**MINOR PROJECT REPORT**

**ON**

**Age And Gender Detection Using CNN**

*Submitted in partial fulfillment of the requirements for the award of the degree of*

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING(CSE)

BY

**K.POOJA 22P65A0511**

**K.RATHIMA 22P65A0512**

**K.NISHA 22P65A0508**

*Under the esteemed guidance of*

***Mrs. KUMARI JELLI***

***Assistant Professor***

***Dept. of CSE***



##### VIGNANA BHARATHI INSTITUTE OF TECHNOLOGY

(A UGC Autonomous Institution, Approved by AICTE, Affiliated to JNTUH,Accredited by NBA & NAAC) Aushapur (V), Ghatkesar (M), Medchal (Dt.) Telangana-501301

2024-2025



*Department of Computer Science and Engineering*

***CERTIFICATE***

*This Is To Certify That The Minor Project Titled* ***" Age And Gender Detection Using CNN”*** *Submitted By K.Pooja(22p65a0511),**K.Rathima(22p65a0512),**K.Nisha(22P65A0508) in B.Tech IV-I semester Computer Science & Engineering(CSE) is a record of the Bonafide work carried out by them*

*The results embodied in this report have not been submitted to any other university for the award of any degree*

##### 

**HEAD OF THE DEPARTMENT:**

*Dr. M.Venkateswara Rao*

*Professor, Dept. of CSE*

**INTERNAL GUIDE:**

Mrs. Chaitanyasri mouli

(Associate Professor)

**DECLARATION**

We, **K.Pooja, K.Rathima, K.Nisha**, bearing hall ticket numbers **22P65A0511, 22P65A0512, 22P65A0508** respectively, hereby declare that the mini project report titled **“*Age And Gender Detection Using CNN*”**, carried out under the guidance of **Mrs. Chaitanyasri mouli**, Assistant Professor, Department of Computer Science and Engineering (CSE), Vignana Bharathi Institute of Technology, Hyderabad, has been submitted to **Jawaharlal Nehru Technological University Hyderabad, Kukatpally**, in partial fulfillment of the requirements for the award of the **Bachelor of Technology** degree in Computer Science and Engineering (CSE).

This report is a record of bonafide work carried out by us, and the results presented in this project are original and have not been reproduced or copied from any source. Furthermore, the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

**K.Pooja(22P65A0511)**  
**K.Rathima(22P65A0512)**  
**K.Nisha(22p65a0508)**

**ACKNOWLEDGEMENT**

Self-confidence, hard work, commitment, and meticulous planning are crucial for successfully completing any task. However, these qualities bear fruit only when paired with the right opportunities.

I sincerely express my heartfelt gratitude to **Dr. P. V. S. Srinivas**, Principal, and **Dr. M. Venkateswara Rao**, Head of the Department of Computer Science and Engineering, for their unwavering encouragement, support, and guidance throughout the course of this technical seminar.

I extend my profound thanks to my Project Guide, **Mrs. Chaitanyasri mouli**, Associate Professor, for her invaluable assistance, thoughtful guidance, and constant support in completing this seminar successfully within the stipulated time.

I also express my deep gratitude to **Mrs. Chaitanyasri mouli**, Associate Professor and Technical Paper Presentation Incharge, for her valuable suggestions and insights, which proved instrumental in shaping this seminar.

Additionally, I acknowledge with appreciation the contributions of the authors and researchers whose references and literature provided the foundation for this seminar.

Finally, I am deeply thankful to my parents and friends for their encouragement and unwavering support, guiding me through every step whenever I needed it. Their belief in me has been a constant source of inspiration.

**ABSTRACT**

Age and gender detection is a fundamental task in computer vision, with wide-ranging applications in security systems, targeted advertising, and human-computer interaction. This study employs Convolutional Neural Networks (CNNs), leveraging their ability to automatically extract meaningful features from facial images to classify age groups and gender effectively. Our approach involves preprocessing the data using normalization techniques and data augmentation to ensure robustness against variations in lighting, pose, and facial expressions. The CNN model incorporates multiple convolutional and pooling layers, followed by fully connected layers, and utilizes activation functions and dropout to enhance learning efficiency and reduce overfitting.The model was trained and evaluated on publicly available datasets, demonstrating high accuracy and generalization across diverse demographics. Comparisons with traditional methods underscore the superiority of CNNs in handling complex variations in image data. The proposed system offers a scalable, real-time solution for applications such as demographic analytics, surveillance, and customer interaction systems, paving the way for enhanced automated demographic analysis.

***Keywords:***

*Keywords : Face Detection, Skin Colour Segmentation, Face Features extraction, Feature's recognition, Fuzzy rules.*

**CHAPTER 1 INTRODUCTION**

**INTRODUCTION**

* 1. **Introduction to the System**

Age and gender detection are fundamental yet challenging tasks in computer vision, holding significant importance in applications such as identity verification, personalized marketing, surveillance, and social behavior analysis. These demographic traits provide critical insights into user characteristics, enabling systems to deliver tailored experiences and enhance decision-making processes. However, accurate detection remains challenging due to variations in facial attributes caused by ethnicity, lighting conditions, occlusions, makeup, and facial expressions.

Traditional machine learning methods relied on handcrafted features, such as texture descriptors or geometric measurements, to perform age and gender classification. While effective in controlled environments, these approaches struggle to generalize across diverse datasets and dynamic real-world conditions. The advent of deep learning, particularly Convolutional Neural Networks (CNNs), has brought significant advancements in image-based tasks by enabling automated feature extraction and hierarchical learning. CNNs are well-suited for age and gender detection as they efficiently capture spatial and semantic features in facial images.

This study presents a CNN-based framework for age and gender classification, aiming to address the challenges of variability and complexity in facial data. The approach involves data preprocessing techniques such as normalization and augmentation to ensure robustness against real-world inconsistencies. The proposed model comprises multiple convolutional and pooling layers to extract features, followed by dense layers for classification. By utilizing dropout and regularization, the model mitigates overfitting and achieves superior generalization.

* 1. **PROBLEM STATEMENT:**

Facial identification in real-world situations faces issues like partially hidden faces, non-frontal angles, and multiple people in one image, which can hurt accuracy and reliability. To improve performance in these settings, we need to develop strong algorithms that can effectively process this challenging visual data.

**1.3 OBJECTIVE**

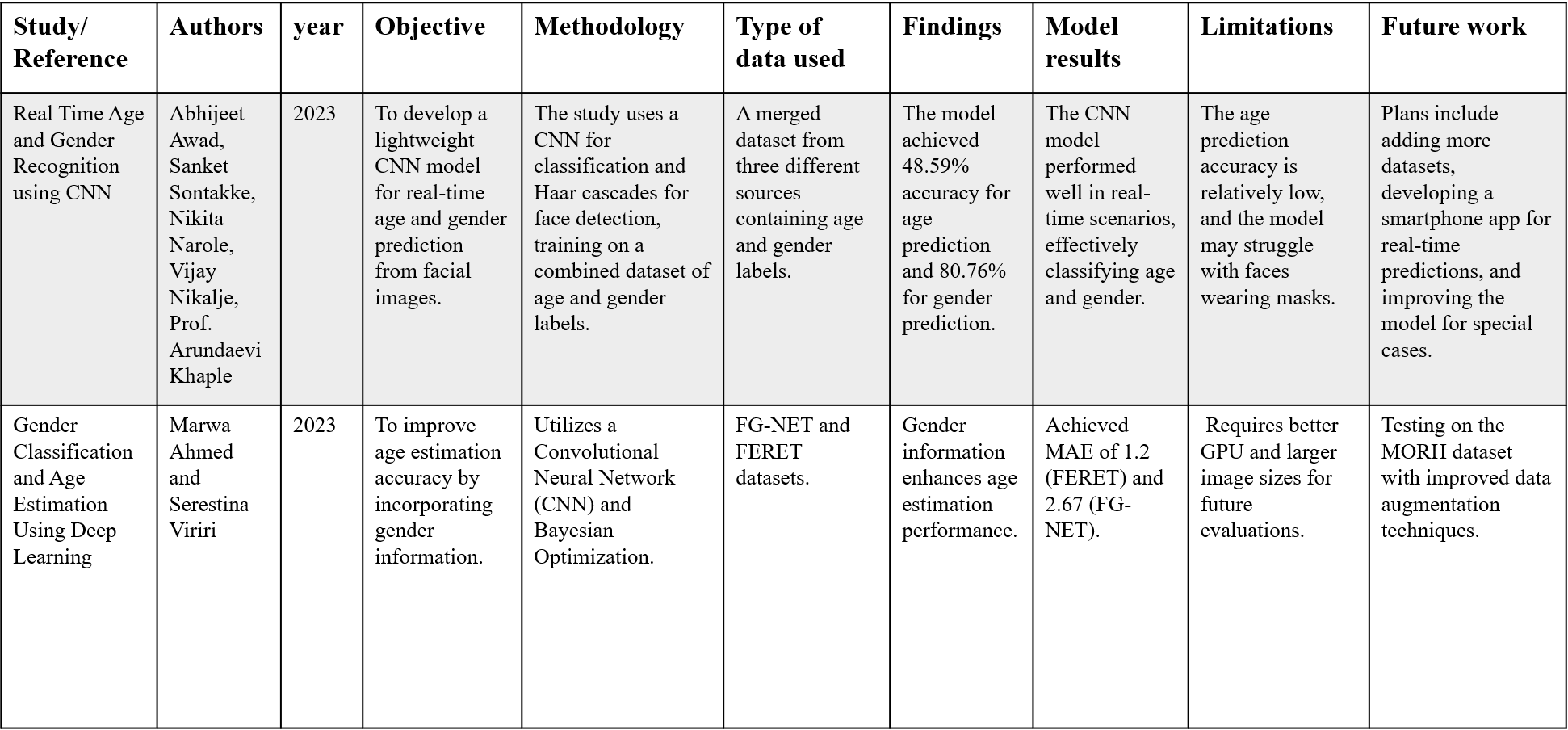
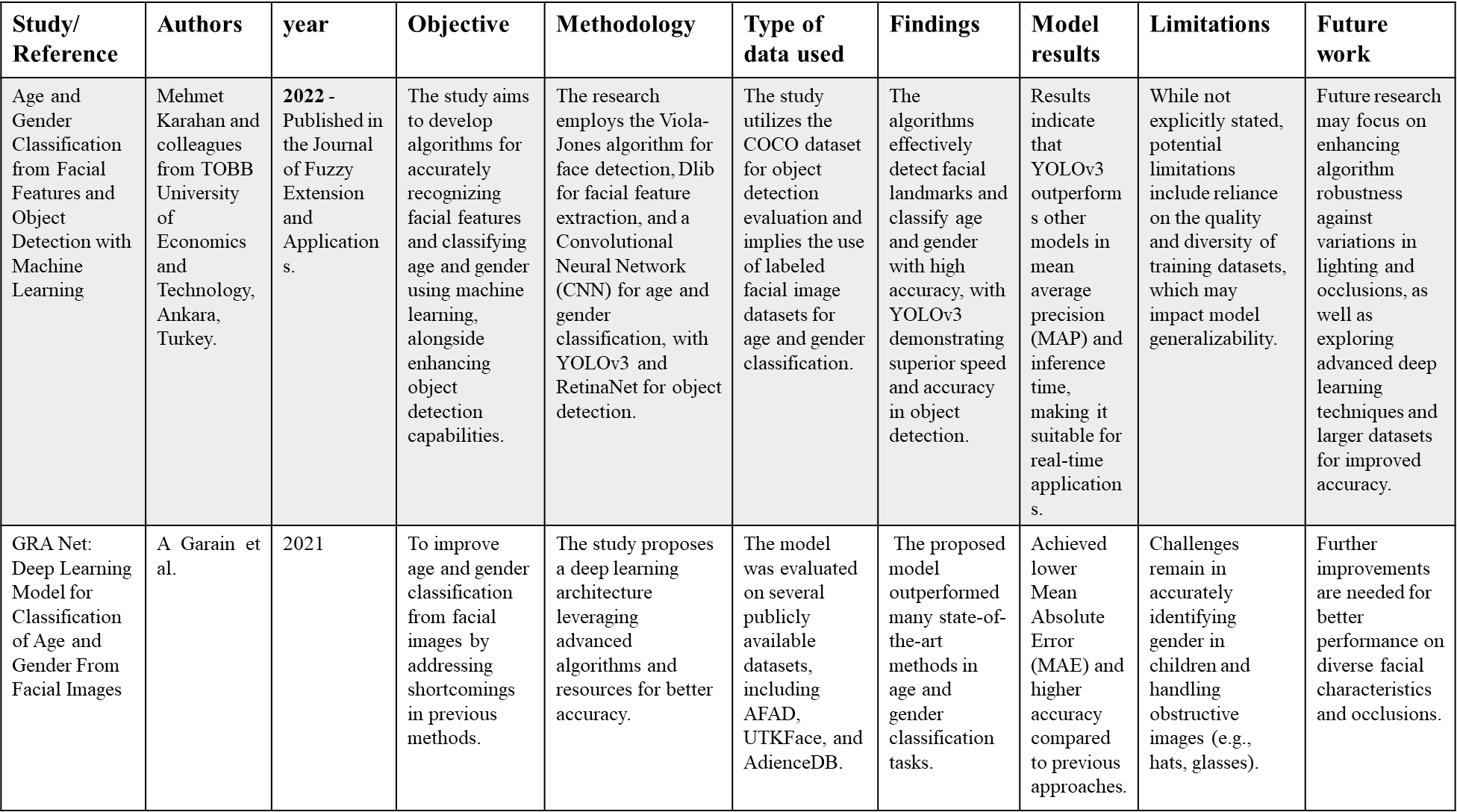
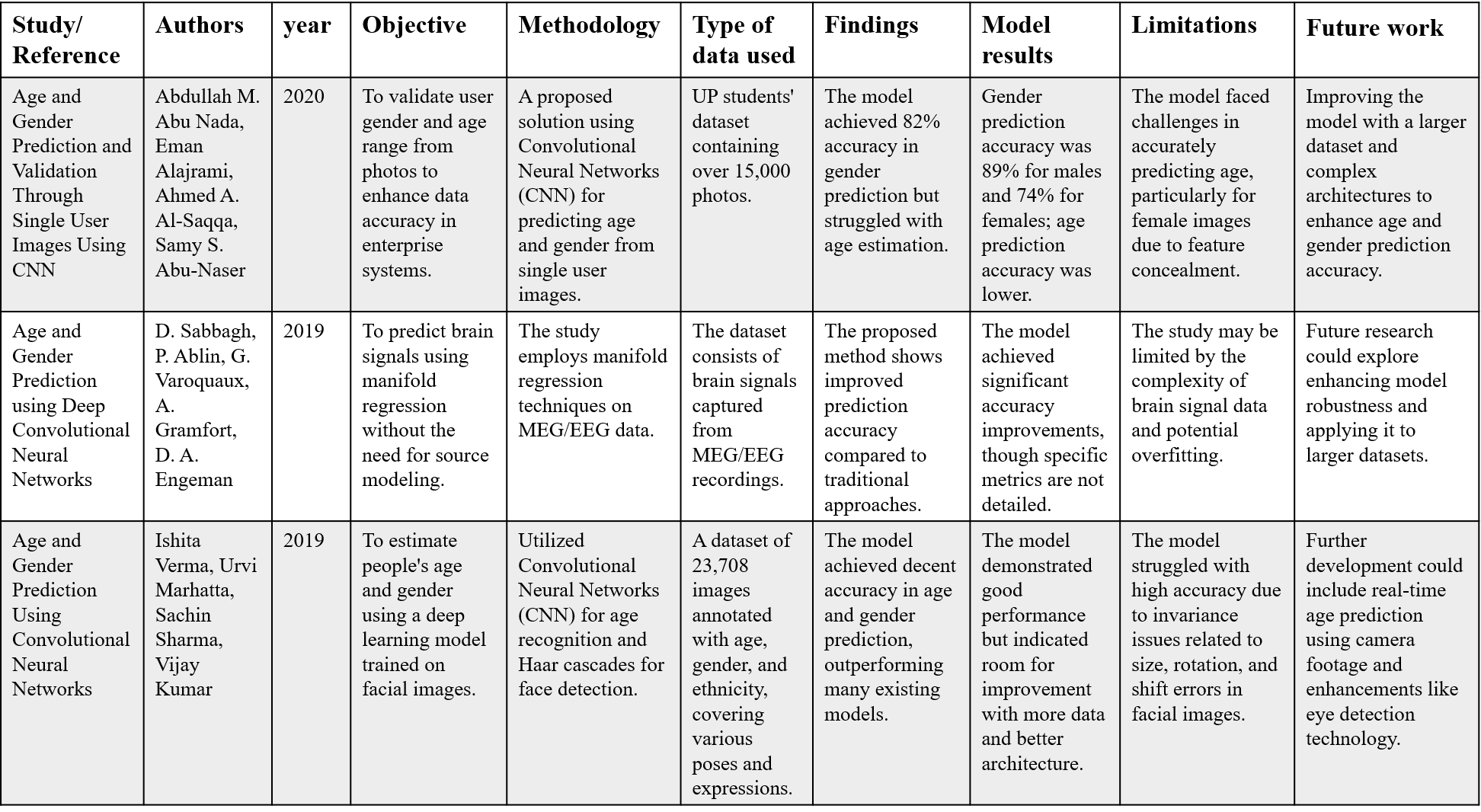
Develop a system that use a CNN model to accurately determine the age group and gender of human faces used as input images.

**1.4 AIM OF THE PROJECT:**

The goal of this research is to use Convolutional Neural Networks (CNNs) to create an accurate and effective age and gender detection system for real-time forecasts and useful applications in a variety of fields.

**CHAPTER 2 LITERATURE SURVEY**

**2.LITERATURE SURVEY:**

**2.1 Existing System**

There are several existing systems for age and gender detection that use various techniques and algorithms to achieve accurate results. Some of the popular techniques used in existing systems include deep learning, support vector machines (SVMs), and AdaBoost.

Many existing systems suffer from accuracy issues when dealing with complex images.

**Limited accuracy** - Traditional methods for age and gender detection, such as using handcrafted features and machine learning classifiers, have limited accuracy. These methods rely on human expertise to identify the relevant features and patterns for age and gender detection, which may not capture the full complexity and variability of the data.

**1.Deep Learning Approaches:** Modern systems primarily use Convolutional Neural Networks (CNNs) for age and gender detection due to their ability to automatically learn complex features from images, outperforming traditional methods.

**2.Traditional Machine Learning Methods:** Older systems rely on algorithms like Support Vector Machines (SVMs) and AdaBoost, using handcrafted features for classification

**3.Accuracy Challenges:** While CNN-based models have improved accuracy, they still face challenges, particularly with diverse demographic groups and varying image conditions such as lighting or angle.

**4.Real-Time Detection:** Some systems aim for real-time performance by balancing model complexity with processing speed, crucial for applications like surveillance.

**5.Applications:** Age and gender detection systems are used in fields such as security, personalized marketing, and healthcare.

**2.2 Proposed System**

Age and gender detection using CNN is designed to overcome the limitations of existing systems and achieve high accuracy and efficiency in real-world scenarios. The proposed system uses a deep learning approach, specifically a Convolutional Neural Network (CNN), for age and gender detection. The CNN is trained on facial images that are labeled with age and gender information.

The CNN architecture consists of several layers, including convolutional layers, pooling layers, and fully connected layers. The input image is passed through the convolutional layers, which extract the features of the image at different levels of abstraction. The pooling layers then downsample the features to reduce the computational complexity. Finally, the fully connected layers classify the features into the corresponding age and gender labels. To train the CNN, a loss function is defined that measures the difference between the predicted age and gender labels

**Proposed Methodology**

**Data Collection**: Gather a large, diverse dataset of facial images labeled with age and gender information.

**Preprocessing**: Process the images by detecting and localizing faces, aligning them to a consistent size and orientation, and applying data augmentation techniques (e.g., rotations, scaling) to expand the dataset.

**Model Training**: Train a Convolutional Neural Network (CNN) on the preprocessed images

**Evaluation**: Test the trained model using a separate dataset to measure performance

**2.3 Scope of the Project**

The scope of the project on age and gender detection using Convolutional Neural Networks (CNNs) includes the following:

**1.Development of a Robust Model:** The project aims to design a deep learning model using CNN architectures (such as VGGNet, ResNet) that can accurately predict age and gender from facial images. This involves selecting and preprocessing appropriate datasets, training the model, and fine-tuning for optimal performance.

**2.Real-time Application:** The project will explore the ability to implement the model for real-time age and gender detection, ensuring it works efficiently with different types of input data (e.g., images, videos) in real-world scenarios.

**3.Evaluation of Accuracy and Efficiency**: The model will be evaluated based on accuracy (for gender detection) and error metrics (such as Mean Absolute Error for age), addressing challenges like variations in age ranges, diverse facial features, and environmental conditions like lighting and angles.

**4.Wide Application Potential**: The scope includes exploring the practical applications of the technology in various domains, such as security surveillance, user authentication, personalized marketing, healthcare, and age verification.

**5.Exploring Improvements and Optimization:** The project will also look at ways to optimize processing speed without compromising accuracy, focusing on real-time performance and computational efficiency, especially for resource-constrained environments like mobile devices or embedded systems.

**CHAPTER 3**

**ANALYSIS**

**3.ANALYSIS:**

The analysis phase focuses on understanding the problem domain, identifying requirements, and evaluating the feasibility of the proposed **Age And Gender Detection Using CNN**. This includes a detailed **Feasibility Study** to assess the technical, operational, and economic aspects of the project.

**3.1 Feasibility Study**

The feasibility study evaluates whether the project is practical and achievable within the given constraints. It involves the following key aspects:

**3.1.1 Technical Feasibility**

**CNN Architecture:** Implementing Convolutional Neural Networks (CNNs) for age and gender detection is technically feasible, as CNNs are proven to be highly effective for image classification tasks. Well-established architectures like VGGNet, ResNet, and InceptionNet can be adapted for age and gender detection, offering high accuracy in facial feature extraction.

**Data Availability:** Large-scale datasets such as IMDB-WIKI, UTKFace, and Adience are publicly available and widely used for training age and gender detection models. These datasets provide sufficient variety in terms of age ranges, gender, and ethnicity, although challenges with dataset bias still exist.

**Computational Resources:** Implementing the project requires significant computational resources, especially for training deep learning models. High-performance GPUs are recommended for efficient training, though cloud-based services (e.g., AWS, Google Cloud) provide affordable alternatives for individuals and small teams.

**3.1.2 Operational Feasibility**

**Real-Time Detection:** Real-time age and gender detection is operationally feasible, with modern CNN models capable of delivering predictions within milliseconds once trained. The challenge lies in optimizing for various environmental conditions (e.g., lighting, facial occlusion) to ensure consistent performance.

**Integration with Existing Systems:** The system can be integrated into various real-world applications, such as security surveillance, automated customer profiling, or personalized marketing, with minimal additional operational overhead. APIs or model deployment on cloud platforms can facilitate smooth integration.

**3.1.3 Economic Feasibility**

**Cost of Implementation:** The initial costs of implementing a CNN-based age and gender detection system include computational resources for training (cloud services or dedicated GPUs) and the development tools (such as TensorFlow or PyTorch). However, these costs are relatively low compared to the long-term benefits, especially considering the availability of pre-trained models and open-source libraries.

**Scalability:** Once the model is trained, deploying it on cloud servers or edge devices is cost-effective for real-time applications like surveillance, healthcare, or personalized marketing. The system's scalability depends on optimizing it for lower-powered devices (e.g., smartphones, IoT devices), but initial development for more powerful systems is feasible.

**3.1.4 Legal and Ethical Feasibility:**

**Privacy Concerns:** The use of facial recognition and demographic prediction raises privacy concerns, especially in applications like surveillance and personal data profiling. Compliance with data protection laws (e.g., GDPR, CCPA) is critical to ensure that the system adheres to legal requirements and respects individual privacy.

**Bias and Fairness:** A key challenge is ensuring that the model is not biased against certain demographic groups. The system needs to be trained on diverse datasets and regularly tested for fairness to avoid discriminatory practices based on gender, ethnicity, or age group.

**3.1.5Market Feasibility:**

**Demand for Technology:** The demand for age and gender detection technologies is growing, especially in industries like security, marketing, healthcare, and entertainment. Applications such as targeted advertising, smart healthcare diagnostics, and automated surveillance systems present significant market potential.

**Competitor Analysis:** Many companies are already exploring or implementing similar technologies, so standing out in the market requires delivering higher accuracy, faster processing, and ethical data usage. Ensuring that the system performs reliably across a wide range of conditions will be crucial for commercial success.

**CHAPTER 4**

**HARDWARE AND SOFTWARE REQUIREMENTS**

**Hardware Requirements:**

* **Processor:** CPU
* **RAM:** 4GB or More
* **Storage:** 10GB SSD
* **GPU:** NVIDIA 4GB RTX1650

**Software Requirements:**

* **Operating System:** windows10 or above
* **Language:** python
* **Libraries and frameworks:** OpenCV, Caffe Framework , NumPy
* **IDE/Editor:** Jupyter Notebook or VS code

**CHAPTER 5**

**SYSTEM DESIGN**

**5.SYSTEM DESIGN**

The system design for age and gender detection using Convolutional Neural Networks (CNNs) involves several components that work together to accurately predict age and gender from facial images. Below is an overview of the key components in the system architecture:

**1.Input Layer (Image Capture/Acquisition):**

**Camera/Video Feed:** The system captures facial images through a high-resolution camera, which could be part of a real-time video feed or static images.

**Preprocessing:** Before feeding the images into the CNN model, they are preprocessed to ensure proper quality. This includes resizing, normalization (scaling pixel values), and facial region detection (using techniques like OpenCV or pre-trained models for face detection).

**2.Face Detection and Alignment:**

**Face Detection:** Use algorithms like Haar Cascades, HOG, or Dlib for detecting faces in images or videos. This step isolates the facial region to focus the CNN on relevant features.

**Face Alignment:** After detecting faces, the face may need alignment to ensure that key facial landmarks (eyes, nose, mouth) are positioned consistently for accurate predictions. This is especially important when dealing with various angles or orientations.

**3. Convolutional Neural Network (CNN) Model:**

**Pre-trained Model:** The system uses pre-trained CNN models like VGGNet, ResNet, or InceptionNet, trained on large facial datasets. These models are known for their efficiency in extracting features like age, gender, and other facial attributes.

**Model Training:** If required, the pre-trained model is fine-tuned using a domain-specific dataset (e.g., IMDB-WIKI, UTKFace) for better performance on age and gender detection.

**Age Detection:** Age is typically predicted as a continuous value (regression task) or grouped into age categories (classification task).

**Gender Detection:** Gender is usually predicted as a binary classification task (Male/Female), but it can also include additional categories (e.g., non-binary) in more advanced systems.

**4. Post-Processing and Results:**

**Output Interpretation:** Once the model makes predictions for age and gender, the results are processed. For age, this could be either a direct prediction (e.g., 25 years old) or categorized into age ranges (e.g., 18-24 years). Gender results are typically binary (Male/Female).

**Visualization:** The detected age and gender can be displayed as part of the system's user interface (e.g., a web app or mobile app) alongside the original image or video feed.

**5.User Interface/Interaction:**

**Web/Mobile Application:** A user-friendly interface for interacting with the system. The user can upload an image or use a live camera feed. The application will display the predicted age and gender in real-time.

**API Integration:** The system can also expose its functionality via APIs for third-party applications (e.g., for surveillance, marketing, or healthcare applications).

**6. Storage and Data Management:**

**Database:** A database (e.g., MySQL, MongoDB) can be used to store images, prediction results, and user data (if necessary for application purposes). This is especially useful for applications where predictions are logged for later analysis or monitoring.

**7. Model Deployment:**

**Cloud Deployment:** The model can be hosted on cloud platforms like AWS, Google Cloud, or Microsoft Azure, which provide easy scalability for handling large datasets and real-time predictions.

**Edge Deployment:** For real-time performance, the model can be deployed on edge devices such as smartphones or embedded systems using TensorFlow Lite or Core ML, enabling predictions without the need for a constant cloud connection.

**8. Continuous Model Improvement:**

**Model Evaluation:** Continuous evaluation of the model’s accuracy and performance is necessary, especially in real-world applications. The system should periodically retrain the model using new, diverse datasets to improve prediction accuracy and reduce bias.

**Model Monitoring:** Monitoring tools to track the system’s performance in real-time and address issues like drift or bias in predictions, ensuring that it operates correctly over time.

**System Design Flow:**

**1. Input Capture:** Camera captures image/video.

**2. Face Detection:** Detect and extract face region from the image.

**3. Preprocessing:** Resize, normalize, and align the face.

**4. CNN Processing:** The preprocessed image is fed into the CNN model for age and gender classification.

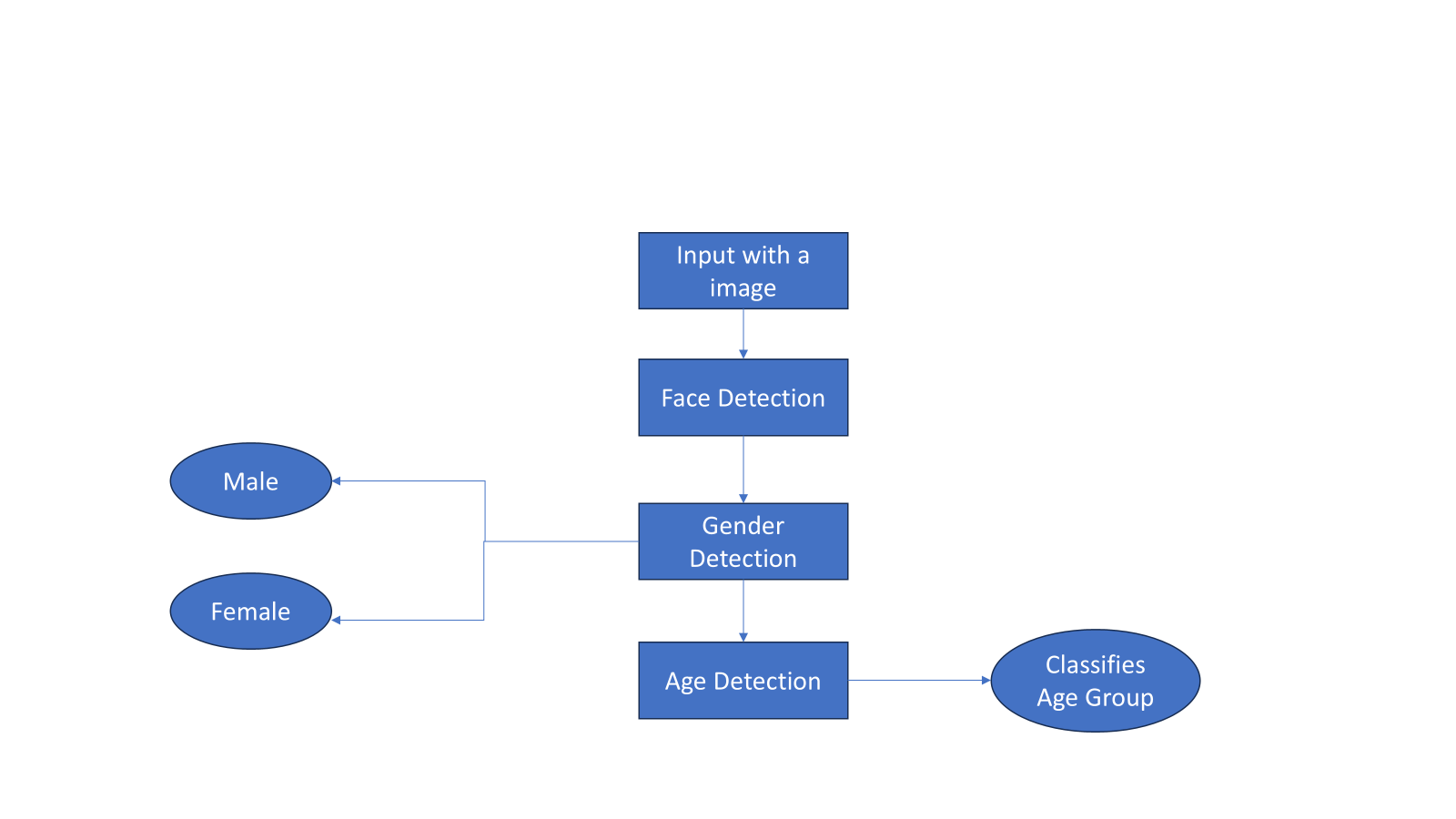
**5. Prediction Output:** Age and gender predictions are made and displayed.

**6. Post-processing:** Interpretation and display of the results.

**7. UI Interaction:** Display results on a web/mobile interface, or via an API.

**8. Storage and Logging:** Save the data and results in the database for further analysis.

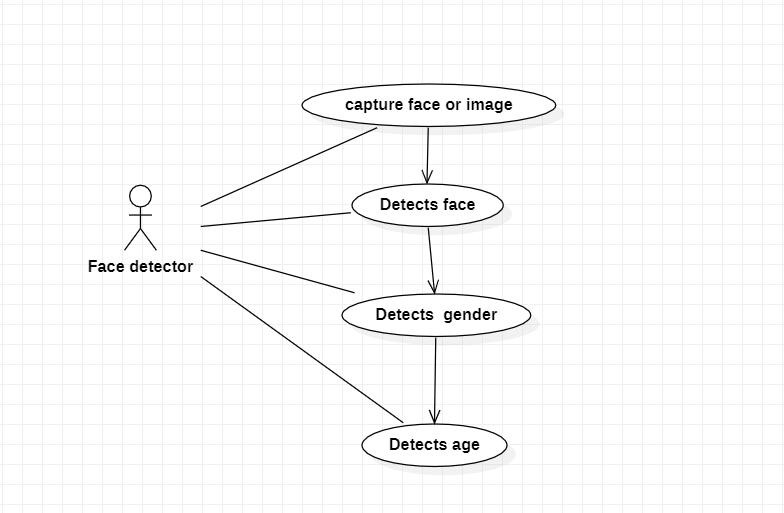
**5.2 Architecture Diagram:**



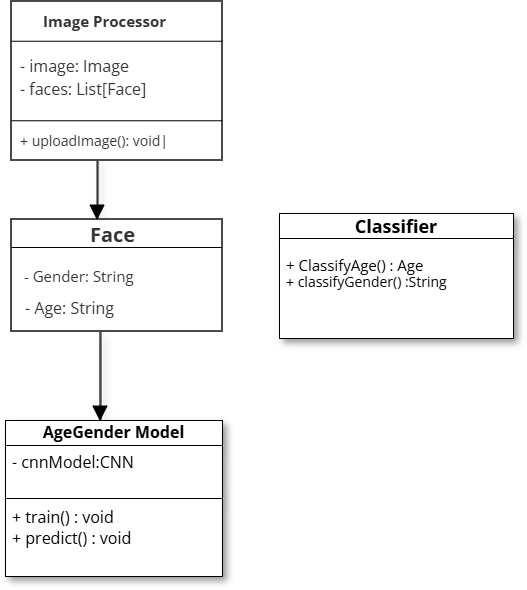
**UML DIAGRAMS:**

* Use Case Diagram
* Class Diagram
* State Diagram
* Sequence Diagram
* Component Diagram
* Deployment Diagram

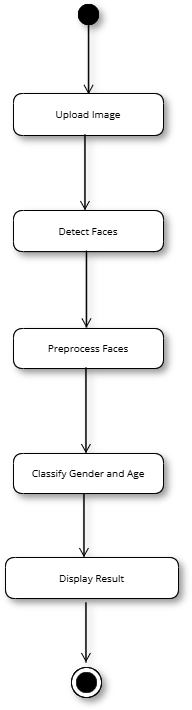
1.Use Case Diagram: A use case diagram illustrates the interactions between the users (actors) and the system (use cases) in a simplified manner. Below is a description of the use case diagram for Age and Gender Detection Using CNN.



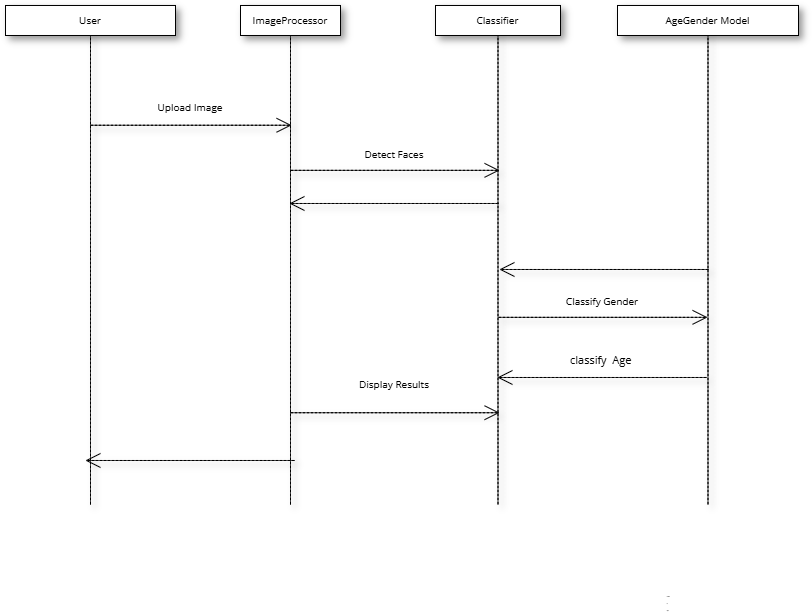
2.Class Diagram: A class diagram provides a static view of the system's structure, showing the system's classes, their attributes, methods, and relationships between them. Below is an example of a class diagram for the Age and Gender Detection system using Convolutional Neural Networks (CNN)



3.State Diagram: A state diagram (also called a state machine diagram) describes the states an object can be in and how it transitions between those states in response to various events. For the Age and Gender Detection System using CNN, the state diagram would describe the system’s behavior as it processes an image or video for detecting age and gender.



4.Sequence diagrams: A sequence diagram illustrates the sequence of interactions between various components (objects) in the system over time. It shows the order of messages exchanged between objects to achieve a specific functionality.



5.Component Diagram: A component diagram is used to model the components of a system and their relationships. It shows the structure of the system by representing the components or modules and how they interact with each other. For the Age and Gender Detection Using CNN system, the component diagram illustrates how various modules interact to perform the task of age and gender detection.

6.Deployment Diagram: Shows where and how the system is hosted and used by visualizing the physical deployment of system components, such as servers, cloud services, and user interfaces.

**CHAPTER 6**

**METHODOLOGY**

**6.METHODOLOGY:**

The methodology section outlines the approach, technologies, and implementation steps used to develop the **Age And Gender Detection Using CNN**. This includes the identification of key modules, an overview of the technologies used, and a detailed implementation plan.

**6.1 Modules**

**1.Data Collection and Preprocessing Module**

This module handles the preparation of input data for the CNN model.

**Data Collection:** Fetch datasets like Adience, IMDB-WIKI, or custom-labeled facial datasets with annotated age and gender information.

**Preprocessing:** Resize images to a standard input size (e.g., 224x224).

Normalize pixel values to a range suitable for the CNN (e.g., [0, 1]).

Detect and crop faces using tools like OpenCV or Dlib.

Apply data augmentation (rotation, scaling, flipping) to increase data diversity.

**2.Feature Extraction Module**

This is the core of the CNN architecture, where hierarchical features are extracted from input images.

**Convolutional Layers:** Detect low-level (e.g., edges) and high-level (e.g., shapes) features.

**Pooling Layers:** Reduce spatial dimensions and retain important features.

Activation Functions: Use functions like ReLU to introduce non-linearity.

**3.Classification Module**

Handles the final output of the CNN model.

**Fully Connected Layers:** Combine extracted features for classification.

**Output Layer:**

**Gender Detection:** A softmax layer with two outputs (male/female).

**Age Detection: Either:**

**Regression:** Predicts continuous age values.

**Classification:** Outputs predefined age groups (e.g., 0-18, 19-30, etc.).

**4.Training and Optimization Module**

This module focuses on model training and optimization.

**Loss Functions:**

**Categorical Cross-Entropy:** For gender classification and age group classification.

**Mean Squared Error (MSE):** For continuous age prediction.

**Optimizers:** Use algorithms like Adam to minimize the loss.

**Regularization:** Include techniques like dropout and batch normalization to prevent overfitting.

**5.Evaluation and Testing Module**

Measures the performance of the trained CNN.

Metrics:

**Accuracy:** For gender classification.

**Mean Absolute Error (MAE):** For age prediction.

**Validation and Test Set:** Use unseen data to evaluate the model’s generalization capability.

**Confusion Matrix:** Provides insights into misclassifications.

**6.2 Introduction to Technologies Used**

**1.Convolutional Neural Networks (CNNs)**

CNNs are specialized deep learning architectures designed for image-related tasks. They extract hierarchical features from images, such as edges, textures, and complex patterns, making them ideal for facial feature analysis. Key components include convolutional layers, pooling layers, and fully connected layers for classification or regression.

**2.Image Processing Tools**

Preprocessing facial images ensures high-quality input for the CNN.

**Face Detection:** Tools like OpenCV or Dlib are used to locate and crop faces in images.

**Image Augmentation:** Techniques like rotation, flipping, and scaling improve model generalization by artificially expanding the dataset.

**3.Deep Learning Frameworks**

Frameworks like TensorFlow, Keras, and PyTorch simplify building, training, and deploying CNN models. These tools provide pre-built layers, loss functions, and optimizers, accelerating the development process.

**4.Transfer Learning**

Pre-trained models like VGG16, ResNet, and MobileNet are fine-tuned for specific tasks such as age and gender detection. Transfer learning reduces the need for large datasets and computational resources while maintaining high accuracy.

**5.Optimization Techniques**

To improve performance and prevent overfitting:

**Adam Optimizer:** Used to fine-tune model weights efficiently.

**Regularization:** Techniques like dropout and batch normalization stabilize training and enhance generalization.

**6.3 Implementation**

The implementation follows a systematic process from data collection to model deployment:

**1. Project Setup**

**Objective:** To design a system that detects and classifies a person’s age group and gender from facial images.

**Frameworks and Tools:** Use deep learning frameworks like TensorFlow/Keras or PyTorch for building and training the CNN model. Libraries like OpenCV are used for face detection and preprocessing.

**2. Data Preparation**

**Dataset Collection:**

Datasets like Adience or IMDB-WIKI are widely used, containing labeled facial images with age and gender annotations. The dataset should include a diverse range of facial images for better generalization.

**Data Preprocessing:**

**Face Detection:** Detect and crop faces from images using tools like OpenCV or Dlib to focus only on the region of interest.

**Image Resizing:** Resize the cropped images to a fixed dimension (e.g., 224x224) to ensure compatibility with the CNN model.

**Normalization:** Normalize pixel values to a range (e.g., [0, 1]) to standardize the input data for efficient training.

**Data Augmentation:**

Apply transformations like rotation, flipping, and zooming to increase data diversity and reduce overfitting.

**3. CNN Model Design**

**Architecture Design:**

**Input Layer:** Accepts resized images (e.g., 224x224x3 for RGB).

**Convolutional Layers:** Extract hierarchical features like edges, shapes, and textures.

**Pooling Layers:** Reduce spatial dimensions while retaining essential features.

**Fully Connected Layers:** Combine extracted features to predict age and gender.

**Output Layer:**

**Gender Classification:** A softmax activation function for two categories: male or female.

**Age Prediction:** Either regression for continuous age or classification for age groups (e.g., 0–18, 19–30).

**Transfer Learning:**

Use pre-trained models like VGG16, ResNet, or MobileNet to fine-tune for age and gender detection, saving time and computational resources.

**4. Model Training and Optimization**

**Loss Functions:**

Categorical Cross-Entropy for gender classification or age group classification.

Mean Squared Error (MSE) for continuous age prediction.

**Optimizer:** Use optimization algorithms like Adam for efficient weight updates.

**Regularization:**

**Dropout:** Prevents overfitting by randomly ignoring certain neurons during training.

**Batch Normalization**: Stabilizes and speeds up training by normalizing activations.

**5. Model Evaluation**

**Accuracy:** Evaluate gender classification predictions.

**Mean Absolute Error (MAE):** Assess age prediction performance.

**Confusion Matrix:** Analyze classification performance across all categories.

**6. Deployment**

The trained model can be deployed in real-time systems for applications such as personalized marketing, security, and age-specific services.

**CHAPTER 7**

**RESULT AND**

**PERFORMANCE**

**EVALUATION**

**7.Source Code:**

import cv2

import numpy as np

# Define model paths

face\_pbtxt = "models/opencv\_face\_detector.pbtxt"

face\_pb = "models/opencv\_face\_detector\_uint8.pb"

age\_prototxt = "models/age\_deploy.prototxt"

age\_model = "models/age\_net.caffemodel"

gender\_prototxt = "models/gender\_deploy.prototxt"

gender\_model = "models/gender\_net.caffemodel"

MODEL\_MEAN\_VALUES = [104, 117, 123]

# Load models

face\_net = cv2.dnn.readNet(face\_pb, face\_pbtxt)

age\_net = cv2.dnn.readNet(age\_model, age\_prototxt)

gender\_net = cv2.dnn.readNet(gender\_model, gender\_prototxt)

# Classifications

age\_classifications = ['(0-10)', '(11-20)', '(21-30)', '(31-40)', '(41-50)', '(51-60)', '(61-70)', '(71-80)', '(81-90)', '(91-100)']

gender\_classifications = ['male', 'female']

# Function for face detection and predictions

def detect\_and\_predict(frame, use\_camera=True):

img\_h, img\_w = frame.shape[:2]

blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300), [104, 117, 123], swapRB=False, crop=False)

**for** image\_batch,label\_batch **in** dataset.take(1): print(image\_batch[0].shape)

(512, 512, 3)

# Face detection

face\_net.setInput(blob)

detections = face\_net.forward()

face\_bounds = []

for i in range(detections.shape[2]):

confidence = detections[0, 0, i, 2]

if confidence > 0.7: # Confidence threshold

x1 = int(detections[0, 0, i, 3] \* img\_w)

y1 = int(detections[0, 0, i, 4] \* img\_h)

x2 = int(detections[0, 0, i, 5] \* img\_w)

y2 = int(detections[0, 0, i, 6] \* img\_h)

# Draw rectangle around face

cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

face\_bounds.append([x1, y1, x2, y2])

if not face\_bounds:

print("No faces detected.")

return frame

# Process each detected face

for face\_bound in face\_bounds:

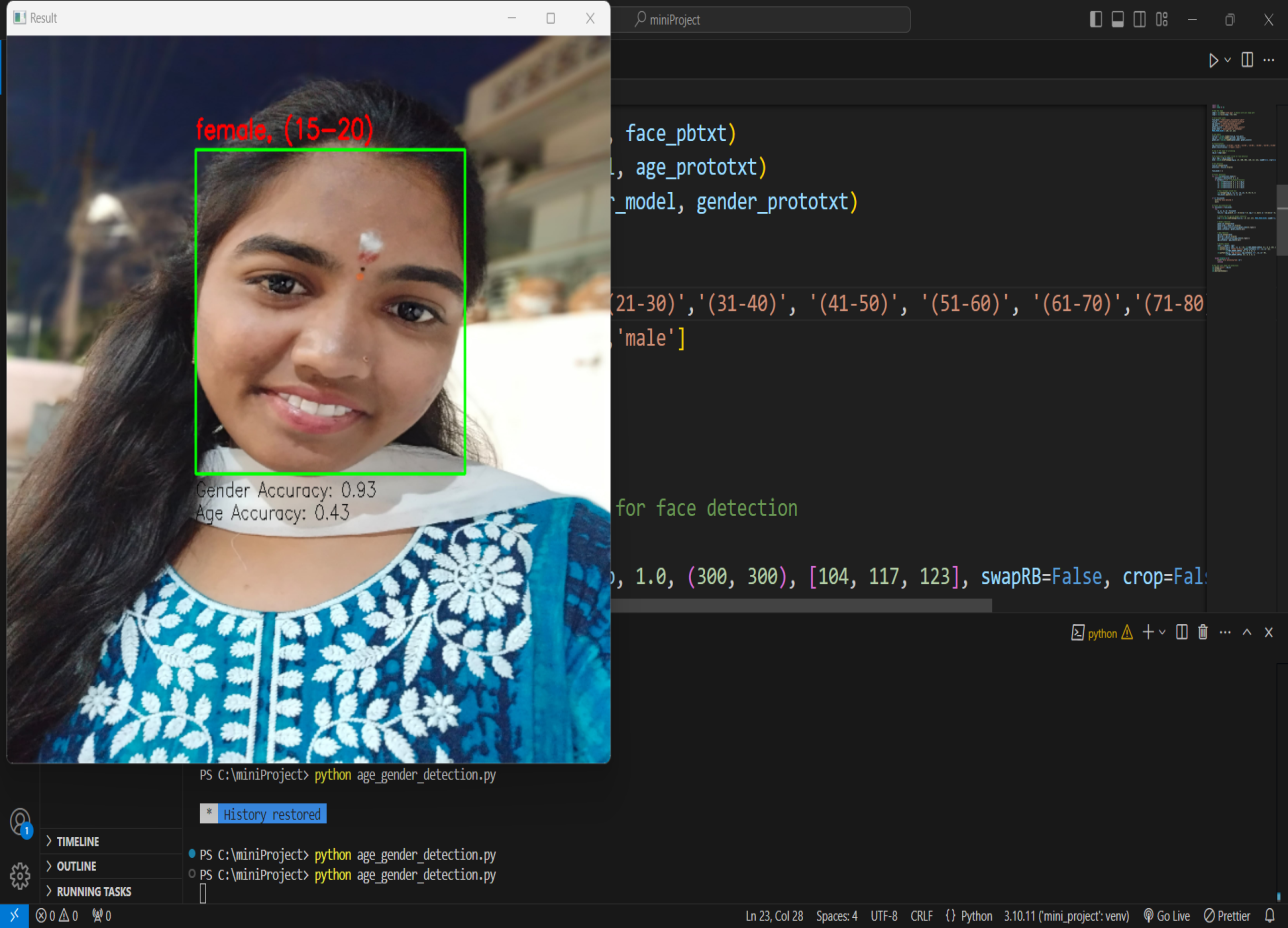
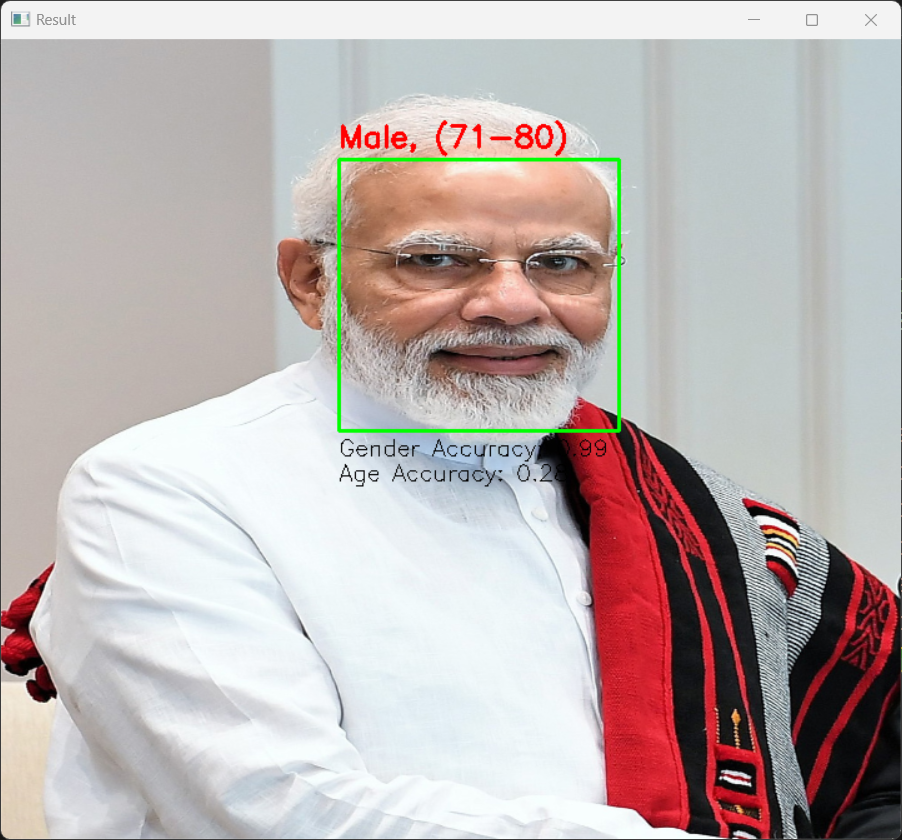
try:

x1, y1, x2, y2 = face\_bound

face\_roi = frame[max(0, y1 - 15):min(y2 + 15, img\_h - 1), max(0, x1 - 15):min(x2 + 15, img\_w - 1)]

# Create blob for age and gender prediction

blob = cv2.dnn.blobFromImage(face\_roi, 1.0, (227, 227), MODEL\_MEAN\_VALUES, swapRB=True, crop=False)



# Gender prediction

gender\_net.setInput(blob)

gender\_preds = gender\_net.forward()

gender = gender\_classifications[gender\_preds[0].argmax()]

gender\_confidence = gender\_preds[0].max()

# Age prediction

age\_net.setInput(blob)

age\_preds = age\_net.forward()

age = age\_classifications[age\_preds[0].argmax()]

age\_confidence = age\_preds[0].max()

# Display results

label = f"{gender}, {age}"

cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.8, (0, 0, 255), 2)

cv2.putText(frame, f"Gender Accuracy: {gender\_confidence:.2f}", (x1, y2 + 20),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.6, (0, 0, 0), 1)

cv2.putText(frame, f"Age Accuracy: {age\_confidence:.2f}", (x1, y2 + 40),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.6, (0, 0, 0), 1)

except Exception as e:

print(f"Error processing face: {e}")

continue

return frame

main()# Main function to run webcam or process an uploaded image

def main():

choice = input("Choose input method: (1) Webcam (2) Image File: ")

if choice == '1': # Webcam

video = cv2.VideoCapture(0)

while cv2.waitKey(1) < 0:

hasFrame, frame = video.read()

if not hasFrame:

cv2.waitKey()

resultImg = detect\_and\_predict(frame, use\_camera=True)

cv2.imshow("Age and Gender Detection", resultImg)

if cv2.waitKey(33) & 0xFF == ord('q'): # Press 'q' to exit

break

video.release()

cv2.destroyAllWindows()

elif choice == '2': # Image file

image\_path = input("Enter image file path: ")

image = cv2.imread(image\_path)

if image is None:

print("Error: Unable to load image.")

return

image = cv2.resize(image, (720, 640))

resultImg = detect\_and\_predict(image, use\_camera=False)

cv2.imshow("Age and Gender Detection", resultImg)

cv2.waitKey(0)

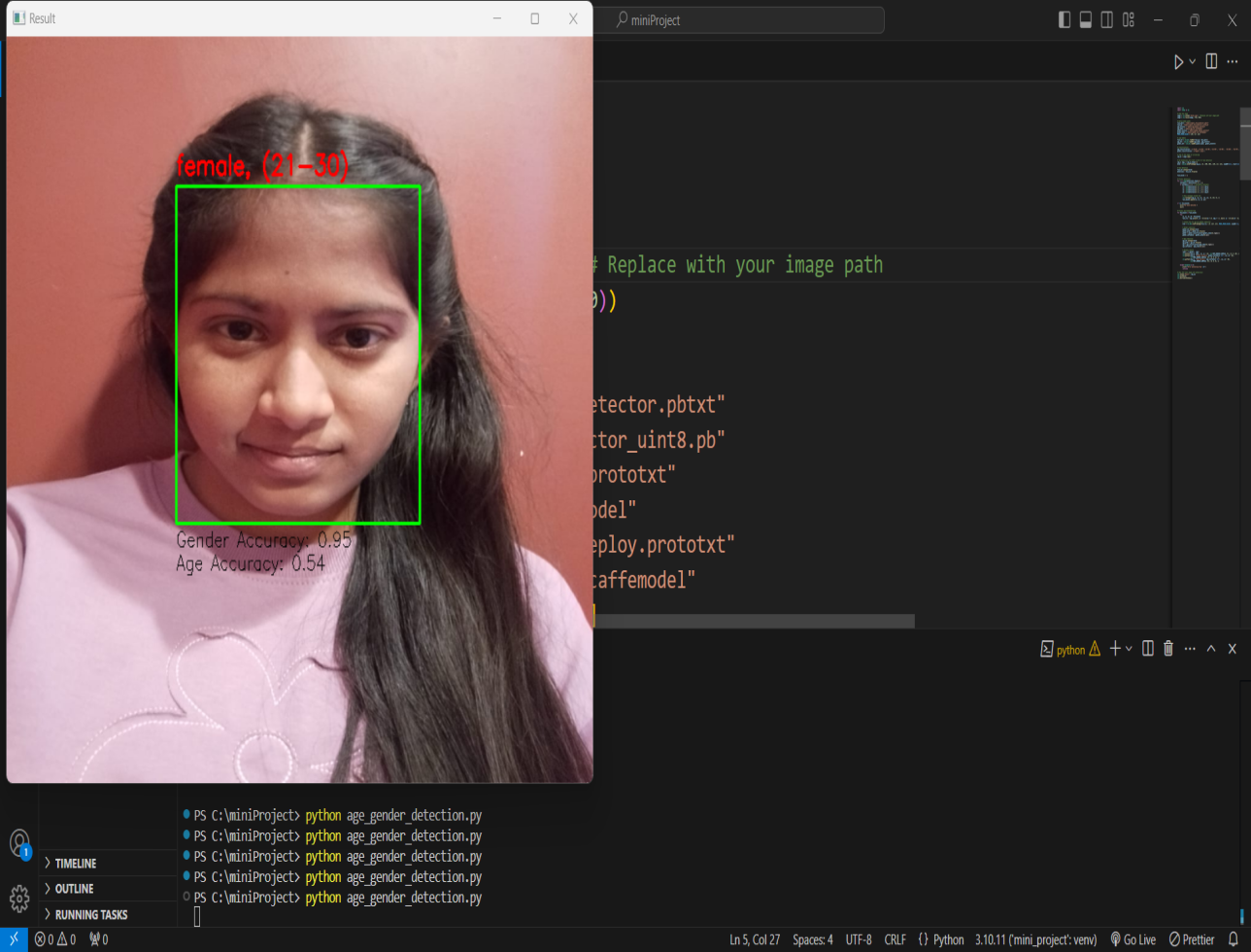
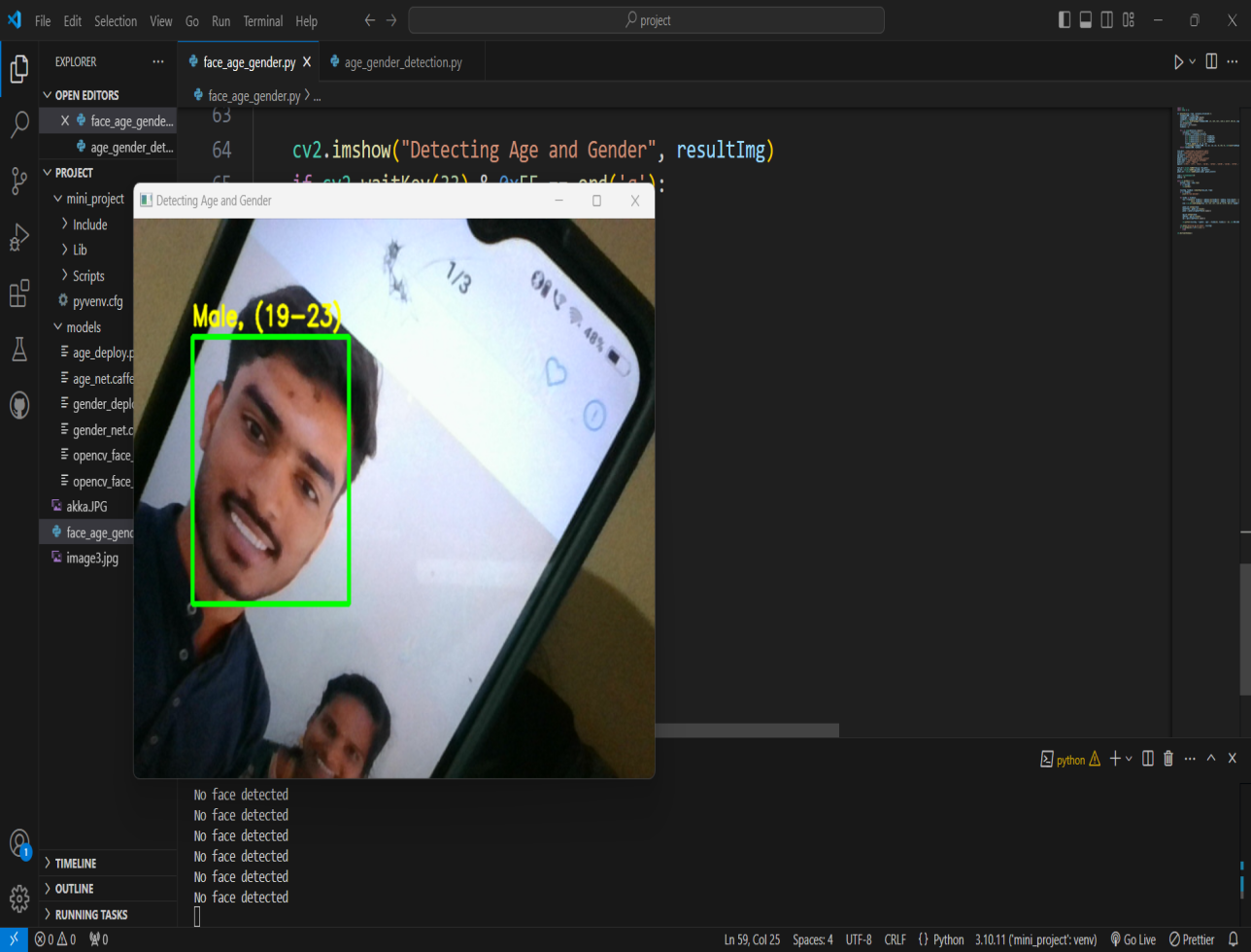
cv2.destroyAllWindows()

else:

print("Invalid choice. Exiting...")

# Run the program

if \_name\_ == "\_main\_":

main()

*#Model Architecture*

**#We use a CNN coupled with software activation in the output layer. We also add the initial layers for resizing,normalization and Data Augumentation.**

input\_shape = (BATCH\_SIZE, IMAGE\_SIZE, IMAGE\_SIZE, CHANNELS)

n\_classes = 4

model = models.Sequential([ resize\_and\_rescale,

layers.Conv2D(32, kernel\_size = (3,3), activation='relu',␣

↪input\_shape=input\_shape),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, kernel\_size = (3,3), activation='relu'), layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, kernel\_size = (3,3), activation='relu'),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D((2, 2)), layers.Flatten(),

layers.Dense(64, activation='relu'), layers.Dense(n\_classes, activation='softmax'),

])

model.build(input\_shape=input\_shape)

model.summary()

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

==========================================================

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | sequential (Sequential) | (5, 512, 512, 3) | 0 | | conv2d (Conv2D) | (5, 510, 510, 32) | 896 | | max\_pooling2d (MaxPooling2D  ) | (5, 255, 255, 32) | 0 | | conv2d\_1 (Conv2D) | (5, 253, 253, 64) | 18496 | | max\_pooling2d\_1 (MaxPooling 2D) | (5, 126, 126, 64) | 0 | | conv2d\_2 (Conv2D) | (5, 124, 124, 64) | 36928 | | max\_pooling2d\_2 (MaxPooling 2D) | (5, 62, 62, 64) | 0 | | conv2d\_3 (Conv2D) | (5, 60, 60, 64) | 36928 | | max\_pooling2d\_3 (MaxPooling 2D) | (5, 30, 30, 64) | 0 | | conv2d\_4 (Conv2D) | (5, 28, 28, 64) | 36928 | | max\_pooling2d\_4 (MaxPooling 2D) | (5, 14, 14, 64) | 0 |   conv2d\_5 (Conv2D) (5, 12, 12, 64) 36298  max\_pooling2d\_5 (MaxPooling (5, 6, 6, 64) 0  2D)  flatten(Flatten) (5,2304) 36298  dense(Dense) (5,64) 147520  dense\_1(Dense) (5,4) 260 | |

=====================================================

Total params: 314,884

Trainable params: 314,884

Non-trainable params: 0

*#Compiling the Model*

*#We use adam Optimizer, SparseCategoricalCrossentropy for losses, accuracy as a*␣

↪*metric*

model.compile( optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=**False**), metrics=['accuracy']

)

history = model.fit( train\_ds, batch\_size=10, validation\_data=val\_ds, verbose=1,

epochs=15,

)

Epoch 1/15

914/914 [==============================] - 239s 235ms/step - loss: 0.9127- accuracy: 0.6274 - val\_loss: 0.6929 - val\_accuracy: 0.7333

Epoch 2/15

914/914 [==============================] - 171s 187ms/step - loss: 0.7000- accuracy: 0.7281 - val\_loss: 0.5990 - val\_accuracy: 0.7614

Epoch 3/15

914/914 [==============================] - 164s 179ms/step - loss: 0.6167- accuracy: 0.7561 - val\_loss: 0.5569 - val\_accuracy: 0.7789

Epoch 4/15

914/914 [==============================] - 169s 185ms/step - loss: 0.5434- accuracy: 0.7916 - val\_loss: 0.5616 - val\_accuracy: 0.7737

Epoch 5/15

914/914 [==============================] - 165s 180ms/step - loss: 0.4740- accuracy: 0.8187 - val\_loss: 0.5083 - val\_accuracy: 0.8158

Epoch 6/15

914/914 [==============================] - 171s 187ms/step - loss: 0.4092- accuracy: 0.8476 - val\_loss: 0.5241 - val\_accuracy: 0.8035

Epoch 7/15

914/914 [==============================] - 169s 184ms/step - loss: 0.3690- accuracy: 0.8588 - val\_loss: 0.3844 - val\_accuracy: 0.8737

Epoch 8/15

914/914 [==============================] - 168s 183ms/step - loss: 0.3246- accuracy: 0.8715 - val\_loss: 0.3304 - val\_accuracy: 0.8789

Epoch 9/15

914/914 [==============================] - 164s 179ms/step - loss: 0.3144- accuracy: 0.8800 - val\_loss: 0.2812 - val\_accuracy: 0.8877

Epoch 10/15

914/914 [==============================] - 170s 186ms/step - loss: 0.2955- accuracy: 0.8899 - val\_loss: 0.2902 - val\_accuracy: 0.8877

Epoch 11/15

914/914 [==============================] - 171s 188ms/step - loss: 0.2711- accuracy: 0.8973 - val\_loss: 0.2069 - val\_accuracy: 0.9281

Epoch 12/15

914/914 [==============================] - 171s 187ms/step - loss: 0.2489- accuracy: 0.9032 - val\_loss: 0.2201 - val\_accuracy: 0.9193

Epoch 13/15

914/914 [==============================] - 171s 187ms/step - loss: 0.2357- accuracy: 0.9109 - val\_loss: 0.2624 - val\_accuracy: 0.8982

Epoch 14/15

914/914 [==============================] - 168s 184ms/step - loss: 0.2363- accuracy: 0.9124 - val\_loss: 0.1800 - val\_accuracy: 0.9474

Epoch 15/15

914/914 [==============================] - 172s 188ms/step - loss: 0.2033- accuracy: 0.9249 - val\_loss: 0.1820 - val\_accuracy: 0.9351

scores = model.evaluate(test\_ds)

115/115 [==============================] - 47s 39ms/step - loss: 0.2138 -

accuracy: 0.9270

Scores

[0.21375764906406403, 0.9269565343856812]

*#Plotting the Accuracy and Loss Curves*

**history**

<keras.callbacks.History at 0x2475453a1d0>

History.params

{'verbose': 1, 'epochs': 15, 'steps': 914}

History.history.keys()

dict\_keys(['loss', 'accuracy', 'val\_loss', 'val\_accuracy'])

type(history.history['loss'])

list

len(history.history['loss'])

15

history.history['loss'][:5]

[0.9126647710800171,

0.6999651789665222,

0.6167153716087341,

0.5433878898620605,

0.4739765226840973]

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss'] val\_loss = history.history['val\_loss']

plt.figure(figsize=(12,6)) plt.subplot(1, 2, 1)

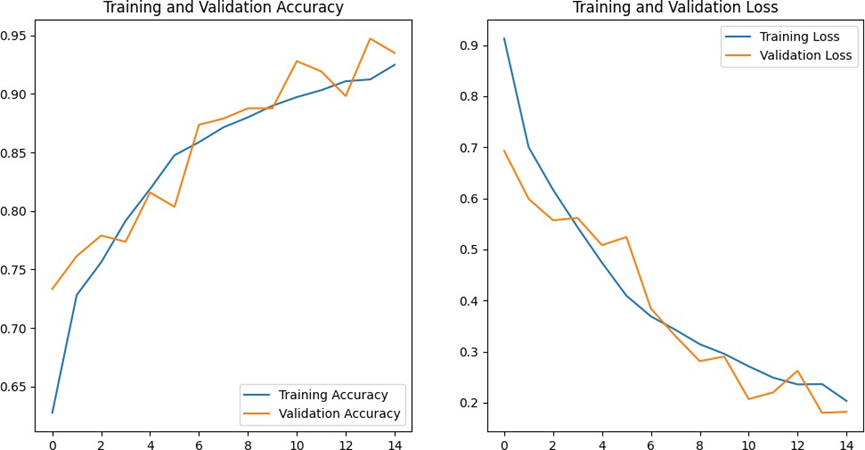
plt.plot(range(EPOCHS), acc, label='Training Accuracy') plt.plot(range(EPOCHS), val\_acc, label='Validation Accuracy') plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)

plt.plot(range(EPOCHS), loss, label='Training Loss') plt.plot(range(EPOCHS), val\_loss, label='Validation Loss') plt.legend(loc='upper right')

plt.title('Training and Validation Loss') plt.show()



*#Run prediction on a sample image*

**import numpy as np**

**for** images\_batch, labels\_batch **in** test\_ds.take(1):

first\_image = images\_batch[3].numpy().astype('uint8') first\_label = labels\_batch[3].numpy()

print("first image to predict") plt.imshow(first\_image)

print("actual label:",class\_names[first\_label])

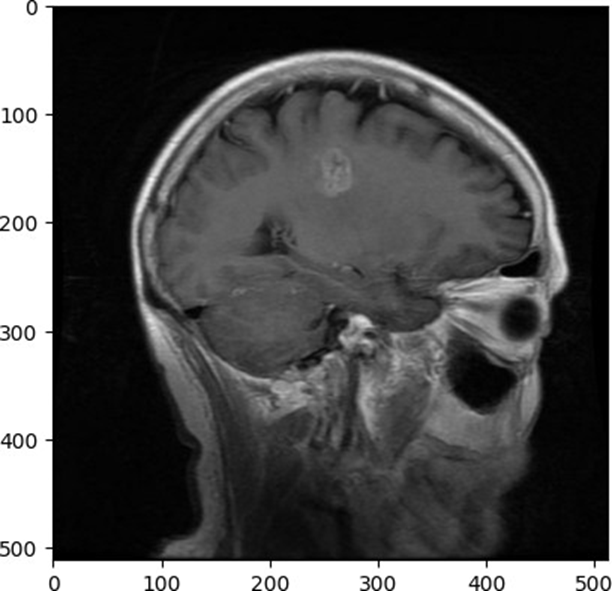
batch\_prediction = model.predict(images\_batch)

print("predicted label:",class\_names[np.argmax(batch\_prediction[0])])

first image to predict

actual label: glioma

predicted label: glioma



*#Write a function for inference*

**def** predict(model, img):

img\_array = tf.keras.preprocessing.image.img\_to\_array(images[i].numpy()) img\_array = tf.expand\_dims(img\_array, 0)

predictions = model.predict(img\_array)

predicted\_class = class\_names[np.argmax(predictions[0])] confidence = round(100 \* (np.max(predictions[0])), 2) **return** predicted\_class, confidence

*#Now run inference on few sample images*

plt.figure(figsize=(15, 15))

**for** images, labels **in** test\_ds.take(1):

**for** i **in** range(5):

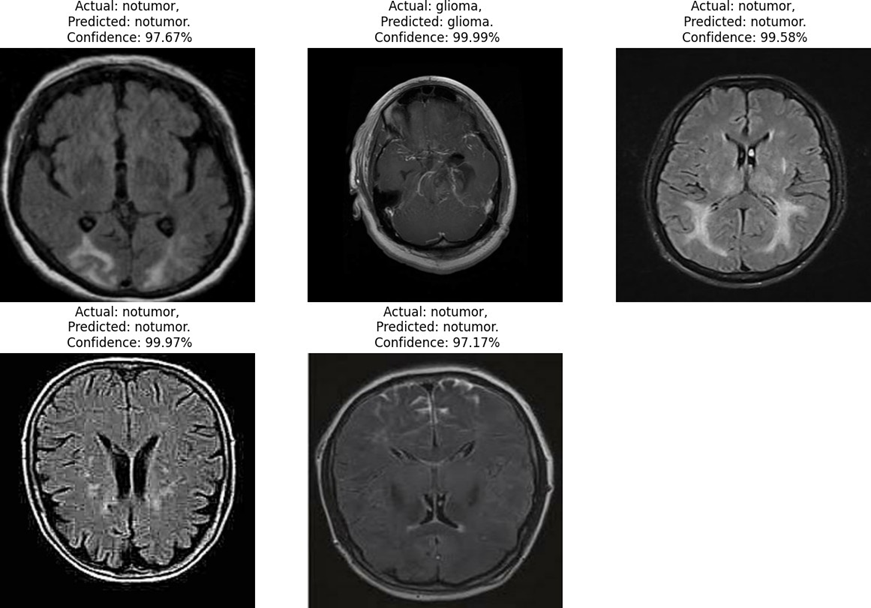
ax = plt.subplot(3, 3, i + 1) plt.imshow(images[i].numpy().astype("uint8"))

predicted\_class, confidence = predict(model, images[i].numpy()) actual\_class = class\_names[labels[i]]

plt.title(f"Actual: **{**actual\_class**}**,**\n** Predicted: **{**predicted\_class**}**.**\n**␣

↪Confidence: **{**confidence**}**%")

plt.axis("off")



*#Saving the Model*

model.save("./models/barin\_tumor.h5")

*# Save the history variable*

**import pickle**

**with** open('brain\_tumor.pkl', 'wb') **as** f: pickle.dump(history.history, f)

**with** open('brain\_tumor.pkl', 'rb') **as** f: history = pickle.load(f)

*# Use the history variable as needed*

print(history)

{'loss': [0.9126647710800171, 0.6999651789665222, 0.6167153716087341,

0.5433878898620605, 0.4739765226840973, 0.4092278778553009, 0.368973970413208,

0.3425751328468323, 0.3143966794013977,0.29548484086990356,0.2710789740085602,

0.2488735020160675, 0.23565390706062317, 0.23630201816558838,

0.20327456295490265], 'accuracy': [0.6274080276489258, 0.7281085848808289,

0.7561296224594116, 0.7915936708450317, 0.8187390565872192, 0.8476357460021973,

0.8588003516197205, 0.871497392654419, 0.8800350427627563, 0.8898861408233643,

0.8973292708396912, 0.903239905834198, 0.9109019041061401, 0.9124343395233154,

0.9249124526977539], 'val\_loss': [0.692949652671814, 0.5989788174629211,

0.5568957924842834, 0.5616412162780762, 0.508283793926239, 0.5240986943244934,

0.38437792658805847, 0.33041295409202576, 0.2811759114265442,

0.29024627804756165, 0.20691806077957153, 0.2200576364994049,

0.26239684224128723, 0.1800486147403717, 0.18201439082622528], 'val\_accuracy':

[0.7333333492279053, 0.761403501033783, 0.7789473533630371, 0.7736842036247253,

0.8157894611358643, 0.8035087585449219, 0.8736842274665833, 0.878947377204895,

0.8877192735671997, 0.8877192735671997, 0.9280701875686646, 0.9192982316017151,

0.898245632648468, 0.9473684430122375, 0.9350877404212952]}

# **import pandas as pd**

# **import seaborn as sns**

acc = history['accuracy']

val\_acc = history['val\_accuracy']

epochs= range(len(acc))

*# Extracting data from history dictionary*

accuracy = history['accuracy']

loss = history['loss']

val\_accuracy = history['val\_accuracy']

val\_loss = history['val\_loss']

epochs = range(1, len(accuracy) + 1)

*# Plotting accuracy*

plt.plot(epochs, accuracy, 'bo', label='Training accuracy')

plt.plot(epochs, val\_accuracy, 'b', label='Validation accuracy')

plt.title('Training and validation accuracy') plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend() plt.show()

*# Plotting loss*

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')

plt.plot(epochs, val\_loss, 'b', label='Validation loss')

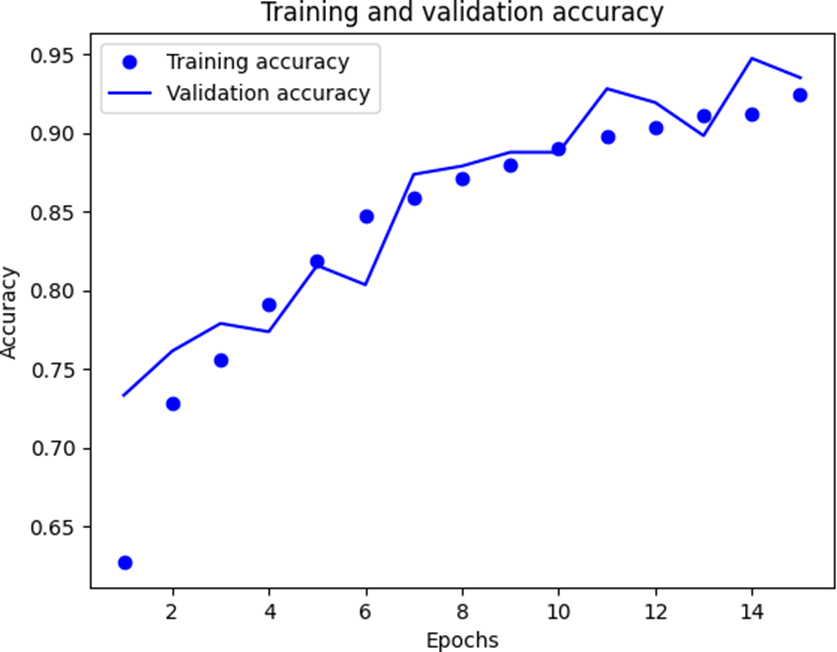
plt.title('Training and validation loss')

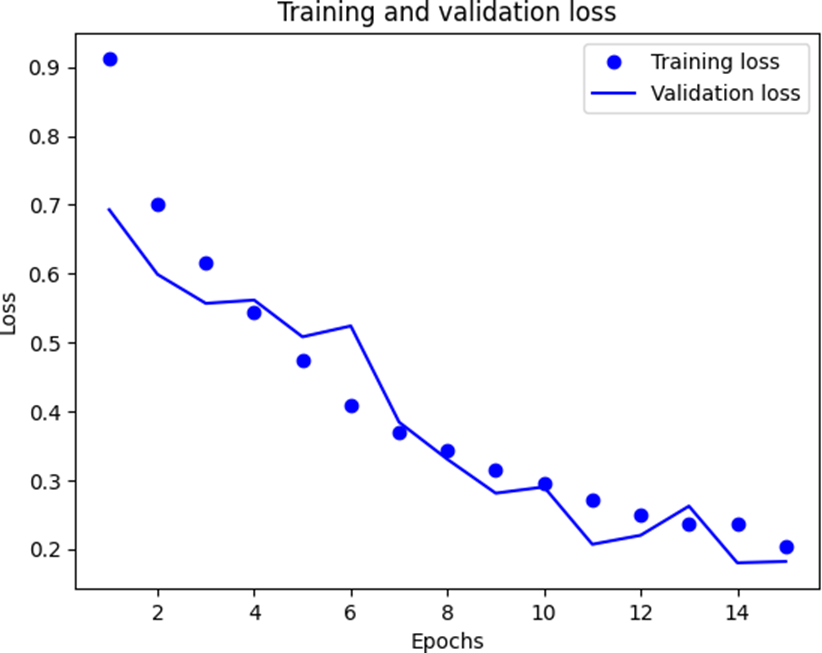
plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()





*# Converting history to a dataframe*

history\_df = pd.DataFrame(history)

*# Heatmap for accuracy*

plt.figure(figsize=(10,6)) sns.heatmap(history\_df[['accuracy','val\_accuracy']], annot = **True**, fmt='.4f',␣

↪cmap ='coolwarm')

plt.title('Accuracy Heatmap') plt.xlabel('Metrices') plt.ylabel('Epochs') print(plt.show())

*# Heatmap for Loss*

plt.figure(figsize=(10,6)) sns.heatmap(history\_df[['loss','val\_loss']], annot = **True**, fmt='.4f',␣

↪cmap='coolwarm')

plt.title('Loss Heatmap') plt.xlabel('Metrices')

plt.ylabel('Epochs')

print(plt.show())

