BONAFIDE CERTIFICATE

Certified that this project report title **"A METHOD BASED ON DEEP LEARNING FOR IDENTIFYING INDIVIDUALS THROUGH THE USE OF EAR BIOMETRICS"** is the bonafide work of **HARI PRASATH M (Registration Number: 311421622016)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Supervisor	**Head of the Department**
(Ms. J. Irine Priya)	**(Dr. N. Naga Saranya)**

Submitted to Project and Viva Examination held on _______________

Internal Examiner	**External Examiner**

ABSTRACT

In order to determine the elements of the human ear recognition is based on gender categorization using ear image analysis and ear symmetry analysis. We use model-based analysis and deep learning methods to improve both framework and performance. The proposal made by this system makes use of transfer learning and ResNet50 to categorize the gender of images of ears. According to past studies, human ears can be used for identification, gender classification, and verification of family. Despite this, the symmetry of human ears has been discovered to be a poor predictor of identity. Hence, we focus on other qualities, such as ridge patterns, the contour of the earlobe, the size of the ear, and the shape of the ear itself. We use a dataset of ear images that is diverse and representative of the population in order to test the generalizability of the model by evaluating how well it performs on different subsets of the total data. Our research has shown that transfer learning using ResNet50 may be used to accurately classify a subject's gender based on a photograph of their ear. This approach could result in the generation of novel concepts in the field and have repercussions for biometric identification systems in the real world.

ACKNOWLEDGEMENT

I am indebted to few people who have contributed a lot towards this project.

I am extremely thankful to the chairman **Tmt. Gomathi Radhakrishnan**, and our beloved director **Mrs. R. Jayanthi Prabhakaran** for providing the full encouragement and all the necessary facilities to carry out this project.

I would like to thank our beloved principal **Dr. R. Ganesan** for providing me withall necessary facilities to complete this project.

I am thankful to **Dr. N. Naga Saranya,** Head of the Department, who helped in choosing a good project and for her constant encouragement to complete this project successfully.

I would like to thank my internal guide **Ms. J. Irine Priya**, who gave me full support and assisted me in all critical situations.

I seem it is a great privilege to express my sincere thanks to **Majestic Technologies, Paadi,** who arranged me such a good atmospherethroughout the project.

I would like to thank my project Leader **Mr. Mohammed Abbas, Majestic Technologies, Paadi,** who made this project lively and led me in the right wayto complete the project and once again my sincere thanks to this interest, that he showedin my project and the encouragement he gave to make this project successful one.

I would like to thank many other individuals at "**Meenakshi College of Engineering**" who have contributed greatly towards the success of my project.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**iii**
	LIST OF FIGURES	**viii**
	LIST OF ABBREVIATIONS	**ix**
1	**INTRODUCTION**	**1**
	1.1 ABOUT THE COMPANY	2
	1.2 ABOUT THE PROJECT	2
	1.3 PROBLEM DEFINITION	3
2	**SYSTEM ANALYSIS**	**4**
	2.1 EXISTING SYSTEM	4
	2.2 PROPOSED SYSTEM	6
	2.3 FEASIBILITY STUDY	8
	2.4 MODULE DESCRIPTION	9
	2.4.1 DATASET COLLECTION	10
	2.4.2 PREPROCESSING	10
	2.4.3 DATASET SPLITTING	10
	2.4.4 TRANSFER LEARNING	11
	2.4.5 EVALUATION	11

CHAPTER NO.	TITLE	PAGE NO.
3	**DEVELOPMENT ENVIRONMENT**	**12**
	3.1 HARDWARE REQUIREMENT	12
	3.2 SOFTWARE REQUIREMENT	12
	3.3 LANGUAGE REVIEW	12
	3.3.1 INTRODUCTION TO MACHINE LEARNING	12
	3.3.1.1 PYTHON	14
	3.3.1.2 ANACONDA SOFTWARE	16
	3.3.1.3 JUPYTER NOTEBOOK	17
	3.3.1.4 FEATURES OF JUPYTER NOTEBOOK	18
	3.3.1.5 TENSORFLOW	18
	3.3.2 FEATURES OF PYTHON	19
4	**SYSTEM DESIGN**	**20**
	4.1 SYSTEM ARCHITECTURE	21
	4.2 E-R DIAGRAM	21
	4.3 USE CASE DIAGRAM	23
	4.4 SEQUENCE DIAGRAM	25
	4.5 ACTIVITY DIAGRAM	27
	4.6 DATA FLOW DIAGRAM	29

CHAPTER NO.	TITLE	PAGE NO.
5	**SYSTEM IMPLEMENTATION**	**32**
	5.1 CODING	32
6	**TESTING**	**39**
	6.1 TESTING METHODOLOGY	39
	6.1.1 UNIT TESTING	42
	6.1.2 BLACK BOX TESTING	42
	6.1.3 INTEGRATION TESTING	43
	6.1.4 SYSTEM TESTING	43
	6.1.5 REGRESSION TESTING	44
7	**PERFORMANCE AND LIMITATION**	**45**
	7.1 PERFORMANCE	45
	7.2 LIMITATION	45
	7.3 CONCLUSION AND FUTURE ENHANCEMENT	45
8	**APPENDICES**	**47**
	8.1 SAMPLE OUTPUTS	47
	REFERENCES	**52**

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
4.1	System Architecture	21
4.2	E-R Diagram	22
4.3	Use Case Diagram	24
4.4	Sequence Diagram	26
4.5	Activity Diagram	28
4.6	Data Flow Diagram	31
8.1	Data Collection	47
8.2	Training and Validation	48
8.3	Confusion Matrix	48
8.4	Accuracy Prediction	49
8.5	Loss Prediction	49
8.6	Model Prediction	50
8.7	Evaluation	50
8.8	Result Analysis	51

LIST OF ABBREVIATIONS

APGWO	Adaptive Particle Grey Wolf Optimization
CLI	Command Line Interface
CNN	Convolutional Neural Network
CNRI	Corporation for National Research Initiatives
CWI	Centrum Wiskunde & Informatica
DFD	Data Flow Diagram
DNN	Deep Neural Network
GUI	Graphical User Interface
IDE	Integrated Development Environments
KNN	K-Nearest Neighbors
PIL	Python Imaging Library

CHAPTER 1

INTRODUCTION

Ears have been recognized for a long time as being singular to their owners and, as a result, comprise a trustworthy biometric. A holistic and model-based approach have traditionally been used in ear recognition research, with more recent research focusing on deep learning. Although though hair can hide the ear, employing the ear as a biometric system has a number of distinct benefits due to the fact that it is not affected by facial expressions. Numerous different approaches have been aimed at, and tested on, standardized datasets in which the ear position has been controlled. These datasets include early datasets such as XM2VTS as well as later and larger ones such as SC face. Because a subject's individuality is the most important factor in biometrics, this is a phase that must always be included in any biometric system. We need to demonstrate and comprehend the capacity on unconstrained ear images in order to bring ear biometrics into real-world applications. This means that the ear does not necessarily need to be in a plane that is orthogonal to the camera's view for us to be able to do this. The answer to the question of whether or not there is bilateral symmetry between the ears and which sections, if any, should be emphasized for recognition is not yet known based on the research that has been conducted thus far.

This article discusses the research that has been done in an effort to study these concerns as part of the process of developing ear biometrics for use in settings other than the laboratory. Even though there is a heightened concern for privacy, there is frequently very little attention given to concealing a subject's ears. Ears are rarely recorded in identifying documents, and even when they are, there is rarely any consideration made to concealing them.When compared to fingerprints, the age of ears (8–70 years old) appears to vary less slowly than that of faces. The ability to acquire ear pictures from a larger distance and without the subject's permission is useful. In 1890, French criminologist Alphonse Bertillon became the first person to identify human ears as a biometric.Later, Iannarelli studied 10,000 ear samples to demonstrate that each person's ears are unique and created a manual technique for ear identification.

As the first to demonstrate recognition, Burge et al. introduced an automated ear biometric technique. With soft biometric characteristics like gender as a supplement, the whole identity information from ear pictures is possible.

1.1 ABOUT THE COMPANY

MAJESTIC TECHNOLOGIES providing Industrial, Academic projects, Research solutions and Internship Training, Since a two decade in CHENNAI-PADI NAGAR, TAMIL NADU, INDIA established in the year of 2000.Twenty years of experience with perfect infrastructure, lab set up, work shop, Expertise faculties make us competitive service providers in the field Industrial, Academic projects and Internships in various locations of India. We offer final year academic projects for all the branches of Engineering, Diploma and Research Scholars. All the projects are developed according to students' requirements and university standards. To provide research-based policy advice to help governments tackle major global challenges of the G20 and beyond. To start a world-wide movement that aims to recouple economic, political and environmental prosperity with social prosperity**.** Our Internship program is a new concept in developing students into industrial ready professionals.

1.2 ABOUT OF THE PROJECT

Ears have been recognized as a trustworthy biometric due to their uniqueness and are not affected by facial expressions.Various research approaches, including deep learning, have been used to study ear recognition on standardized datasets like XM2VTS and SC face.The individuality of a person's ear is a crucial factor in biometrics.The capacity of ear biometrics on unconstrained ear images needs to be understood for real-world applications, regardless of the ear's position relative to the camera.The bilateral symmetry of ears and the specific sections that should be emphasized for recognition are still unknown based on current research.Ears are often overlooked when it comes to privacy concerns and they are rarely recorded in identifying documents.Compared to fingerprints, the age of ears varies less slowly than that of faces and ear pictures can be acquired from a distance without the subject's permission.

1.3 PROBLEM DEFINITION

The problem statement for this research is to improve the accuracy and effectiveness of biometric identification systems by exploring the potential of using human ear characteristics as a reliable method for identification, particularly for gender categorization. The research aims to address the limitations of previous studies, particularly the poor predictive value of ear symmetry for identification purposes. The study proposes the use of model-based analysis and deep learning methods, such as transfer learning and ResNet50, to enhance the performance of ear recognition systems. The research also aims to develop a diverse dataset of ear images to evaluate the generalizability of the model and improve its applicability to real-world biometric identification systems. Ultimately, the research seeks to generate novel concepts in the field and contribute to the development of more effective and reliable biometric identification systems.

CHAPTER 2

SYSTEM ANALYSIS

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and another is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then the successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of the existing running system is also difficult, improper understanding of the present system can lead to diversion from solution.

2.1 EXISTING SYSTEM

In the field of ear biometrics, several existing systems have been developed and studied for various applications such as identification, gender classification, and kinship verification. These systems utilize different techniques and algorithms to extract features from ear images and make predictions or classifications based on those features. One common approach in existing systems is to focus on the symmetry analysis of the human ear. These systems analyze the structural symmetry of the ear by comparing the left and right ear images. They extract symmetrical features such as the position of key landmarks, shape contours, or ridge patterns and use them for identification or classification purposes. However, it has been observed that the performance of existing systems based solely on ear symmetry is not highly accurate or reliable.

Existing systems also face certain drawbacks, including false positives and privacy concerns. False positives occur when the system incorrectly identifies or classifies an individual's gender or identity. This can be problematic, especially in applications where accuracy is crucial. Privacy concerns arise when biometric systems require the collection of sensitive personal information or when the images used for analysis are not adequately protected.To overcome these limitations, the proposed system aims to improve upon the existing system by expanding the analysis beyond ear symmetry. It considers additional characteristics such as ridge patterns, earlobe contour, ear size, and shape. By incorporating these additional features, the proposed system seeks to enhance the accuracy of ear recognition and reduce false positives. It also addresses privacy concerns by utilizing biometric identification systems that do not require the collection of sensitive personal information.

In the domain of ear biometrics, numerous systems have been conceptualized, developed, and investigated to cater to a diverse range of applications, including identification, gender classification, and kinship verification. These systems leverage a wide array of techniques and algorithms, all aimed at distilling pertinent information from ear images, subsequently facilitating predictions or classifications based on the extracted features.

A prevailing approach in the realm of existing ear biometric systems revolves around the meticulous analysis of ear symmetry. These systems meticulously scrutinize the structural symmetry intrinsic to the human ear by meticulously comparing images of the left and right ears. Through this comparative analysis, they successfully distill symmetrical features of paramount importance. These features often encompass the precise positioning of key landmarks, intricate shape contours, or the complex patterns of ridges. These symmetrical attributes are then harnessed to fulfill various goals, be it the accurate identification of individuals or the discernment of particular gender classifications.

However, a discerning observation emerges upon evaluating the performance of these existing systems that heavily lean on the concept of ear symmetry. While the underlying concept is theoretically sound and intuitively aligned with the inherent bilateral symmetry of the human body, the actual performance outcomes present a more nuanced reality. The accuracy and reliability demonstrated by these systems, while notable, do not consistently meet the stringent benchmarks demanded by modern biometric applications.

In light of the observed limitations in relying solely on ear symmetry as the cornerstone of biometric systems, the field is ripe for innovative approaches that not only respect the fundamental tenets of symmetry but also incorporate supplementary dimensions that could potentially enhance accuracy and reliability. These dimensions could span additional anatomical features, textural attributes, or even advanced machine learning paradigms. By embracing a more comprehensive and multi-faceted perspective, researchers and developers stand to unlock the latent potential of ear-based biometric systems and elevate their practical utility to unprecedented heights.

In summation, while the existing systems grounded in the study of ear symmetry represent a commendable stride in the realm of biometrics, the performance constraints they exhibit highlight the need for a paradigm shift. This shift entails an evolution towards holistic, multi-pronged approaches that harness symmetry while embracing a broader array of features and techniques. Through such an evolution, the field can aspire to surmount current limitations and chart a course toward biometric systems that seamlessly integrate into a myriad of real-world applications with unmatched accuracy and dependability.

Disadvantages

- Mistakenly identify someone’s gender or identity.
- Posing risks of unauthorized access or misuse of biometric data.
- They focus on symmetry and overlook other important ear features.
- It may struggle to accurately identify individuals to classify their gender.

2.2 PROPOSED SYSTEM

Ear biometrics stands to benefit from the suggested method, which zeroes in on gender classification via image analysis and symmetry analysis of the external auditory canal. To enhance its structure and functionality, the system makes use of model-based analysis and deep learning techniques. The suggested system is described in great depth below.The suggested system's main goal is to correctly identify a person's gender from an image of their ears. The system's goal is to aid in identification, gender classification, and family relationship verification through the use of acoustic fingerprinting of the ear.

The suggested approach gathers a large sample of ear pictures from which to draw. The gender categorization model relies heavily on this dataset for both training and testing purposes. It guarantees the system's adaptability to specific populations.To guarantee uniformity and to further assist analysis, the ear pictures in the dataset are preprocessed. Common image preprocessing methods like scaling, cropping, and normalization are used.

The suggested system makes use of the ResNet50 deep learning model to perform transfer learning. The system may utilize pre-trained models on big datasets through transfer learning to automatically extract useful information from ear pictures. The gender classification model is based on ResNet50, a widely used deep neural network architecture.The system's main focus is on extracting information from ear pictures that can be used for gender classification. The suggested system takes into account characteristics other than symmetry analysis, including ridge patterns, earlobe contour, ear size, and ear shape. Including these details in the biometric representation of the ear makes it more accurate.

The suggested approach assesses the efficacy of the gender categorization model on various subsets of the information to determine its generalizability. This test verifies that the model can correctly identify people's sex outside of the training data. Overfitting is avoided and model performance is maximised through hyperparameter fine-tuning.Accuracy and Validation The proposed method evaluates the effectiveness of the gender classification model using a dedicated validation set. In this stage, we test how well the model can identify a person's gender from an image of their ears. The system's goal is to improve the precision with which gender is identified in practical settings.

Research and discoveries from the suggested system may have applications in other biometric identification systems. It has the potential to advance research and new ideas in the study of the ear as a biometric. Due to the system's precision and scalability, it may find use in sectors including law enforcement, healthcare, and the financial sector, among others.

Advantages:

- It speed up the authentication process by analysing ear pictures.
- More accurate gender classification.
- It can handle large volumes of information and complex patterns.
- It exploring the relationship between ear features and gender classification.
- It aims to minimize false positives and negatives by combining different attributes.

2.3 FEASIBILITY STUDY

The preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study isto test the Technical, Operational and Economical feasibility for adding new modules and debugging the oldest running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- ➢ Technical Feasibility
- ➢ Operational Feasibility
- ➢ Economic Feasibility

2.3.1Technical Feasibility

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- ➢ Does the necessary technology exist to do what is suggested?
- ➢ Does the proposed equipment have the technical capacity to hold the datarequired to use the new system?
- ➢ Will the proposed system provide an adequate response to inquiries, regardlessof the number or location of users?
- ➢ Can the system be upgraded if developed?
- ➢ Are there technical guarantees of accuracy, reliability, ease of access and datasecurity?

Earlier no system existed to cater to the needs of the ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is abrowser-based user interface for audit workflow. Thus, it provides easy access to users.The purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles.Therefore, it provides a technical guarantee of accuracy, reliability, and security. The software andhard requirements for the development of this project are not many and are already available or are available as free as open source.The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing fast feedback to the users irrespective of the number of users using the system.

2.3.2 Operational Feasibility

The analyst considers the extent the proposed system will fulfill his departments. That is, whether the proposed system covers all aspects of the working system and whether it has considerable improvements. We have found that the proposed“Secure transaction” will certainly have considerable improvements over the existing system.

2.3.3Economic Feasibility

The proposed system is economically feasible because the cost involved in purchasing the hardware and the software is approachable. Working in this system need not require a highly qualified professional. The operating-environment costs are marginal. The less time involved also helped in its economic feasibility.

2.4 MODULE DESCRIPTION

MODULES:

1. Dataset Collection
2. Preprocessing
3. Dataset Splitting
4. Transfer Learning
5. Evaluation

2.4.1 DATASET COLLECTION

This module involves collecting a diverse set of ear images that are representative of the population. The images should cover different ethnicities, ages, and genders, with varying lighting conditions and angles. The dataset be obtained from public sources. Within this module, the primary objective revolves around the meticulous assembly of a diverse array of ear images that faithfully mirror the intricacies of our global populace. To achieve this, a comprehensive selection of images is required, encompassing a spectrum of ethnic backgrounds, age groups, and genders. This amalgamation ensures that the resulting dataset transcends homogeneity and accurately portrays the rich tapestry of human variation. Moreover, the dataset's robustness is fortified by the inclusion of images captured under an assortment of lighting conditions and from various angles.

2.4.2 PREPROCESSING

This module involves preprocessing the ear images to ensure consistency and reduce noise. Techniques such as resizing, cropping, and normalization may be applied to ensure that the images have a consistent size and brightness level. Additionally, the module involves feature extraction to extract meaningful features such as earlobe shape, ridge patterns, and ear shape. In the realm of image analysis, the preprocessing phase within this module emerges as a pivotal step, wielding the power to transform raw ear images into a harmonious and coherent dataset. With a focus on standardization and noise reduction, a suite of techniques comes into play. The application of resizing, cropping, and normalization stands as a trio of measures aimed at achieving uniformity in image dimensions and brightness levels. By resizing images to a common scale, cropping excess information, and normalizing pixel values, a level playing field is established, facilitating subsequent analysis unencumbered by inconsistencies.

2.4.3 DATASET SPLITTING

This module involves preprocessing the ear images to ensure consistency and reduce noise. Techniques such as resizing, cropping, and normalization may be applied to ensure that the images have a consistent size and brightness level. Additionally, the module involves feature extraction to extract meaningful features such as earlobe shape, ridge patterns, and ear shape. In this module, the art of dataset splitting takes center stage as a strategic maneuver to wield the potential of the curated ear image dataset effectively. By carving the dataset into distinct segments, typically training, validation, and testing subsets, this process establishes the scaffolding upon which robust machine learning models can be erected. This partitioning is used to prevent overfitting, gauge model performance, and ultimately enable the development of algorithms.

2.4.4 TRANSFER LEARNING

This module involves using transfer learning with the ResNet50 model to train a gender classification model on the ear images. Transfer learning allows the model to leverage pre-trained weights to improve its performance on a new task. Within the framework of this module, the strategic concept of transfer learning assumes a prominent role in the quest to fashion a proficient gender classification model from the ear image dataset. This approach capitalizes on the capabilities of the ResNet50 model, an established deep learning architecture. Leveraging the prowess of transfer learning, the model embarks on a journey to transcend its original purpose and excel in the novel task of gender classification. By initially training the ResNet50 model on a large dataset unrelated to the current objective, it acquires a foundational understanding of features common to various images. Subsequently, these pre-learned weights serve as a springboard, enabling the model to swiftly adapt its knowledge to the nuanced challenge of categorizing genders based on ear images.

2.4.5 EVALUATION

This module involves testing the performance of the model on the testing set to evaluate its generalizability. Additionally, the module involve the use of visualization techniques to analyze the learned features and gain insights into the model's decision-making process.This evaluation takes place by rigorously subjecting the model to the testing subset of the ear image dataset. The objective is to gauge its generalizability – its capacity to accurately classify genders beyond the confines of its training environment. By meticulously analyzing the model's performance metrics, such as accuracy, precision, recall, and F1 score, a comprehensive understanding of its strengths and limitations is garnered. This process forms a critical checkpoint, ensuring that the model's proficiency translates into real-world applicability and robustness.

CHAPTER 3

DEVELOPMENT ENVIRONMENT

3.1 HARDWARE REQUIREMENT

RAM	: 8GB RAM
PROCESSOR	: Intel CORE i5
HARD DISK	: 100 GB

3.2 SOFTWARE REQUIREMENT

LANGUAGE	: Python
PLATFORM	: Anaconda
PYTHON LIBRARIES	: Keras, NumPy, Seaborn
OPERATING SYSTEM	: WINDOWS 10
IDE	: Jupyter Notebook

3.3 LANGUAGE REVIEW

3.3.1 Introduction To Machine Learning

Machine Learning is the most popular technique of predicting the future or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and also analyze the historical data. Therefore, as it trains over the examples, again and again, it is able to identify patterns in order to make predictions about the future. Data is the core backbone of machine learning algorithms. With the help of the historical data, we are able to create more data by training these machine learning algorithms.

For example, Generative Adversarial Networks are an advanced concept of Machine Learning that learns from the historical images through which they are capable of generating more images. This is also applied towards speech and text synthesis. Therefore, Machine Learning has opened up a vast potential for data science applications.

Machine Learning combines computer science, mathematics, and statistics. Statistics is essential for drawing inferences from the data. Mathematics is useful for developing machine learning models and finally, computer science is used for implementing algorithms. However, simply building models is not enough. You must also optimize and tune the model appropriately so that it provides you with accurate results. Optimization techniques involve tuning the hyperparameters to reach an optimum result.

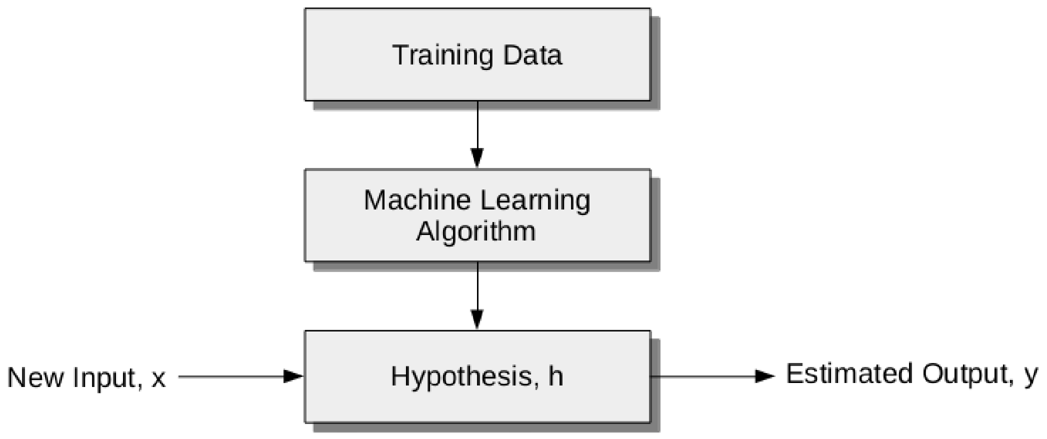
The world today is evolving and so are the needs and requirements of people. Furthermore, we are witnessing a fourth industrial revolution of data. In order to derive meaningful insights from this data and learn from the way in which people and the system interface with the data, we need computational algorithms that can churn the data and provide us with results that would benefit us in various ways. Machine Learning has revolutionized industries like medicine, healthcare, manufacturing, banking, and several other industries.

Data is expanding exponentially and in order to harness the power of this data, added by the massive increase in computation power, Machine Learning has added another dimension to the way we perceive information. Machine Learning is being utilized everywhere. The electronic devices you use, the applications that are part of your everyday life are powered by powerful machine learning algorithms.

Types of Machine Learning

Machine Learning Algorithms can be classified into 3 types as follows:

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning



3.3.1.1Python

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems.Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.Python coding style comprises physical lines as well as logical lines or statements. A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as C and C++ where a semicolon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines.The use of a semi-colon is not prohibited in Python, although it is not mandatory. The NEWLINE token denotes the end of the logical line. A logical line that only contains spaces, comments, or tabs are called blank lines and they are ignored by the interpreter.

Multiple Statements on a Single Line:

In Python, it is possible to club multiple statements in the same line using a semi-colon; however, most programmers do not consider this to be a good practice as it reduces the readability of the code.

Whitespaces and Indentation:

Unlike most of the programming languages, Python uses indentation to mark a block of code. According to Python coding style guideline or PEP8, we should keep an indent size of four. Most of the programming languages provide indentation for better code formatting and do not enforce to have it. But in Python it is mandatory. This is why indentation is so crucial in Python. Comments in any programming language are used to increase the readability of the code. Similarly, in Python, when the program starts getting complicated, one of the best ways to maintain the readability of the code is to use Python comments. It is considered a good practice to include documentations and notes in the python syntax since it makes the code way more readable and understandable to other programmers as well, which comes in handy when multiple programmers are simultaneously working on the same project.

Following are different kinds of comments that can be included in our Python program:

Single Line Comments:

Single line Python comments are marked with # character. These comments end at the end of the physical line, which means that all characters starting after the # character (and lasts till the end of the line) are part of the comment.

Docstring Comments:

Python has the documentation strings (or docstrings) feature which is usually the first statement included in functions and modules. Rather than being ignored by the Python Interpreter like regular comments, docstrings can actually be accessed at the run time using the dot operator. It gives programmers an easy way of adding quick notes with every Python module, function, class, and method. To use this feature, we use triple quotes in the beginning of the documentation string or comment and the closing triple quotes at the end of the documentation comment. Docstrings can be one-liners as well as multi-liners. docstrings can actually be accessed at the run time using the dot operator.

Python Block Comments:

We can use several single line comments for a whole block. This type of comment is usually created to explain the block of code that follows the Block comment. Python Block comment is the only way of writing a real comment that can span across multiple lines. It is supported and preferred by Pythons PEP8 style guide since Block comments are ignored by Python interpreter or parser.

Data Types:

One of the most crucial part of learning any programming language is to understand how data is stored and manipulated in that language. Users are often inclined toward Python because of its ease of use and the number of versatile features it provides. One of those features is dynamic typing. In Python, unlike statically typed languages like C or java, there is no need to specifically declare the data type of the variable. In dynamically typed languages such as Python, the interpreter itself predicts the data type of the Python Variable based on the type of value assigned to that variable.

3.3.1.2Anaconda Software

Anaconda is the data science platform for data scientists, IT professionals and business leaders of tomorrow. It is a distribution of Python, R, etc. With more than 300 packages for data science, it becomes one of the best platforms for any project. Anaconda helps in simplified package management and deployment. Anaconda comes with a wide variety of tools to easily collect data from various sources using various machine learning and AI algorithms. It helps in getting an easily manageable environment setup which can deploy any project with the click of a single button. Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help you with data collection through artificial intelligence and machine learning algorithms.

Anaconda Navigator is a desktop GUI that ships with Anaconda and lets you launch applications and manage conda packages, environments, and channels without having to use a command-line interface. It can search for packages in a local Anaconda repository or on Anaconda Cloud. With Navigator, you don’t need to type commands in a terminal, it lets you work with packages and environments with just a click. The individual edition includes popular package names like numpy, pandas, scipy, sklearn, tensorflow, pytorch, matplotlib and more. The Anaconda Prompt and Power shell make working within the filesystem easy and manageable. Anaconda helps in simplified package management and deployment. Anaconda comes with a wide variety of tools to easily collect data from various sources using various machine learning and AI algorithms.

3.3.1.3 Jupyter Notebook

Jupyter Lab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.The jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.Jupyter Lab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality. The jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience. As a server-client application, the jupyter Notebook App allows you to edit and run your notebooks via a web browser. The application can be executed on a PC without Internet access, or it can be installed on a remote server, where you can access it through the Internet.

Its two main components are the kernels and a dashboard.

- A kernel is a program that runs and introspects the users code. The jupyter Notebook App has a kernel for Python code, but there are also kernels available for other programming languages.
- The dashboard of the application not only shows you the notebook documents that you have made and can reopen but can also be used to manage the kernels: you can which ones are running and shut them down if necessary.

3.3.1.4 Features Of Jupyter Notebook

- **Pluggable authentication**

  Manage users and authentication with PAM, OAuth or integrate with your own directory service system.
- **Centralized deployment**

  Deploy the jupyter Notebook to thousands of users in your organization on centralized infrastructure on- or off-site.
- **Container friendly**

  Use Docker and kubernetes to scale your deployment, isolate user processes, and simplify software installation.
- **Live coding environments**

  Code can be changed and run in real-time with feedback provided directly in the browser
- **Code meets data**

  Deploy the Notebook next to your data to provide unified software management and data access within your organization.

3.3.1.5 Tensorflow

Deep learning is a subfield of machine learning that is a set of algorithms that is inspired by the structure and function of the brain. Deep learning is a subset of machine learning. There are certain specialties in which we perform machine learning, and that's why it is called deep learning. For example, deep learning uses neural networks, which are like a simulation of the human brain. Deep learning also involves analyzing large amounts of unstructured data, unlike traditional machine learning, which typically uses structured data. This unstructured data could be fed in the form of images, video, audio, text, etc.TensorFlow is the second machine learning framework that Google created and used to design, build, and train deep learning models. You can use the TensorFlow library do to numerical computations, which in itself does not seem all too special, but these computations are done with data flow graphs. In these graphs, nodes represent mathematical operations, while the edges represent the data, which usually are multidimensional data arrays or tensors, that are communicated between these edges.

The name TensorFlow is derived from the operations which neural networks perform on multidimensional data arrays or tensors! Its literally a flow of tensors.Using tf.keras allows you to design, fit, evaluate, and use deep learning models to make predictions in just a few lines of code. It makes common deep learning tasks, such as classification and regression predictive modeling. The other important aspect is TensorFlow is highly scalable. You can write your code and then make it run either on CPU, GPU, or across a cluster of these systems for the training purpose. Generally, training the model is where a large part of the computation goes. Also, the process of training is repeated multiple times to solve any issues that may arise.

This process leads to the consumption of more power, and therefore, you need a distributed computing. If you need to process large amounts of data, TensorFlow makes it easy by running the code in a distributed manner.GPUs, or graphical processing units, have become very popular. Nvidia is one of the leaders in this space. It is good at performing mathematical computations, such as matrix multiplication, and plays a significant role in deep learning. TensorFlow also has integration with C++ and Python API, making development much faster.A tensor is a mathematical object represented as arrays of higher dimensions. These arrays of data with different sizes and ranks get fed as input to the neural network. These are the tensors.

3.3.2 Features Of Python

- Universal Language Construct
- Support both High Level and Low Level Programming Language Interoperability
- Fastest Development life cycle therefore more productive coding environment
- Less memory used because a single container hold
- Multiple data types and each type doesn’t require its own function Learning
- Ease and open source development
- Speed and user-friendly data structure Extensive and extensible libraries.

CHAPTER 4

SYSTEM DESIGN

Software design sits in the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement has been specified and analyzed, system design is the first of the three technical activities -design, code and test that is requiredto build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. The design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design, we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can beviewed from either a technical or project management perspective. From the technical point of view, the design is comprised of four activities – architectural design, data structure design, interface design, and procedural design.

4.1 SYSTEM ARCHITECTURE

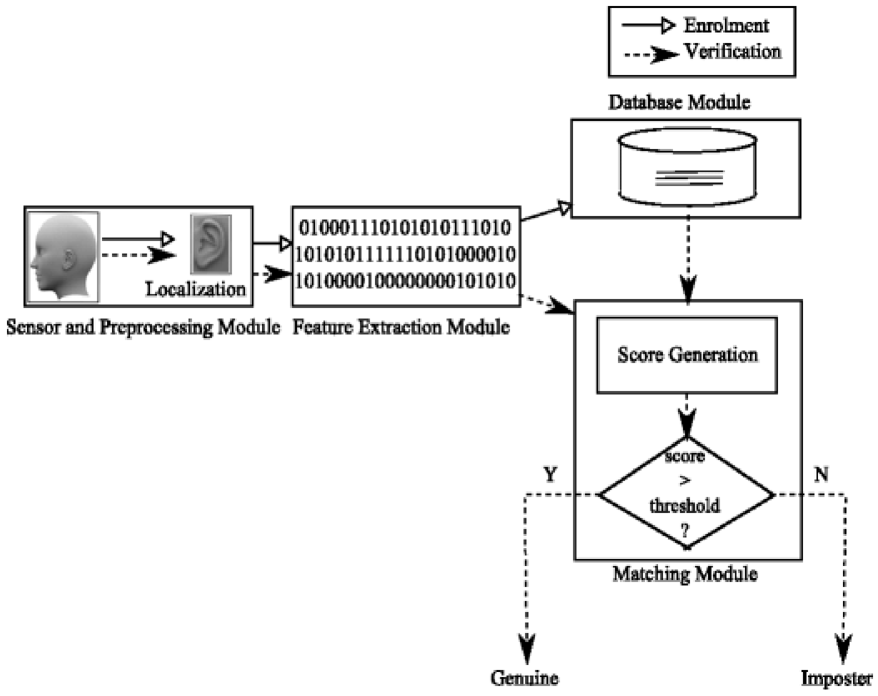


Figure 4.1 Architecture Diagram

4.2 E-R DIAGRAMS

The relation upon the system is structured through a conceptual ER-Diagram, which not only specifics the existing entities, but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.The Entity-Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct, the date modeling activity the attributes of each data object noted is the ERD can be described resign a data object description.

The set of primary components that are identified by the ERD are

- Data object
- Relationships
- Attributes
- Various types of indicators.

The primary purpose of the ERD is to represent data objects and their relationships.

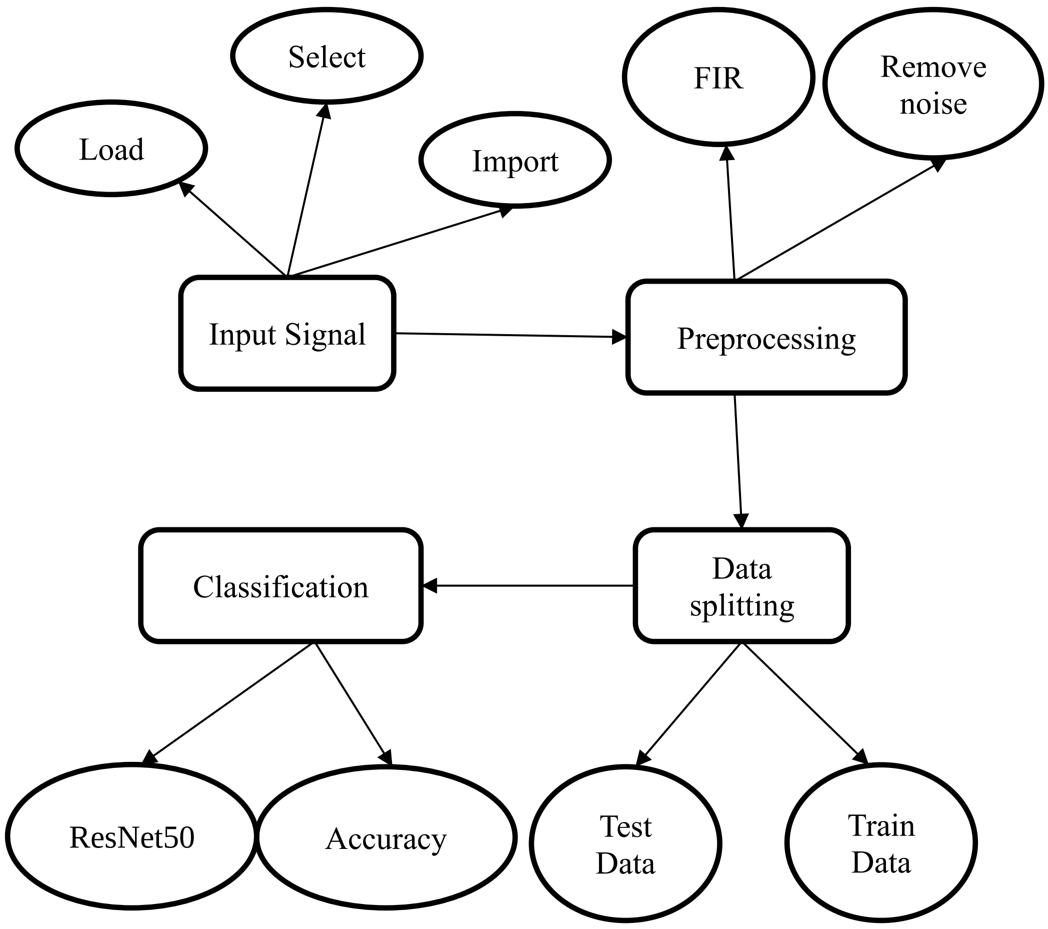


Figure 4.2 E-R Diagram

4.3 USE CASE DIAGRAM

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

A use case is a list of actions or event steps typically defining the interactions between a role (known in the Unified Modelling Language (UML) as an actor) and a system to achieve a goal. The actor can be a human or other external system.

UML use case diagrams are ideal for:

- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system
- Modelling the basic flow of events in a use case

Notations:

- **Use cases**: Horizontally shaped ovals that represent the different uses that a user might have.
- **Actors**: Stick figures that represent the people actually employing the use cases.
- **Associations**: A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
- **System boundary boxes**: A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.
- **Packages**: A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.

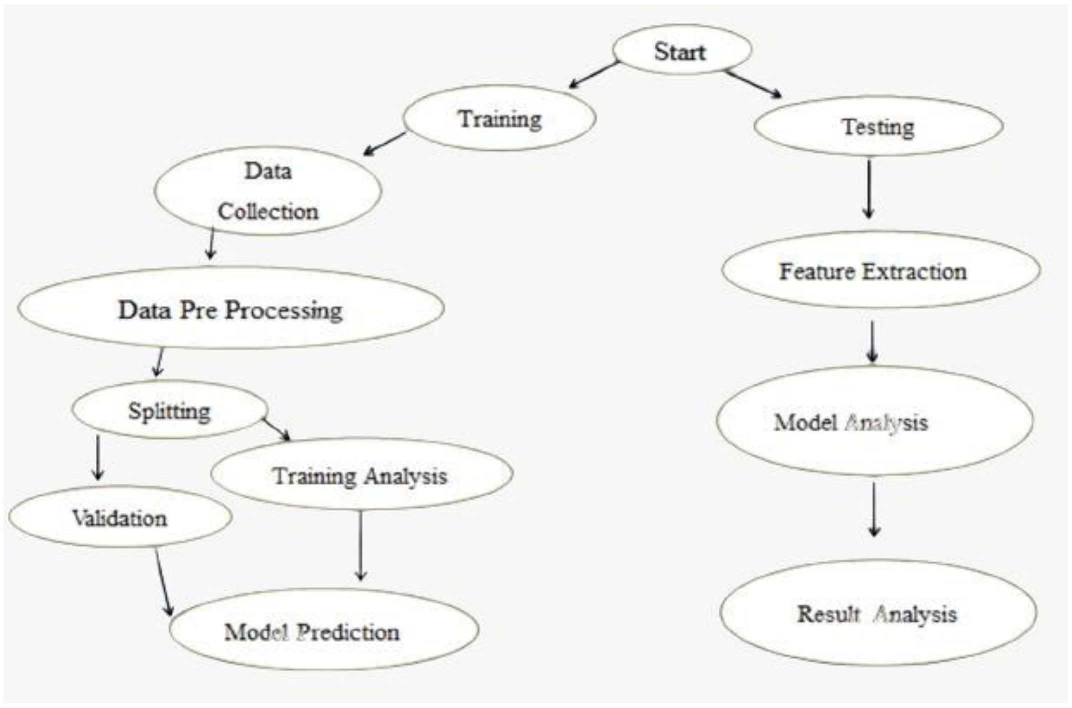


Figure 4.3 Use case Diagram

4.4 SEQUENCE DIAGRAM

Sequence diagrams document the interactions between classes to achieve a result, such as a use case. Because UML is designed for object-oriented programming, these communications between classes are known as messages. The Sequence diagram lists objects horizontally, and time vertically, and models these messages over time.

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

- **Graphical Notation**: In a Sequence diagram, classes and actors are listed as columns, with vertical lifelines indicating the lifetime of the object over time.
- **Object**: Objects are instances of classes, and are arranged horizontally. The pictorial representation for an Object is a class (a rectangle) with the name prefixed by the object.
- **Lifeline**: The Lifeline identifies the existence of the object over time. The notation 2for a Lifeline is a vertical dotted line extending from an object.
- **Activation**: Activations, modelled as rectangular boxes on the lifeline, indicate when the object is performing an action.
- **Message**: Messages, modelled as horizontal arrows between Activations.

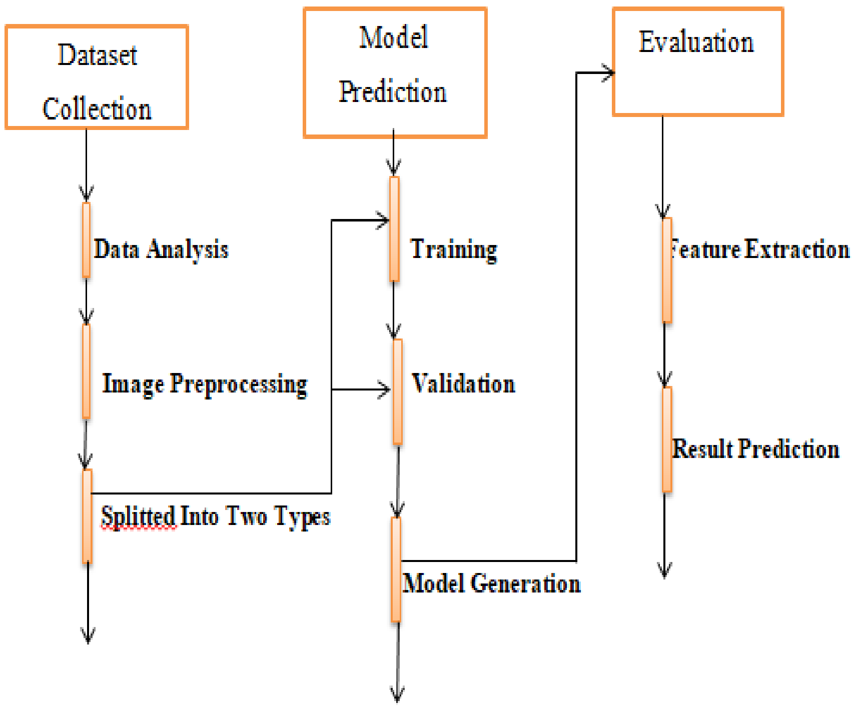


Figure 4.4 Sequence Diagram

4.5 ACTIVITY DIAGRAM

This shows the flow of events within the system. The activities that occur within a use case or within an objects behaviour typically occur in a sequence. An activity diagram is designed to be simplified look at what happens during an operations or a process. Each activity is represented by a rounded rectangle the processing within an activity goes to compilation and then an automatic transmission to the next activity occurs. An arrow represents the transition from one activity to the next. An activity diagram describes a system in terms of activities. Activities are the state that represents the execution of a set of operations.

These are similar to flow chart diagram and dataflow.

Initial state: which state is starting the process?

Action State: An action state represents the execution of an atomic action, typically the invocation of an operation. An action state is a simple state with an entry action whose only exit transition is triggered by the implicit event of completing the execution of the entry action.

Transition: A transition is a directed relationship between a source state vertex and a target state vertex. It may be part of a compound transition, which takes the static machine from one static configuration to another, representing the complete response of the static machine to a particular event instance.

Final state: A final state represents the last or "final" state of the enclosing composite state. There may be more than one final state at any level signifying that the composite state can end in different ways or conditions. When a final state is reached and there are no other enclosing states it means that the entire state machine has completed its transitions and no more transitions can occur.

Decision: A state diagram (and by derivation an activity diagram) expresses decision when guard conditions are used to indicate different possible transitions that depend on Boolean conditions of the owning object.

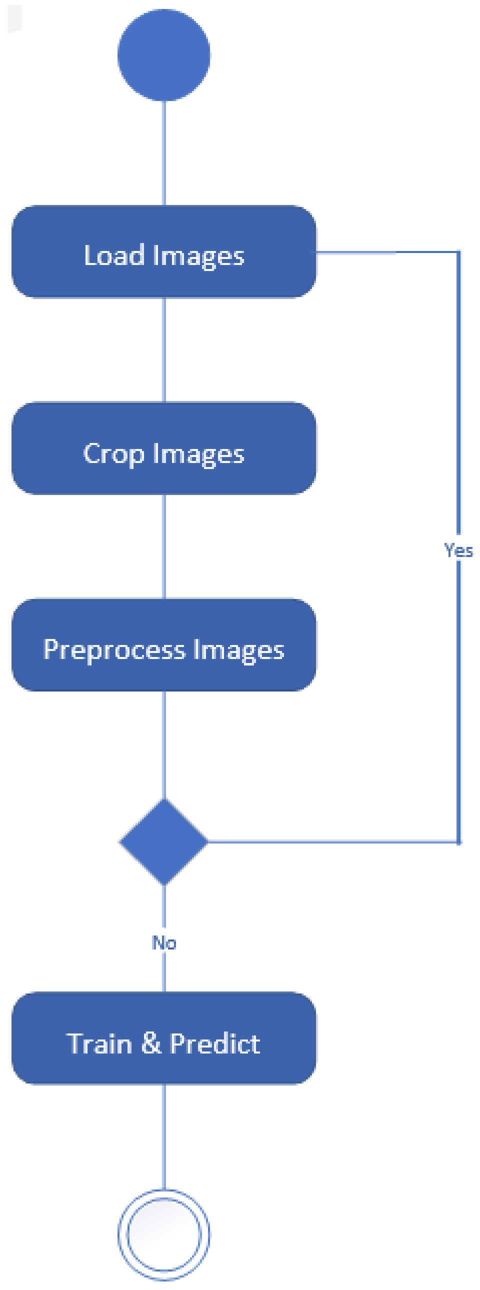


Figure 4.5 Activity Diagram

4.6 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical tool used to describe and analyze the movement of data through the system. It is a graphical representation of the “flow”of data through a computer system or a data or it looks at how data flows through a system. These are central tools and basic from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. The development of DFD is done at several levels. The flow diagram describes the boxes that describe computations, decisions, interactions & loops. It is important to keep in mind that the flow diagrams are not flowcharts and should not include control elements.

Characteristics

- Information and/or data flow is represented by a labeled arrow
- Processes (transformations) are represented by labeled circles (bubbles)
- Information sources and sinks are represented by boxes
- Files and depositories are represented by a rounded rectangle or a double line.

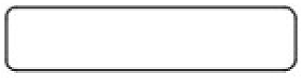
Features

- The DFD shows data, not the control loops and decisions are controlled considerations do not appear on a DFD.
- The DFD does not indicate the time factor involved in any process whether the dataflow takes place easily daily, weekly, monthly or yearly.
- The sequence of events is to bring out on DFD.

DFD Symbols

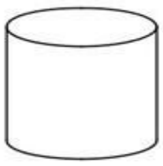
Process

A process transforms incoming data flow into outsourcing data flow.



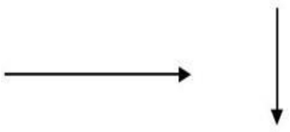
Data Store

Data sources are repositories of data in the system. They are sometimes also referred toas files.



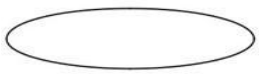
Data flow

Data flows are pipelines through which packets of information flow. Label the arrowswith the name of the data that moves through it.



External Entity

External entities are objects outside the system, with which the system communicates.



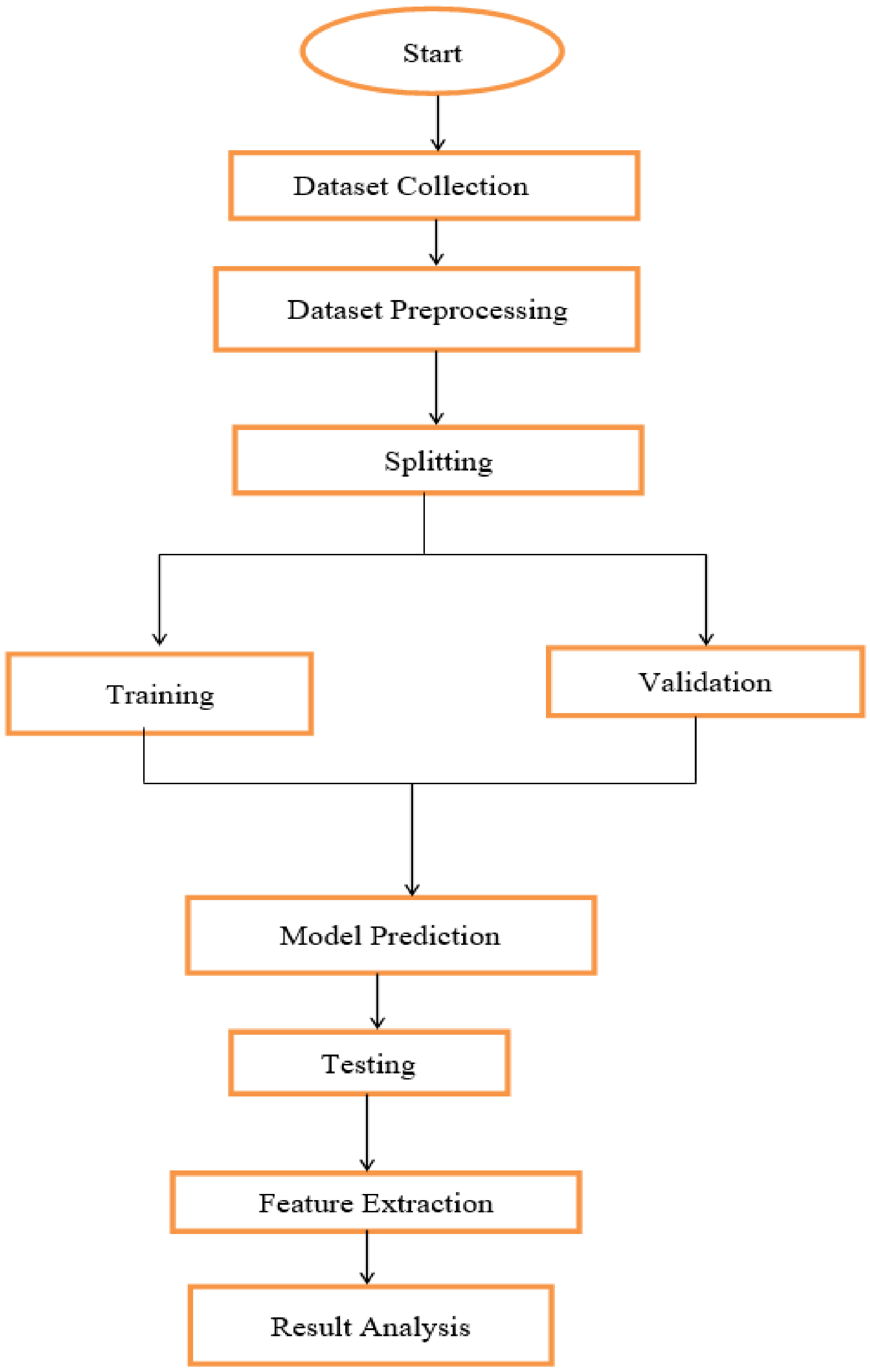


Figure 4.6 DFD Diagram

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 CODING

Ear recognition classification

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
BATCH_SIZE = 16
IMAGE_SIZE = 224
train_dataset = tf.keras.preprocessing.image_dataset_from_directory
(
    "Dataset/train",
    shuffle = True,
    image_size = (IMAGE_SIZE,IMAGE_SIZE),
    batch_size = BATCH_SIZE
)
test_dataset = tf.keras.preprocessing.image_dataset_from_directory
(
    "Dataset/test",
    shuffle = True,
    image_size = (IMAGE_SIZE,IMAGE_SIZE),
    batch_size = BATCH_SIZE
)
class_names = train_dataset.class_names
class_names
plt.figure(figsize = (15,15))
for image_batch, label_batch in train_dataset.take(1):
  for i in range(8):
```

```
ax = plt.subplot(3,4,i+1)
            plt.imshow(image_batch[i].numpy().astype("uint8"))
           plt.title(class_names[label_batch[i]],fontsize = 8)
           plt.axis("off")
start_time=time.perf_counter()
im_height = 224
im_width = 224
batch_size = 16
epochs = 5
classNum=2
saveModelName='genderclassification.h5'
image_path = "Dataset/"
train_dir = image_path + "train"
if not os.path.exists("save_weights"):
    os.makedirs("save_weights")
train_image_generator = ImageDataGenerator(rescale=1. / 255, rotation_range=40,
width_shift_range=0.2,  height_shift_range=0.2,  shear_range=0.2,  zoom_range=0.2,
horizontal_flip=True, fill_mode='nearest')
train_data_gen = train_image_generator.flow_from_directory(directory=train_dir,
batch_size=batch_size,shuffle=True,  target_size=(im_height, im_width), class_mode
= 'binary')
train_datagen =ImageDataGenerator(rescale=1./255, validation_split=0.25)
validation_generator = train_datagen.flow_from_directory
(
   'Dataset/test',
    target_size=(im_height, im_width),
    batch_size=batch_size,
    class_mode = 'binary')
# set as validation data
total_train = train_data_gen.n
covn_base = tf.keras.applications.ResNet50(weights='imagenet', include_top=False,
input_shape=(im_width, im_height, 3))
```

```
covn_base.trainable = True
model = tf.keras.Sequential()
model.add(covn_base)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
model.summary()
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),loss='bina
ry_crossentropy',metrics=["accuracy"])
history = model.fit(x=train_data_gen,steps_per_epoch=total_train //
batch_size,epochs=epochs,validation_data = validation_generator)
```

Ear recognition prediction

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
import os
im_height = 224
im_width = 224
batch_size =16
class_num=2
image_path = "Dataset/test/"
saveModelName='Dataset/genderclassification.h5'
names=os.listdir(image_path)
validation_dir = image_path
validation_image_generator = ImageDataGenerator(rescale=1. / 255)
val_data_gen =
validation_image_generator.flow_from_directory(directory=validation_dir,
batch_size=batch_size,shuffle=False,  target_size=(im_height, im_width),
class_mode='binary')
total_val = val_data_gen.n
covn_base = tf.keras.applications.ResNet50(weights='imagenet', include_top=False,
input_shape=(im_width, im_height, 3))
model = tf.keras.Sequential()
model.add(covn_base)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
model.summary()
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),loss='bina
ry_crossentropy',metrics=["accuracy"])
model.load_weights(saveModelName)
Y_pred = model.predict_generator(val_data_gen, total_val // 10+1)
Y_pred_classes=[]
for i in Y_pred:
    if(i[0]<0.5):
        Y_pred_classes.append(0)
    else:
        Y_pred_classes.append(1)
 trueLabel=val_data_gen.classes
trueLabel
 y_test=trueLabel
prediction=Y_pred_classes
from sklearn.metrics import precision_score,)
            recall_score, confusion_matrix, classification_report, \
            accuracy_score, f1_score,ConfusionMatrixDisplay
print ('Accuracy:', accuracy_score(y_test, prediction))
print ('F1 score:', f1_score(y_test, prediction,average="macro"))
print ('Recall:', recall_score(y_test, prediction,average="macro"))
print ('Precision:', precision_score(y_test, prediction,average="macro"))
class_names=names
cm = confusion_matrix(y_test, prediction)
```

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
fig, ax = plt.subplots(figsize=(15,15))
disp.plot(ax=ax,cmap=plt.cm.Blues)
plt.show()
indices = np.arange(len(val_data_gen.filenames))
np.random.shuffle(indices)
indices=list(indices)
rows = 4
from PIL import Image,ImageOps
classes=class_names
plt.figure(figsize=(15,15))
for k, n in enumerate(val_data_gen.filenames):
    i=indices[k]
    plt.subplot(rows,3,k+1)
    lk=Y_pred[i].tolist()
    if(lk[0]<0.5):
        da=0
    else:
        da=1
    plt.title("Predicted: "+classes[da] +"\n  Real : "+classes[y_test[i]])
    plt.axis('off')
    plt.imshow(Image.open("Dataset/test/"+n).convert("RGB"))
    if k==11:
        break
```

Test prediction

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
import os
im_height = 224
im_width = 224
saveModelName='genderclassification.h5'
covn_base = tf.keras.applications.ResNet50(weights='imagenet', include_top=False,
input_shape=(im_width, im_height, 3))
model = tf.keras.Sequential()
model.add(covn_base)
model.add(tf.keras.layers.GlobalAveragePooling2D())
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
model.summary()
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),loss='bina
ry_crossentropy',metrics=["accuracy"])
model.load_weights(saveModelName)
import cv2
img_path = "E:/mcaprojects/ear detection/ear/Dataset/test/female/4.jpg"
img = cv2.imread(img_path)
data_list=[]
img=cv2.resize(img, (224,224)) # resize the image
img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) # cv2 creates bgr images, convert
to rgb images
indices = np.arange(len(val_data_gen.filenames))
np.random.shuffle(indices)
indices=list(indices)
rows = 4
from PIL import Image,ImageOps
classes=class_names
plt.figure(figsize=(15,15))
for k, n in enumerate(val_data_gen.filenames):
    i=indices[k]
    plt.subplot(rows,3,k+1)
    lk=Y_pred[i].tolist()
    if(lk[0]<0.5):
        da=0
    else:
        da=1
    plt.title("Predicted: "+classes[da] +"\n  Real : "+classes[y_test[i]])
    plt.axis('off')

data_list.append(img)  # append processed image to the list
data=np.array(data_list)/255 # convert to an np array and rescale images
predictions=model.predict(data,batch_size=1, verbose=0 )
names=["Female","Male"]
result=names[1]
if(predictions[0][0]<0.5):
        result=names[0]
print("Predicted Result ", result)
plt.imshow(img)
```

CHAPTER 6

TESTING

6.1 TESTING METHODOLOGY

The software engineering process can be viewed as a spiral. Initially, system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction at each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress is done by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed.Finally, we arrive at system testing, where the software and other system elements are tested as a whole.

The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases. In the test scenario, there is a detailed testing process due to many associated test cases. Before performing the test scenario, the tester has to consider the test cases for each scenario.Testers need to put themselves in the place of the user because they test the software application under the users point of view. Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.

As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing. This is one of the most cost effective approaches to testing. If the documentation is not right: there will be major and costly problems. The documentation can be tested in a number of different ways to many different degrees of complexity. These range from running the documents through a spelling and grammar checking device, to manually reviewing the documentation to remove any ambiguity or inconsistency. Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since the earlier a defect is found the less it will cost to be fixed. The most popular testing documentation files are test reports, plans, and checklists. These documents are used to outline the teams workload and keep track of the process. Lets take a look at the key requirements for these files and see how they contribute to the process.

Test strategy:

An outline of the full approach to product testing. As the project moves along, developers, designers, product owners can come back to the document and see if the actual performance corresponds to the planned activities.

Test data:

The data that testers enter into the software to verify certain features and their outputs. Examples of such data can be fake user profiles, statistics, media content, similar to files that would be uploaded by an end- user in a ready solution.

Test plans:

A file that describes the strategy, resources, environment, limitations, and schedule of the testing process. Its the fullest testing document, essential for informed planning. Such a document is distributed between team members and shared with all stakeholders.

Test scenarios:

In scenarios, testers break down the products functionality and interface by modules and provide real-time status updates at all testing stages. A module can be described by a single statement, or require hundreds of statuses, depending on its size and scope.

Test cases:

If the test scenario describes the object of testing (what), a scenario describes a procedure (how). These files cover step-by-step guidance, detailed conditions, and current inputs of a testing task. Test cases have their own kinds that depend on the type of testing, functional, UI, physical, logical cases, etc. Test cases compare available resources and current conditions with desired outcomes and determine if the functionality can be released or not.

Traceability Matrix:

This software testing documentation maps test cases and their requirements. All entries have their custom IDs team members and stakeholders can track the progress of any tasks by simply entering its ID to the search. The combination of internal and external documentation is the key to a deep understanding of all testing processes. Although stakeholders typically have access to the majority of documentation, they mostly work with external files, since they are more concise and tackle tangible issues and results. Internal files, on the other hand, are used by team members to optimize the testing process.

6.1.1 Unit Testing

Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

A unit can be almost anything you want it to be -- a line of code, a method, or a class. Generally though, smaller is better. Smaller tests give you a much more granular view of how your code is performing. There is also the practical aspect that when you test very small units, your tests can be run fast; like a thousand tests in a second fast. Unit Testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test drove development TDD or test-first development).

6.1.2. Black Box Testing

During functional testing, testers verify the app features against the user specifications. This is completely different from testing done by developers which is unit testing. It checks whether the code works as expected. Because unit testing focuses on the internal structure of the code, it is called the white box testing. On the other hand, functional testing checks apps functionalities without looking at the internal structure of the code, hence it is called black box testing. Despite how flawless the various individual code components may be, it is essential to check that the app is functioning as expected, when all components are combined. Here you can find a detailed comparison between functional testing vs unit testing.

6.1.3. Integration Testing

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems. Integration testing is a key aspect of software testing.

Integration tests determine if independently developed units of software work correctly when they are connected to each other. The term has become blurred even by the diffuse standards of the software industry, so I've been wary of using it in my writing. In particular, many people assume integration tests are necessarily broad in scope, while they can be more effectively done with a narrower scope. In a larger project, we would have a design phase that would specify the interface and behavior of the various modules in the system. Modules would then be assigned to developers to program. It was not unusual for one programmer to be responsible for a single module, but this would be big enough that it could take months to build it. All this work was done in isolation, and when the programmer believed it was finished they would hand it over to QA for testing.

6.1.4. System Testing

System testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the systems compliance with the specified requirements. System Testing means testing the system as a whole. All the modules/components are integrated in order to verify if the system works as expected or not. System Testing is done after Integration Testing. This plays an important role in delivering a high-quality product. System testing is a method of monitoring and assessing the behaviour of the complete and fully-integrated software product or system, on the basis of pre-decided specifications and functional requirements. It is a solution to the question "whether the complete system functions in accordance to its pre-defined requirements?"

System testing is a process of testing the entire system that is fully functional, in order to ensure the system is bound to all the requirements provided by the client in the form of the functional specification or system specification documentation. In most cases, it is done next to the Integration testing, as this testing should be covering the end-to-end systems actual routine. This type of testing requires a dedicated Test Plan and other test documentation derived from the system specification document that should cover both software and hardware requirements. By this test, we uncover the errors. It ensures that all the system works as expected. We check System performance and functionality to get a quality product. System testing is nothing but testing the system as a whole. This testing checks complete end-to-end scenario as per the customer’s point of view. Functional and Non-Functional tests also done by System testing. All things are done to maintain trust within the development that the system is defect-free and bug-free. System testing is also intended to test hardware/software requirements specifications. System testing is more of a limited type of testing.

6.1.5. Regression Testing

Regression Testing is a Software Testing type in which test cases are re-executed in order to check whether the previous functionality of the application is working fine and the new changes have not introduced any new bugs. This test can be performed on a new build when there is a significant change in the original functionality that too even in a single bug fix. For regression testing to be effective, it needs to be seen as one part of a comprehensive testing methodology that is cost-effective and efficient. Regression Testing is a type of testing that is done to verify that a code change in the software does not impact the existing functionality of the product. This is to make sure the product works fine with new functionality, bug fixes or any change in the existing feature.

CHAPTER 7

PERFORMANCE AND LIMITATION

7.1 PERFORMANCE

- High accuracy in identifying individuals.
- Robust against variations in pose, lighting, and occlusions.
- Efficient for real-time or near real-time identification.
- Scalable for large populations.

7.2 LIMITATION

- Challenges in collecting diverse and comprehensive training data.
- Privacy concerns and the need for data protection measures.
- Performance dependent on the quality of the sensor used.

7.3 CONCLUSION AND FUTURE ENHANCEMENT

This article discusses a model-based approach as well as a machine learning approach for ear recognition with bilateral symmetry, gender categorization, and ear recognition with bilateral symmetry, gender classification, and ear recognition, respectively. Our findings provide 100% performance, confirming symmetry. We are the first to employ deep learning for bilaterally symmetric ear recognition.

In the process, we also consider ear characteristics that are crucial for ear recognition, gender distinction, and ear bilateral symmetry. The heatmaps displayed in this article indicate that the centre region is crucial for bilateral symmetry and ear recognition. Also, we compare the heatmaps for ear recognition and gender classification. It seems that gender is largely determined by the upper helix and the lobe. Yet, neither bilateral symmetry nor ear recognition are considerably aided by the lobe. Also, our research on bilateral ear symmetry demonstrates that jewellery has no effect on the rate of recognition. Moreover, the insights drawn from the heatmap analysis present intriguing prospects for future investigations. The emphasis on the central region's significance for bilateral symmetry and ear recognition opens the door to focused feature extraction from this area. Additionally, the divergence in heatmaps between ear recognition and gender classification, indicating distinct influential regions, sparks curiosity for the underlying biological or anatomical reasons driving such distinctions. Future studies could unravel these intricacies, potentially refining the accuracy of gender classification and ear recognition tasks. By combining model-based and machine learning approaches, unveiling the significance of specific ear regions, and investigating the influence of jewellery, our work not only contributes to current knowledge but also paves the way for more sophisticated, accurate, and robust ear-based recognition systems in the future.

CHAPTER 8

APPENDICES

8.1 SAMPLE SCREENS

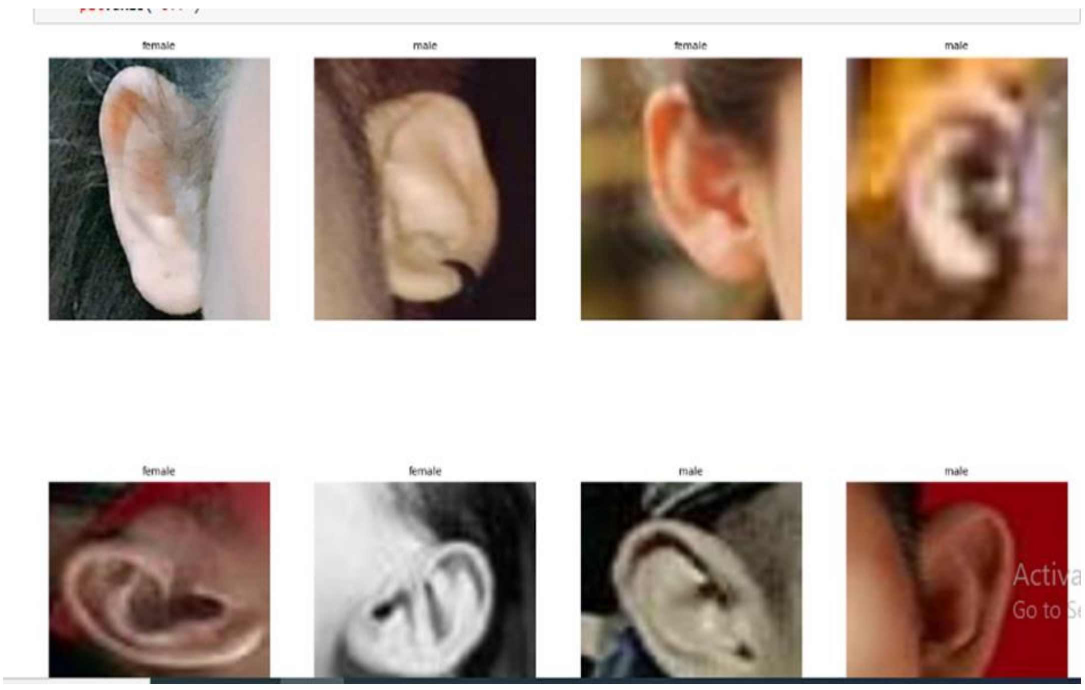
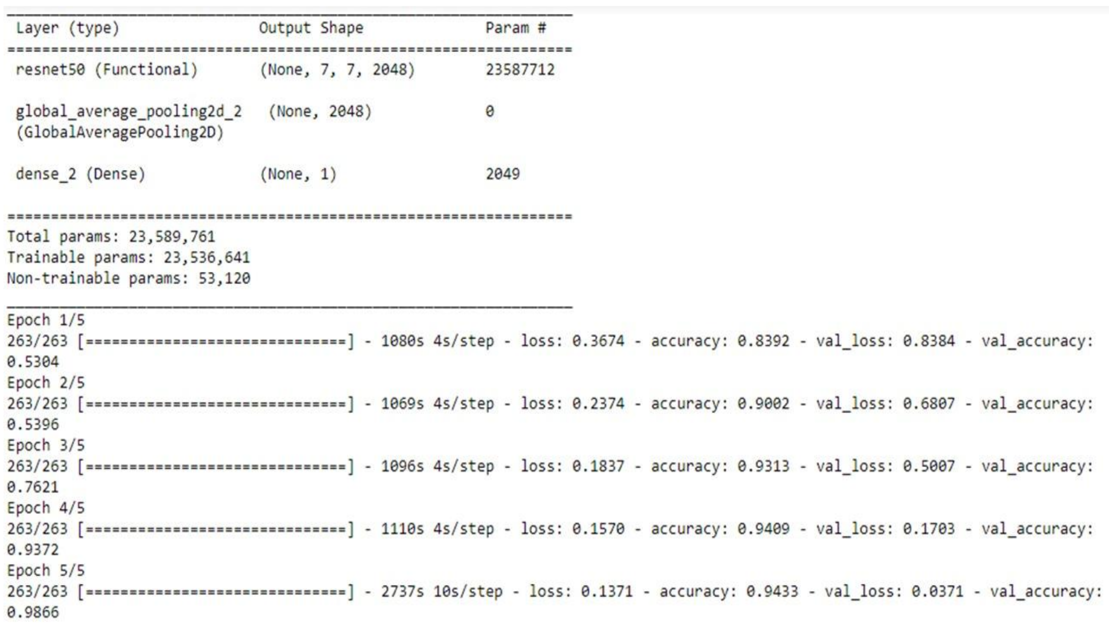


Figure 8.1 Data Collection



```
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d_2   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_2 (Dense)             (None, 1)                 2049

=================================================================
Total params: 23,589,761
Trainable params: 23,536,641
Non-trainable params: 53,120
_________________________________________________________________
Epoch 1/5
263/263 [==============================] - 1080s 4s/step - loss: 0.3674 - accuracy: 0.8392 - val_loss: 0.8384 - val_accuracy:
0.5304
Epoch 2/5
263/263 [==============================] - 1069s 4s/step - loss: 0.2374 - accuracy: 0.9002 - val_loss: 0.6807 - val_accuracy:
0.5396
Epoch 3/5
263/263 [==============================] - 1096s 4s/step - loss: 0.1837 - accuracy: 0.9313 - val_loss: 0.5007 - val_accuracy:
0.7621
Epoch 4/5
263/263 [==============================] - 1110s 4s/step - loss: 0.1570 - accuracy: 0.9409 - val_loss: 0.1703 - val_accuracy:
0.9372
Epoch 5/5
263/263 [==============================] - 2737s 10s/step - loss: 0.1371 - accuracy: 0.9433 - val_loss: 0.0371 - val_accuracy:
0.9866
```

Figure 8.2 Training And Validation

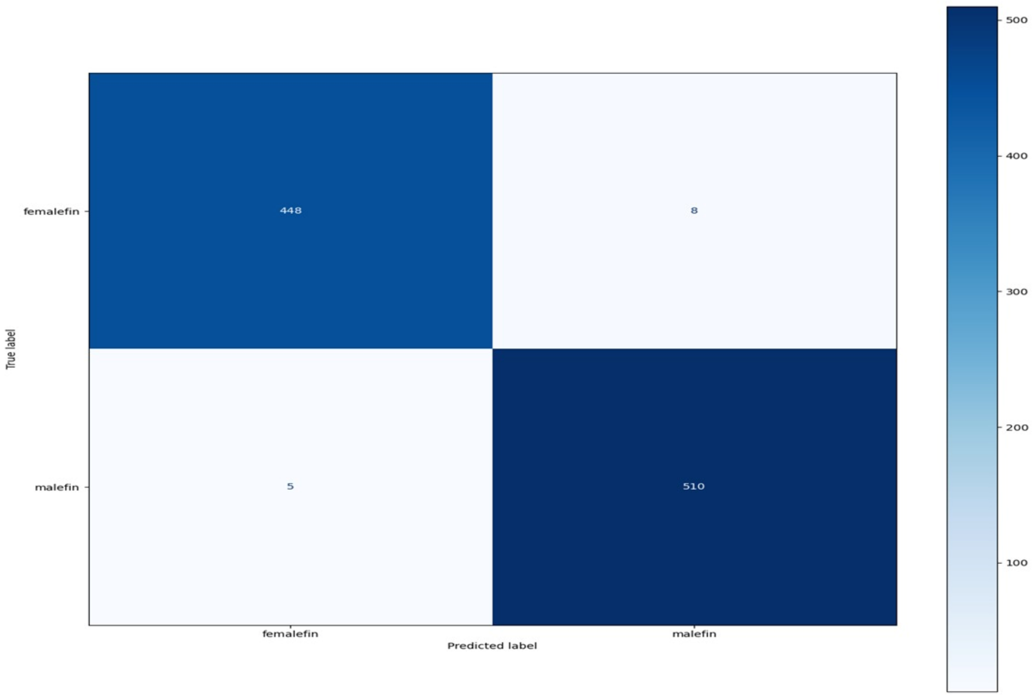


Figure 8.3 Confusion Matrix

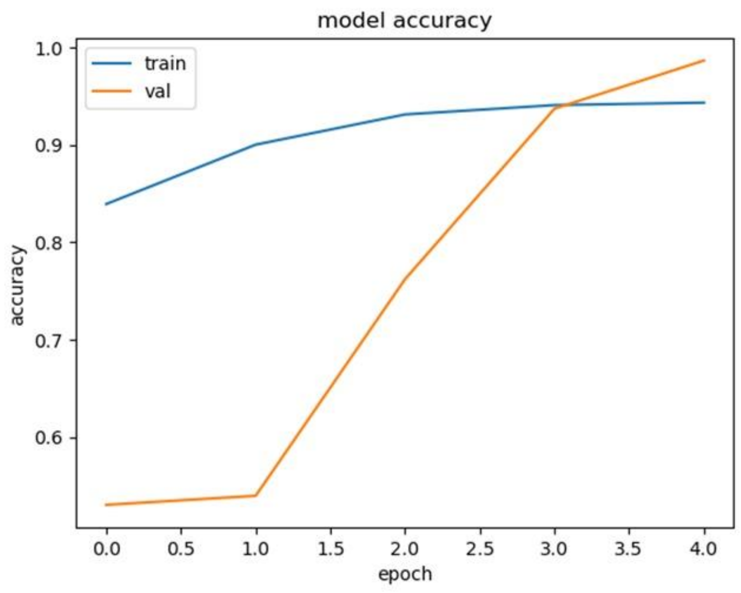


Figure 8.4 Accuracy Prediction

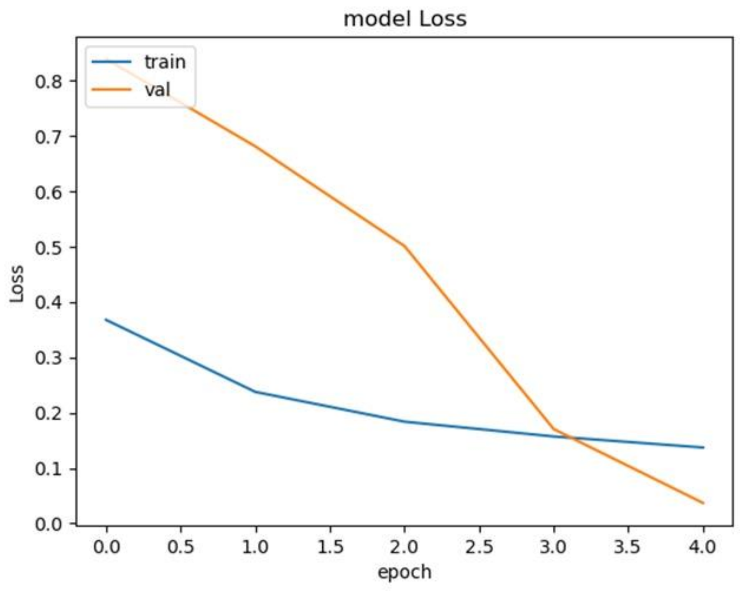
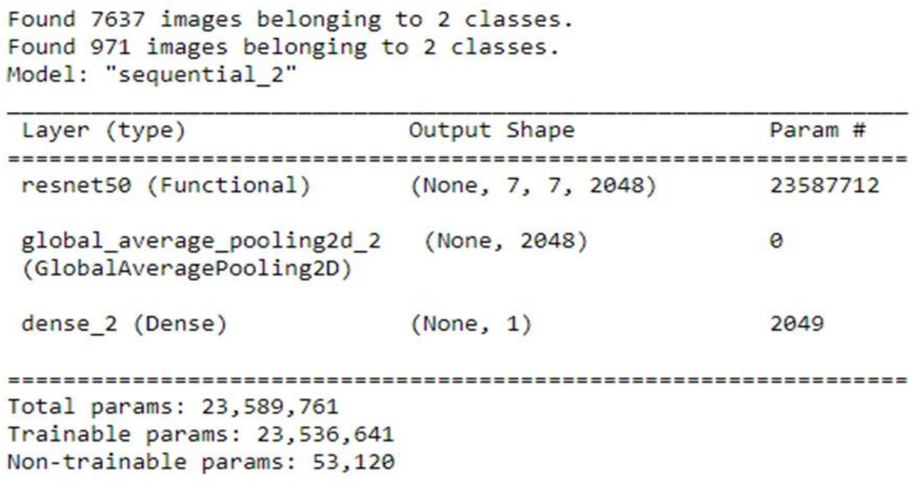
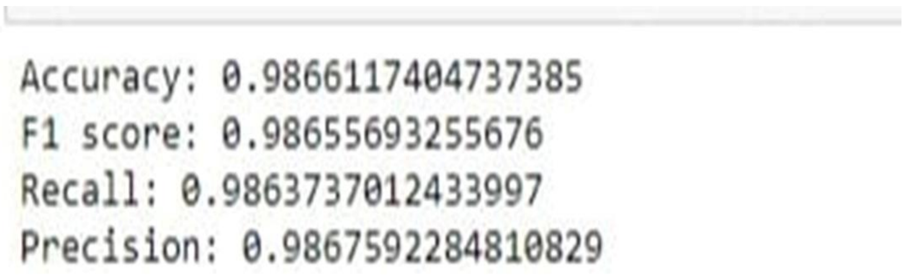


Figure 8.5 Loss Prediction



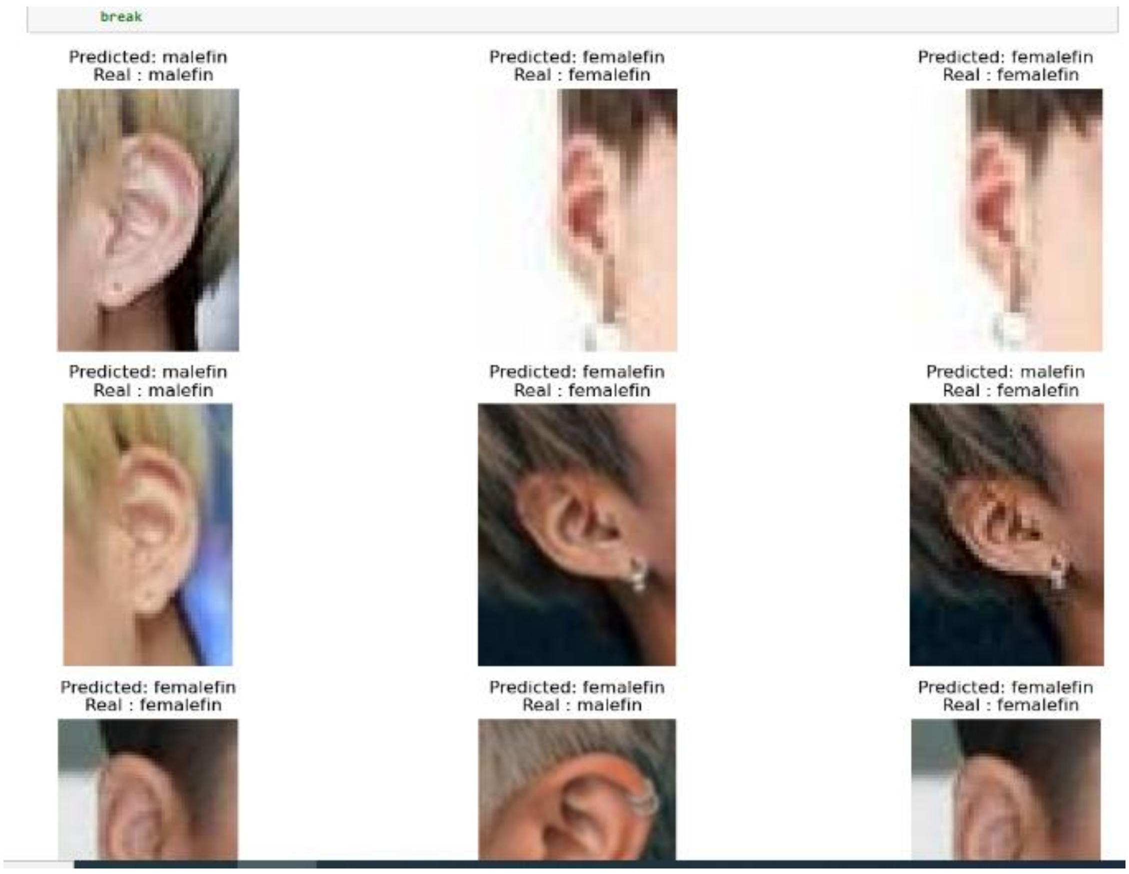
```
Found 7637 images belonging to 2 classes.
Found 971 images belonging to 2 classes.
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d_2   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_2 (Dense)             (None, 1)                 2049

=================================================================
Total params: 23,589,761
Trainable params: 23,536,641
Non-trainable params: 53,120
```

Figure 8.6 Model Prediction



```
Accuracy: 0.9866117404737385
F1 score: 0.98655693255676
Recall: 0.9863737012433997
Precision: 0.9867592284810829
```

Figure 8.7 Evaluation



```
        break
```

Predicted: malefin
Real : malefin

Predicted: femalefin
Real : femalefin

Predicted: femalefin
Real : femalefin

Predicted: malefin
Real : malefin

Predicted: femalefin
Real : femalefin

Predicted: malefin
Real : femalefin

Predicted: femalefin
Real : femalefin

Predicted: femalefin
Real : malefin

Predicted: femalefin
Real : femalefin

Figure 8.8 Result Analysis

REFERENCES

1. Abaza, A. Ross, C. Hebert, M. A. F. Harrison and M. S. Nixon, "A survey on ear biometrics", ACM Comput. Surveys, vol. 45, no. 2, pp. 1-35, 2013.
2. V. Iannarelli, Forensic Identification Series: Ear Identification, Fremont, CA, USA:Paramont, 1989.
3. Bertillon, La Photographie Judiciaire: Avec Un Appendice Sur La Classification Et L'Identification Anthropométriques, Paris, France:Gauthier-Villars, 1890.
4. M. S. Nixon, P. L. Correia, K. Nasrollahi, T. B. Moeslund, A. Hadid and M. Tistarelli, "On soft biometrics", Pattern Recognit. Lett., vol. 68, pp. 218-230, Dec. 2015.
5. R. Khorsandi and M. Abdel-Mottaleb, "Gender classification using 2-D ear images and sparse representation", Proc. IEEE Workshop Appl. Comput. Vision (WACV), pp. 461-466, 2013.
6. J. Lei, J. Zhou and M. Abdel-Mottaleb, "Gender classification using automatically detected and aligned 3D ear range data", Proc. Int. Conf. Biometrics (ICB), pp. 1-7, 2013.
7. D. Yaman, F. I. Eyiokur and H. K. Ekenel, "Multimodal age and gender classification using ear and profile face images", Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, pp. 2414-2421, 2019.
8. D. Meng, M. S. Nixon and S. Mahmoodi, "Gender and kinship by model-based ear biometrics", Proc. Int. Conf. Biometrics Special Interest Group (BIOSIG), pp. 1-5, 2019.
9. P. Yan and K. W. Bowyer, "Biometric recognition using 3D ear shape", IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 8, pp. 1297-1308, Aug. 2007.
10. E. Alqaralleh and A. Afaneh, "Symmetric ear and profile face fusion for identical twins and non-twins recognition", Signal Image Video Process., vol. 12, no. 6, pp. 1157-1164, 2018.
11. K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 770-778, 2016.

12. R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization", Proc. IEEE Int. Conf. Comput. Vis., pp. 618-626, 2017.
13. M. S. Nixon and A. S. Aguado, Feature Extraction and Image Processing for Computer Vision, Amsterdam, The Netherlands:Academic, 2019.
14. I. Ganapathi, S. S. Ali and S. Prakash, "Geometric statistics-based descriptor for 3D ear recognition", Visual Comput., vol. 36, no. 1, pp. 161-173, 2020.
15. D. Meng, S. Mahmoodi and M. S. Nixon, "Which ear regions contribute to identification and to gender classification?", Proc. 8th Int. Workshop Biometrics Forensics (IWBF), pp. 1-6, 2020.
16. Introduction to USTB Ear Image Databases. Accessed: Jan. 28, 2020.[Online]. Available: http://www.ustb.edu.cn/resb/en/doc/Imagedb_123_intro_en.pdf
17. K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre, “XM2VTSDB: The extended M2VTS database,” in Proc. 2nd Int. Conf. Audio Video Based Biometrics Person Authentication, vol. 964, 1999, pp. 965–966.
18. D. Meng, S. Mahmoodi, and M. S. Nixon, “Which ear regions contribute to identification and to gender classification?” in Proc. 8th Int. Workshop Biometrics Forensics (IWBF), 2020, pp. 1–6.
19. B. Arbab-Zavar and M. S. Nixon, “On shape-mediated enrolment in ear biometrics,” in Proc. Int. Symp. Visual Comput., 2007, pp. 549–558.
20. S. Yoga, J. Balaih, V. Rangdhol, S. Vandana, S. Paulose, and L. Kavya,“Assessment of age changes and gender differences based on anthropometric measurements of ear: A cross-sectional study,” J. Adv. Clin. Res.Insights, vol. 4, no. 4, pp. 92–95, 2017.
21. dm-vlc/dataset. Accessed: Jun. 27, 2020. [Online]. Available:
22. https://github.com/dm-vlc/dataset/find/master
23. Ear Recognition Research. Accessed: Sep. 18, 2020. [Online]. Available:
24. http://awe.fri.uni-lj.si/
25. Ž. Emeršic, V. Štruc, and P. Peer, “Ear recognition: More than a survey,” Neurocomputing, vol. 255, pp. 26–39, Sep. 2017.
26. D. J. Hurley, M. S. Nixon, and J. N. Carter, “Force field feature extraction for ear biometrics,” Comput. Vis. Image Understanding, vol. 98,no. 3, pp. 491–512, 2005.

27. R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh,and D. Batra, “Grad-CAM: Visual explanations from deep networks via gradient-based localization,” in Proc. IEEE Int. Conf. Comput. Vis.,2017, pp. 618–626.
28. SCface—Surveillance Cameras Face Database. Accessed: Jan. 28, 2020.[Online]. Available: http://www.scface.org/
29. F. I. Eyiokur, D. Yaman, and H. K. Ekenel, “Domain adaptation for ear
    a. recognition using deep convolutional neural networks,” IET Biometrics,vol. 7, no. 3, pp. 199–206, 2018.
30. I. I. Ganapathi, S. S. Ali, and S. Prakash, “Geometric statistics-based descriptor for 3D ear recognition,” Visual Comput., vol. 36, no. 1, pp. 161–173, 2020.
31. B. Arbab-Zavar, M. S. Nixon, and D. J. Hurley, “On model-based analysis of ear biometrics,” in Proc. 1st IEEE Int. Conf. Biometrics Theory Appl. Syst., 2007, pp. 1–5.
32. Z. Song, S. Zhou, and J. Guan, “A novel image registration algorithm
    a. for remote sensing under affine transformation,” IEEE Trans. Geosci.
    b. Remote Sens., vol. 52, no. 8, pp. 4895–4912, Aug. 2014.