
- Spyder Notebook v3.2.6 or later(x)64bit
- cuDNN v6.0 (x)64bit
- CUDA v8.0 (x)64bit
- Python 3.5

3.3. ASSUMPTIONS AND DEPENDENCIES

As the CUDA and cuDNN both works only for a specific matching version, we need to make sure the matching version of CUDA and cuDNN are installed, or the environment will be crashed. Also, tensorflow v1.3 is required. The version preceding or succeeding will crash the environment. Moreover, the dataset heavily relies on the number of images per subject. As a result, images are augmented first before feeding into the network.

CHAPTER 4. SYSTEM ANALYSIS

4.1. REQUIREMENTS OF THE SYSTEM

4.1.1. Functional requirements

R 1. Image augmentation

Description: It takes the image as an input and it augments the image i.e. it does the filtering, cropping, rotations, etc. and saves the mages to the disk.

Input: A single image.

Output: Augmented image.

R 2. Feature extraction

Description: It extracts the unique features of the ears from the images that were captured and augmented.

Input: A batch of images from same subject, on initial phase.

Output: A classifier with trained weights.

R 3. Visualization of losses and accuracy and layers

Description: It shows the graphs of training and validation losses and accuracy vs epochs. Also, it also shows the visualization of the intermediate layers of convolutional layers.

Input: Parameters like learning rate, verbose, etc. and for layers, the layer and filter numbers.

Output: 2 graphs of losses and accuracy vs epochs after training.

R 4. Evaluation of the model

Description: the model should be able to make a test by taking one of the validations image and test it. Also, the model should also be able to read a new image from directory and be able to test correctly.

Input: Validation image or new testing image

Output: Predicted class or subject of a respective person.

R 5. Image preprocessing

Description: It preprocesses the images, like resizing, changing to gray from RGB and gives those images as an input.

Input: Three dimensional or Two dimensional image

Output: Preprocessed image with filters applied as per our requirements.

R 6. Constructing confusion matrix

Description: It compares the testing value and predicted value and build the matrix for better understanding of how well our model predicts the subjects.

Input: True labels, predicted labels and values.

Output: 2D matrix along with GUI representation.

4.1.2. Non-functional requirements

- Performance – The performance of the system heavily depends on the GPU and size of the inputted 2D images. The execution will become slow if tensorflow do not detect DLL files of CUDA and cuDNN.
- Reliability - The project will be tested on real time system to make it more reliable and increase time efficiency.
- Availability - The project system will be software based and may not be deployed in different environment.
- Image augmentation – The augmented images should be in such a way that it should contain the unique features of specific subject.
- Portability - Portability of the project system depends upon the software system where it is installed.

4.2. FEASIBILITY STUDY

4.2.1. Does the system contribute to the overall objectives of the organization?

Yes, the system contributes to overall objectives of the organization.

4.2.2. Can the system be implemented using the current technology and within the given cost and schedule constraints?

Yes, the system can be implemented using the current technology, in fact it has been implemented using the current technology and within the given cost and schedule constraints.

4.2.3. Can the system be integrated with other system which are already in place?

At present the system developed is a unique one but if new similar systems are developed in the future, then integration might be possible.

4.3. SYSTEM FLOW DIAGRAM

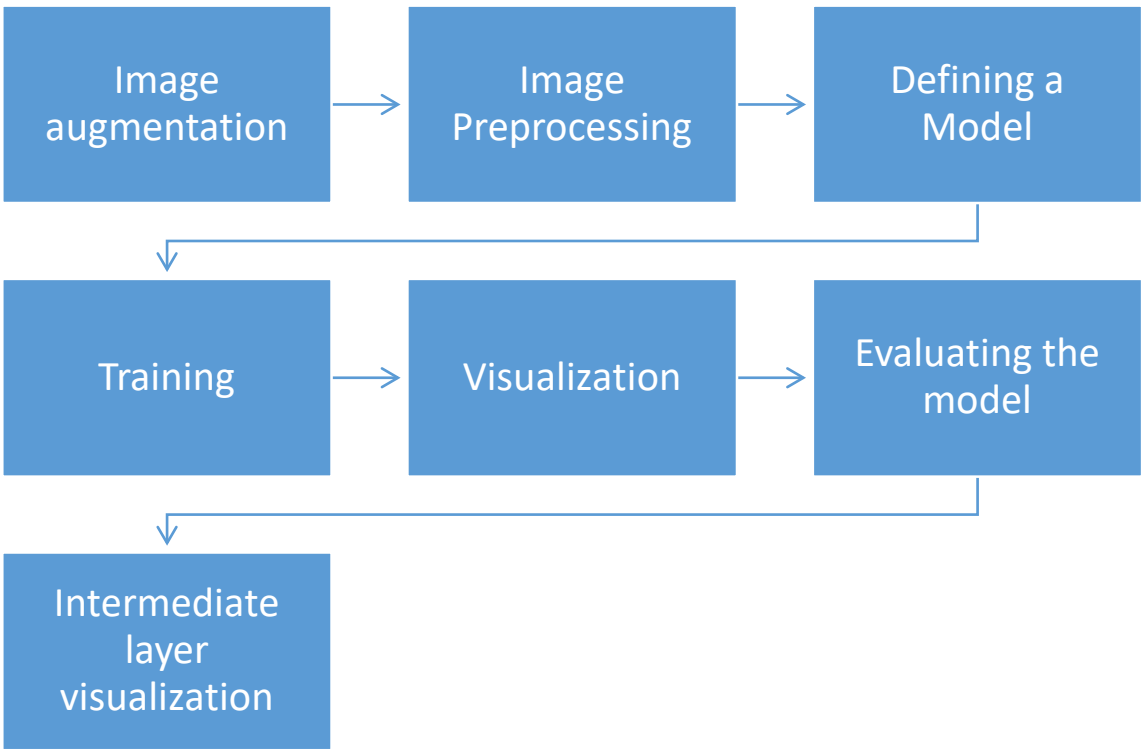

Figure 4.1. System flow diagram

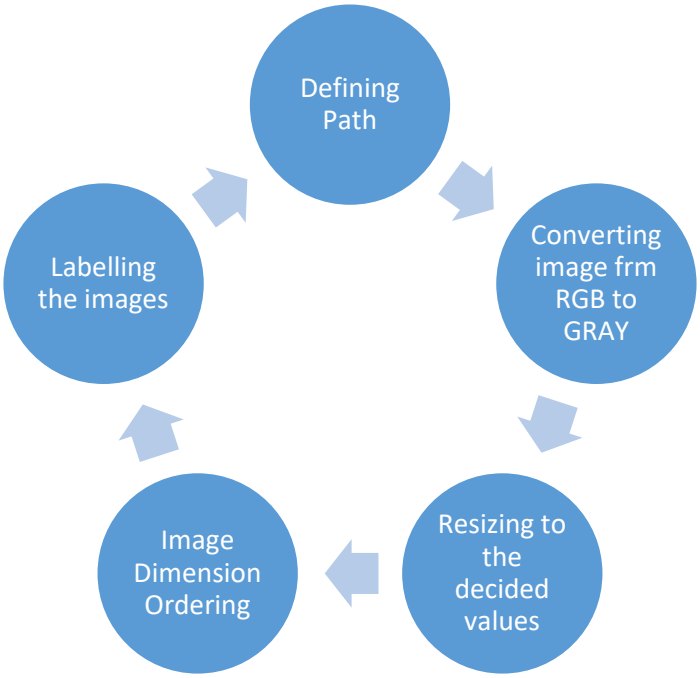

Figure 4.2. Data Preprocessing

PROPOSED APPROACH:

The system is built on anaconda environment and Spyder tool. Python 3.5 is used. There are in total 7 building blocks of the system.

1) IMAGE AUGMENTATION:

The datasets available on the internet consists of maximum 30 images per subject, which is not enough to get 90% accuracy overall. Hence, there are many ways to solve this. One of the simplest way is image augmentation. It is one of easiest ways to achieve ample amount of accuracy. The basic idea is to tune the parameters like rotation angle, ZCA whitening, horizontal-vertical flipping, etc.

FILTER	VALUE	DESCRIPTION
Rotation range	0-45 degrees	If the angle succeeds 45 degrees, the features might not get extracted.
Width shift range	0.2	If it exceeds, the images will not be augmented correctly.
Height shift range	0.2	If it exceeds, the images will not be augmented correctly.
Shear range	0.2	The above value is perfect for the model.
Zoom range	0.2	Zoom range above it will generate blurred images.
ZCA whitening	True	It produces white images which are easy to visualize.
Rescale	1.2	The above value is perfect for the model.

Table 4.1 parameters of image augmentation

2) DATA PREPROCESSING:

The next module is data preprocessing. Normally, in the machine learning model, especially in the CNN, the number of images are usually very large. This results in time consumption and adds load to the CPU or GPU. Hence, images preprocessing is very important.

The size of images is resized and set to 128*128 square sized, so that every images get the same dimension. As a result, image loading becomes a lot faster. The images are changed from 3 channel RGB to 1 channel Grayscale. Number of channels is set to 1 as the images are grayscale.

All the images are stored in a numpy array. The size of numpy array is (1200,128,128,1) where 1200 is the total number of images, 128 is the width and height and 1 is the number of channels. As the backend is tensorflow, the number of channels is set behind. Labels are set because the training type is supervised training. And lastly, data splitting is done to 80% training- 20% testing, just to ensure that more samples are dedicated to train the model.

3) DEFINING A MODEL:

The model definition takes place over here. Here, the model is firstly called by calling sequential() function. Then the layers are defined, which consists of neurons. The system consists of 3 conv layers, 3 activation layers, 2 max pooling layers and 2 dropout layers. The summary of the model is listed below.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 128, 128, 32)	320
activation_1 (Activation)	(None, 128, 128, 32)	0
conv2d_2 (Conv2D)	(None, 126, 126, 32)	9248
activation_2 (Activation)	(None, 126, 126, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 63, 63, 32)	0
dropout_1 (Dropout)	(None, 63, 63, 32)	0
conv2d_3 (Conv2D)	(None, 61, 61, 64)	18496
activation_3 (Activation)	(None, 61, 61, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 30, 30, 64)	0
dropout_2 (Dropout)	(None, 30, 30, 64)	0
flatten_1 (Flatten)	(None, 57600)	0
dense_1 (Dense)	(None, 64)	3686464
activation_4 (Activation)	(None, 64)	0
dropout_3 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 11)	715
activation_5 (Activation)	(None, 11)	0

Table 4.2 model summary

4) TRAINING:

The training of module can be performed in 2 ways. By running the number of epochs, or by running callbacks. It is the function which runs the epochs until it observes that the loss is improved and the values of weights, whose error is minimal, is saved. The model saves an hdf5 file which contains the records of training and validation losses and accuracy. The hist variable stores the accuracy and losses result which can be used to plot the graph by giving as an input to the next module. Batch size is set to 16, and number of epochs are 200.

Training happens over 900 images and 200 images are for validation. 1 epoch takes approximate 5-6 seconds on GPU, whereas 160 images takes 20 seconds to run an epoch. This shows that GPU is favorable.

5) VISUALIZATION:

In this module, the hist variable who had stored losses and accuracy will be used to plot the graphs losses and accuracy. 2 figures will be plotted, at the end, on Y axis are the losses or accuracy and on the X axis are number of epochs. This graph is used to know how well the network is training and we can also know how many number of epochs are beneficial for the system.

6) EVALUATING THE MODEL:

Here, the system takes a random image from the validation set, and predicts the image, and prints the output. Also, the output is in the form of probability values from all the classes. The one with highest value is the prediction of the system.

In the second part, we give entirely new image as an input. Then again preprocessing is done, i.e. the image is resized to 128*128 and grayscale image. Image dimension ordering is done to prevent tensorflow error. Again, the model predicts the image and print the output class.

7) INTERMEDIATE LAYER VISUALIZATION:

The last module is another visualization, which is intermediate visualization layer. Visualization is very important for a model because it is important to know how the image is visualized by a model and also important to know whether the image is trained correctly.

In this module, we can see how the model is visualizing the images that it takes for learning. For that, layer number and filter number is to be mentioned. Then, a 16*16 image is generated which shows the visualization of that specific layer and filter number. To watch what happens over the entire layer, another function is used. The image is automatically saved to the disk and in the current working directory, in png format. We can also save the images manually.

Lastly, it is important to know the individual accuracy of every classes. Confusion matrix is print for that. It is a matrix in which one axis shows true labels and another axis shows predicted labels.

We can also save and load the model using save and load function.

4.4. FEATURES OF THE SYSTEM

The major functionalities of the system are:

- Image augmentation
- Image preprocessing
- Feature extraction
- Visualization of training and validation losses and accuracy vs number of epochs
- Visualization of intermediate layers
- Building confusion matrix

4.5. MAIN MODULES OF THE SYSTEM

Main modules in the system are:

- Augmentation module
- Preprocessing module
- Defining a model module
- Training module
- Visualization module
- Evaluation module

4.6. FUNCTIONS OF THE SYSTEM

- Image augmentation using ImageDataGenerator function (Keras).
- Image preprocessing using cv2 library.
- Splitting the database into training and validation sets.
- Defining a model using Sequential() (Keras).
- Training the model using hist() (Keras).
- Visualizations using matplotlib library
- Visualizations of an intermediate layers.
- Printing the confusion matrix.

4.7. FLOWCHART OF THE SYSTEM

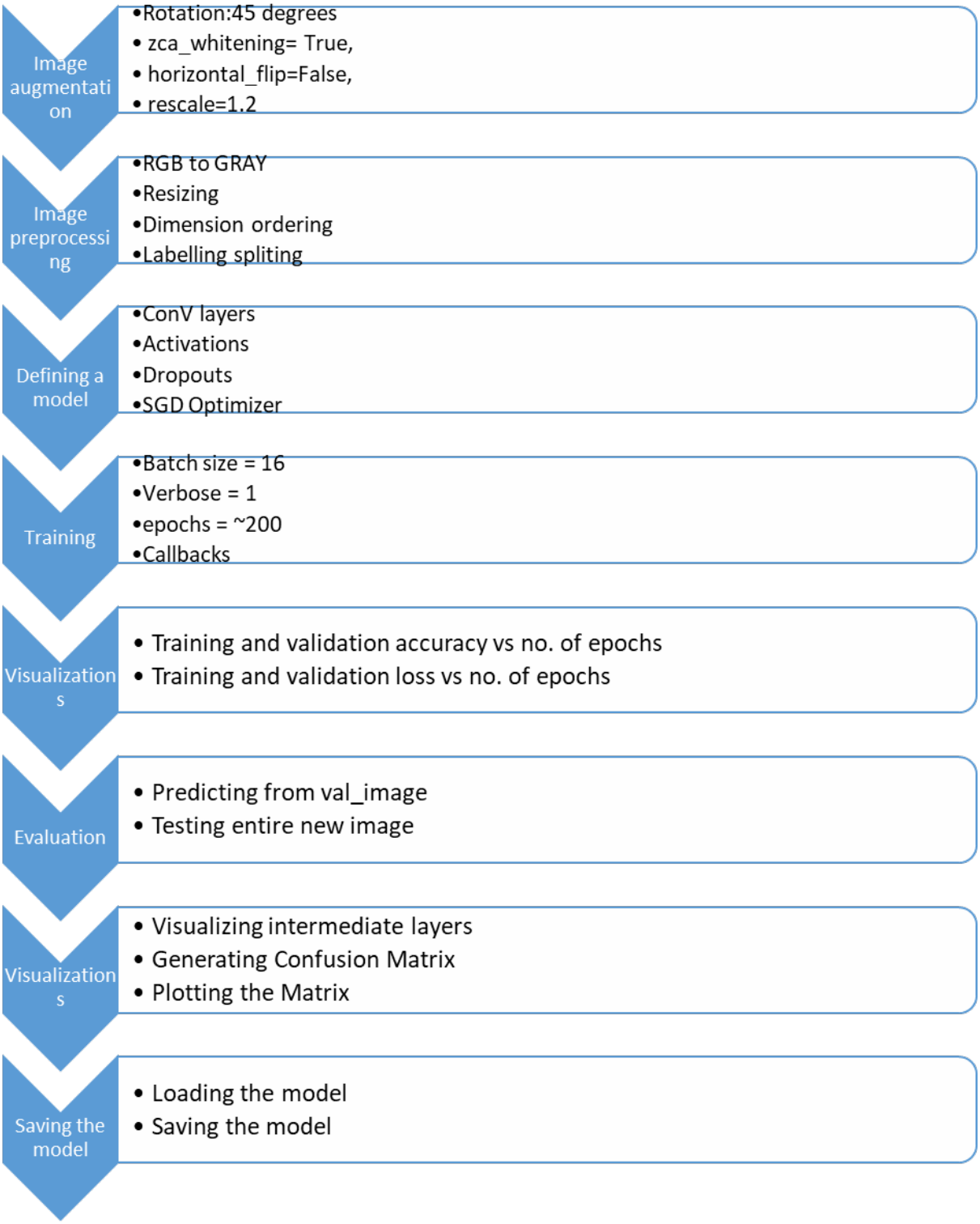

Figure 4.3. Flowchart of the system

4.8. SELECTION OF HARDWARE AND SOFTWARE AND JUSTIFICATION

The whole system is implemented in Python as it is the best programming language for data science and image processing. Recently, with advancements in GPU and tensors, training time significantly reduces to 20x. Hence, the system is proposed to run on GPU and NVIDIA has the software called CUDA which has the configurations to seemingly run on GPU. Also, NVIDIA GPU is recommended as the integrated Intel GPU's do not have the adequate Compute Capability. cuDNN v6.0 works best with CUDA v8.0.

- For the model to run on GPU, CUDA version 8.0 is recommended.
- Along with that, cuDNN v6.0 is preferable as the CUDA version works best and only fits for cuDNN v6.0.
- Syder 3.2.8 is used as it provides scientific python development environment, and it is a Powerful IDE with advanced editing, interactive testing, debugging and introspection features.
- Tensorflow v1.3 is recommended as it will not detect GPU if the version is not matching with CUDA and cuDNN versions.
- Keras library is used as it is an high level API which contains almost every function that is needed in a project.
- Tensorflow is recommended because Theano takes almost 40% more time than tensorflow to train.
- Visualization of validation and training losses and accuracy is done in order to check how well our model is performing.
- Confusion matrix is also calculated to see the comparison between true labels and predicted labels.
- Model is chosen to be running on GPU as it is 20 times faster than CPU.
- A separate environment is build up because the model will not work on python 3.6 and GPU is not supported on tensorflow 1.7 yet. Moreover, there are high chances that the environment might crash if the installation of packages is not done correctly.

CHAPTER 5. SYSTEM DESIGN

5.1. SYSTEM APPLICATION DESIGN

5.1.1. Method pseudo code

```
PATH = os.getcwd()
# Define data path
data_path = PATH + '\\raw'
data_dir_list = os.listdir(data_path)

img_rows=128
img_cols=128
num_channel=1
num_epoch=200

# Define the number of classes
num_classes = 10

img_data_list=[]

for dataset in data_dir_list:
	img_list=os.listdir(data_path+'\\'+ dataset)
	print ('Loaded the images of dataset-'+'{}\n'.format(dataset))
	for img in img_list:
		input_img=cv2.imread(data_path + '\\'+ dataset + '\\'+ img )
		input_img=cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
		input_img_resize=cv2.resize(input_img,(128,128))
		img_data_list.append(input_img_resize)

img_data = np.array(img_data_list)
img_data = img_data.astype('float32')
img_data /= 255
print (img_data.shape)

if num_channel==1:
	if K.image_dim_ordering()=='th':
		img_data= np.expand_dims(img_data, axis=1)
		print (img_data.shape)
	else:
		img_data= np.expand_dims(img_data, axis=4)
		print (img_data.shape)
```

```
else:
        if K.image_dim_ordering()=='th':
                img_data=np.rollaxis(img_data,3,1)
                print (img_data.shape)
num_classes = 11

num_of_samples = img_data.shape[0]
labels = np.ones((num_of_samples,),dtype='int64')

labels[0:120]=1
labels[120:240]=2
labels[240:360]=3
labels[360:480]=4
labels[480:600]=5
labels[600:720]=6
labels[720:840]=7
labels[840:960]=8
labels[960:1080]=9
labels[1080:1200]=10

# convert class labels to on-hot encoding
Y = np_utils.to_categorical(labels, num_classes)

#Shuffle the dataset
x,y = shuffle(img_data,Y, random_state=2)
# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=2)
```

- The above code is written for loading all the images of the folder, preprocessing it, doing dimension ordering, defining classes and labels.
- Then it converts the class labels to on-hot encoding and finally it splits the dataset into training and validation.
- Later on, the model is defined and then training takes place.

CHAPTER 6. IMPLEMENTATION PLANNING

6.1. IMPLEMENTATION ENVIRONMENT

Hardware

- The system has the memory of 8GB RAM because it is sufficient enough to run the model fast.
- The Disk Space has nearly no matter as the system will at most use around 500MB of storage. Still, it depends on the size of the dataset. However, we had 1TB HDD.
- The architecture only matters when we are installing the software. But, the model runs best on 64-bit CPU architecture and so we had 64-bit architecture.
- The GPU is most important when it comes to training and time factor. It is recommended to have a GPU that has minimum compute capability of 5.0. In this system, NVIDIA 4GB GTX 960M GPU is used.

Software

The installation of software components is slight fragile, because if the nonmatching versions of software is installed, it will break the virtual environment and we have to install the environment again and download the libraries. Python 3.5 is used.

- As windows does not have virtual environments where we can modify our libraries and configurations, Anaconda Navigator (x)64bit is used for that purpose.
- Scientific Python Development Environment tool Spyder Notebook v3.2.6 or later(x)64bit is used. It is powerful Python IDE with advanced settings, interactive testing, debugging and introspective features.
- cuDNN v6.0 (x)64bit is used as it works best with CUDA v8.0 and tensorflow 1.3 only works on v6.0.
- CUDA v8.0 (x)64bit is used as it works best with cuDNN v6.0 and tensorflow v1.3 only works on v8.0.

- FIGURES

6.2. PROGRAM/MODULES SPECIFICATION

Dataset[]:

The original dataset is the IIT Delhi database which consists of 493 images from 125 subjects. Each subject has at 3-6 images which are grayscale. However, as the number of images are relatively low for the model to get ample accuracy, the images are augmented to get the sufficient number of images we want.

1. Augmentation module:

It is used to generate multiple images with some filters like cropping, resizing, flipping, ZCA Whitening, etc. applied at the image we want to be augmented. Augmentation is done because the model will over fit if the number of images in the dataset is relatively low. The parameters are to be set manually.

2. Preprocessing module:

It preprocesses the images like resizing all the images to the same dimension, dimension ordering, RGB to GRAY, splitting the dataset, etc. It is done to cope up the time complexity and size reductions.

3. Defining model module:

It builds the neural network architecture where we have to specify the layers, activations and classifiers, like ReLu, Softmax, etc. Moreover, optimizer is also being set in this module.

4. Training module:

This is the main module as the training over the system happens to be here. The system runs the iterations based on the number of epochs mentioned. SGD optimizer is used.

Validation and training losses and accuracy are stored in a variable and then plotted using matplotlib in visualization module.

5. Visualization module:

The important part that comes after training is visualization, as we can know how well our model is performing through it. Training and validation losses and accuracy are plotted against number of epochs so that the number of epochs can be determined and loss and accuracy can be learned. Also, to know what is happening in the layers, visualization of intermediate layers is defined so that we can gain intuition of how the network visualizes the images.

6. Evaluation module:

- After the model is trained, it is evaluated where a random validation image is taken and does the prediction. Moreover, a new testing image is also feed to test and correctly predicts the output. Image can be given as an input, irrelevant of the position of the ear.

6.3. CODING STANDARDS

Below are the coding standards for Python programming language

6.3.1. Use single quotes:

- Use single-quotes for string literals, e.g. 'my-identifier', *but* use double-quotes for strings that are likely to contain single-quote characters as part of the string itself (such as error messages, or any strings containing natural language), e.g. "You've got an error!".
- Single-quotes are easier to read and to type, but if a string contains single-quote characters then double-quotes are better than escaping the single-quote characters or wrapping the string in double single-quotes.

6.3.2. Imports:

- Avoid creating circular imports by only importing modules more specialized than the one you are editing.
- Imports at the top of modules (or the top of functions if they are local).
- Never use from xxx import *.

  Even for importing a lot of names it is better to be able to see where your names come from. Tools like pylint and PyFlakes can help warn you about unused imports.

- I think it looks nicer to put imports on separate lines.

  import xxx should come before from xxx import…

  Standard library before extension modules.

- A single module import per line.

➢ I now prefer this:

  Import os

  Import sys

➢ to this:

  import os,sys

  Not a big deal though.

- Where you are importing lots of names, and targeting Python 2.4 or more recent, you can use the following syntax:

  from namespace import (

name1, name2, name3,

name4, name5, name6)

6.3.3. Line Length:

- Over long lines hurt readability, but so does breaking lines purely because of setting an arbitrary maximum line length. A limit of 79 characters is a good guideline, but readability should come first.
- Longer lines should be wrapped by surrounding expressions in parentheses rather than using \. Long strings can be split across several lines using parentheses:
- Where long expressions go over more than one line, it is helpful to indent lines after the first one. This shows that they belong to each other.

6.3.4. Line Spacings:

- Functions and methods should be separated with two blank lines. Class definitions with three blank lines.
- Normally variable definitions don't need a blank line between them, unless they are logically distinct and you want to separate them. Logical chunks of code within function/method definitions can have a blank line if you want to visually separate them. With these rules the blank lines provide a visual guide to the structure of the code.

6.3.5. Comments:

- No inline comments, comments should be above the line they comment.
- It can be useful to precede comments with FIXME: or NOTE: if it clarifies the intent of the comment.

6.3.6. Programming Rules:

- When testing for None, use is. This is because None is a singleton and the identity test is more efficient than testing for equality. Where what you really mean is an identity rather than equality test then is can also make your code more clearly convey its intent.

- Classes should inherit from object.

  If a class has no base classes, then it is better to make it a new style class and inherit from object.

  ```
  class Something(object):
  ```

- No mutable objects as default arguments.

  Default arguments are created on parsing, not when a function/method is called. This means you *mustn't* use mutable objects (like dictionaries or lists) as default arguments.

- No bare excepts. If you really want to catch *all* exceptions, then use except Exception: but it is *many* times better to only trap exceptions that you expect. Doing otherwise will suppress a multitude of bugs.
- If you are writing a module, always define __all__.

6.3.7. Naming conventions:

Consistency here is certainly good as it helps to identify what sort of object names point to. I think the conventions I use basically follow PEP8.

- Module names should be lowercase with underscores instead of spaces. (And should be valid module names for importing.)
- Variable names and function/method names should also be lowercase with underscores to separate words.
- Class names should be CamelCase (uppercase letter to start with, words run together, each starting with an uppercase letter).
- Module constants should be all uppercase.
- E.g. You would typically have module.ClassName.method_name.

- Module names in CamelCase with a main class name identical to the module name are annoying. (e.g. ConfigParser.ConfigParser, which should *always* be spelt configobj.ConfigObj.)
- *Also*, variables, functions, methods and classes which *aren't* part of your public API, should begin with a single underscore. (using double underscores to make attributes private almost always turns out to be a mistake - especially for testability.)

Whitespace:

- And finally, you should always have whitespace around operators and after punctuation. The exception is default arguments to methods and functions.
- E.g. def function(default=argument): and x = a * b + c

CHAPTER 7. TESTING

7.1. TESTING PLAN

- This test plan describes the test environment and provides the platform for testing the project “Biometric ear recognition”.
- The objective of testing is to see if the project meets the “Software and Hardware Requirements”.
- This plan will provide the guidelines for the testing team to test the functionalities embedded in the project.

7.2. TESTING STRATEGY

- Testing begins "in the small" and progresses "to the large".
- Initially individual components are tested using white box and black box techniques.
- After the individual components have been tested and added to the system, integration testing takes place. Once the full embedded product is completed, system testing is performed.

7.3. TEST SUITS DESIGN

7.3.1. Test Cases

Test suite no: 1
Test suite detail: Perform data augmentation

Table 7.1. Input data to the system

Test case id	Function name	Test case (condition)	Expected results	Actual result	Pass/fail/Accuracy
1	ImageDataGenerator	Images	Should perform augmentation	Does augmentation	Pass
2	ImageDataGenerator	Batch of images	Should perform augmentation	Does augmentation	Pass

Test suites no: 2
Test suite detail: Prediction.

Table 7.2. Image Dimensions

Test case id	Function name	Test case (condition)	Expected results	Actual result	Pass/fail/accuracy
1	Hist()	Labelled augmented images are feed	Should train the images depending on epochs	Trains the images	Val_acc: 90% Train_acc: 97%
2	Hist()	Un-augmented images are feed	Should correctly classify the images	Low accuracy	Val_acc: 40% Train_acc: 90%

Test suites no: 3
Test suite detail: Testing a new image.

Table 7.2. Image Dimensions

Test case id	Function name	Test case (condition)	Expected results	Actual result	Pass/fail
1	test_img	Giving new testing image from directory	Should load the image	Loads the images	Pass

CHAPTER 8. CONCLUSION AND DISCUSSION

8.1. SELF ANALYSIS OF PROJECT VIABILITIES

Implementation of project is much necessary for knowing all the limitation of system. The main advantages of this system is, it provides the desired output with consuming lesser time with better accuracy. The System is working successfully.

8.2. PROBLEM ENCOUNTERED AND POSSIBLE

The installation of packages and environments are very fragile, so before installing, it should be known that which package works best with which version. Also the augmented images are distorted from the sides. This could not be solved right now, as Keras ImageDataGenerator() does not support trimming of images. However, while training, this change does not affect the overall accuracy.

The training time heavily relies on GPU computing capability. If the capability is 5.0 or above, then the model will run fast. Else, if the system does not detect GPU, then it will run in CPU by default. It might be possible that the SGD optimizer might not be the best optimizer in every situation, because optimizers are dependent of data. So unless the user does not try all the optimizers one by one, it is certainly hard to tell which one is the best.

8.3. SUMMARY OF PROJECT WORK

- An image from the directory or camera is first augmented using the specific parameters and then given as an input to the system. It performs pre-processing, features extraction, classification, training, and then recognizes the user. It uses convolutional neural networks to learn the unique patterns of earlobe.
- Backpropagation algorithm is used to learn the network by updating the weights of the nodes and calculating the gradient descent to determine the change required to learn.
- At the end, after passing through several convolutional and pooling layers, it will detect and recognize the correct person no matter how irrelevant the position of an ear is.

CHAPTER 9. LIMITATION AND FUTURE ENHANCEMENT

9.1. LIMITATIONS

- Image dimension ordering can be a challenge because Theano and Tensorflow have different dimension orderings.
- The training time heavily relies on GPU computing capability. If the capability is 5.0 or above, then the model will run fluid. Else, if the system does not detect GPU, then it will run in CPU by default.

9.2. FUTURE ENHANCEMENTS

There are a few issues left to be further studied and implemented in order to make the ear recognition modeling system more robust and stable for various applications.

- The system will automatically augment the images instead of manual augmentation.
- The parameters for ImageDataGeneraor will be automatically selected based on the best fitting parameters.
- We are trying to train the images on multiple GPU's to make the training process faster and adding even more images per subject to increase accuracy.
- Currently, the model has to be trained first before predicting the testing image. To save the huge training time, it is planned to somehow save the weights so that they can be loaded at any time we need.
- Although the validation accuracy reaches over 90%, still there are some subjects whose individual accuracy is comparatively less. Efforts will be done to get the same accuracy of all the classes after training.
- It might be possible that the SGD optimizer might not be the best optimizer in every situation, because optimizers are dependent of data. So unless the user does not try all the optimizers one by one, it is certainly hard to tell which is the best.