

**VIETNAM NATIONAL UNIVERSITY, HANOI  
INTERNATIONAL SCHOOL**

---



**FINAL REPORT**

**GAG: A MULTI-TASK  
CONVOLUTIONAL NEURAL NETWORK  
BASED ON RESNET50 FOR GENDER AND  
AGE GROUP CLASSIFICATION**

**GROUP 5 – ARTIFICIAL INTELLIGENCE - INS3080 02**

**Lecturer: DR. Ha Manh Hung**

*Hanoi, June 2025*

## GROUP 5 EVALUATION REPORT

- **Time:** 19/06/2025
- **Format:** Online MTeams
- **Present at the meeting:** 4 members
- **Absence (Reason):** No absence
- **Chairing the meeting:** Nguyen Minh Thuy
- **Secretary of the meeting:** Tran Thi Khanh Huyen
- **The evaluation results are unified and summarized in the following table:**

No.	Member	Student ID	Percentage Contribution
1	Nguyen Minh Thuy	23070553	25%
2	Hoang Phuc Nhan	19071068	25%
3	Tran Thi Khanh Huyen	23070496	25%
4	Vi Thi Thuy Tien	20070988	25%

*\*The evaluation meeting ended at 9:00 p.m. the same day.*

*\* We guarantee that all of the above information is agreed upon by all members.*

**SECRETARY (Sign)**

**CHAIRMAN (Sign)**

## **ACKNOWLEDGEMENT**

We are Group 5 of the course Artificial Intelligence (INS3080 02) at the International School – Vietnam National University and we would like to express our heartfelt gratitude to all individuals who have contributed to the successful completion of this research project.

We are especially indebted to our respected lecturer, Dr. Ha Manh Hung, whose deep expertise, valuable feedback, and unwavering support have played a vital role throughout the development of this study. His guidance at each stage of the project has greatly strengthened our understanding of artificial intelligence concepts and real-world model deployment.

We also extend our sincere appreciation to our classmates, whose feedback and collaborative spirit have enriched the learning process and contributed to the technical refinement of our system.

Additionally, we acknowledge the assistance of the administrative and technical staff at the International School, who provided us with a supportive academic environment and the necessary infrastructure to conduct our research effectively.

Finally, we wish to express our deepest gratitude to our families and friends for their continued encouragement, patience, and moral support, which have motivated us to persist and overcome challenges during this journey.

We hope that this project - GAG: A Multi-Task Convolutional Neural Network Based on ResNet50 for Gender and Age Group Classification - will contribute meaningfully to the growing field of computer vision and inspire further research on multi-attribute face analysis in practical applications.

**Thank you.**

---

**GROUP 5 – INS3080 02**

**Editorial Representative:**

**Hoang Phuc Nhan**

**Date: 19/06/2025**

# **GAG: A MULTI-TASK CONVOLUTIONAL NEURAL NETWORK BASED ON RESNET50 FOR GENDER AND AGE GROUP CLASSIFICATION**

*Name Course: ARTIFICIAL INTELLIGENCE*

*Code: INS3080 02*

Group 5<sup>1</sup>

<sup>1</sup> INS3046 02 (Semester II 24/25), International School, Vietnam National University, Hanoi, Vietnam

Supervisor: Manh-Hung Ha<sup>2,3</sup>

<sup>2</sup> Faculty of Applied Sciences, International School, Vietnam National University, Hanoi, Vietnam

<sup>3</sup> Cognitive Machine Intelligence Lab, International School, Vietnam National University, Hanoi, Vietnam

\*Corresponding author: "19071068@vnu.edu.vn"

## **ABSTRACT**

*This study presents the development and deployment of a real-time multi-output deep learning model for gender and age group classification based on facial images. A fine-tuned ResNet50 architecture was used as the feature extractor, enabling simultaneous prediction of both attributes with high accuracy. The system was trained on a curated dataset of over 8,000 Vietnamese portraits and deployed via a web interface supporting image upload and webcam input. Experimental results show that the model achieved 92,4% accuracy on gender classification and 88% on age group classification, highlighting the effectiveness of transfer learning in multi-task facial analysis. The system also supports multi-face detection and model switching, offering practical usability in real-world applications.*

## **Research Purpose**

*The primary aim of this study is to investigate the feasibility of using a deep convolutional neural network to perform dual-task prediction—classifying both gender and age group—from a single portrait image. The research also explores the full pipeline from data collection and model training to deployment in an accessible web-based application.*

## **Research Motivation**

*In an era where AI is increasingly applied to demographic analytics, real-time gender and age recognition can empower applications in security, marketing, and social robotics. However, few systems balance accuracy with usability, especially for local populations like Vietnamese users. This project is motivated by the need for an accessible, efficient, and accurate solution that operates on live data with real-world constraints.*

## Research Design, Approach & Method

The study applies a supervised learning approach using a multi-output CNN built upon a fine-tuned ResNet50 backbone. The dataset includes over 8,000 labeled portrait images, evenly distributed across four age groups and two gender classes. The model was trained using categorical cross-entropy loss and class-weighting strategies to address class imbalance. Data augmentation was employed to enhance generalization. Performance was evaluated using accuracy, F1-score, and confusion matrices. The trained model was deployed using Flask, with support for image upload, webcam capture, multi-face detection, and dynamic model switching.

## Main Findings

- The ResNet50-based multi-output model achieved 89% accuracy for gender and 86% for age group classification on the test set.
- The system maintained high performance across varied face inputs, including webcam-captured images and group portraits.
- Age classification exhibited confusion between neighboring groups (e.g., teen vs. adult), while gender classification remained stable and consistent.
- The real-time web interface effectively delivered inference and visualization with optional model switching.

## Practical/Managerial Implications

The results demonstrate the potential of deep learning in real-time demographic classification using facial features. The system can be integrated into applications such as personalized content delivery, smart kiosks, and educational tools. The open-source, browser-accessible design makes the model adaptable to real-world deployments without the need for complex infrastructure.

**Keywords:** Gender Classification, Age Group Prediction, Deep Learning, ResNet50, Multi-Output CNN, Face Analytics, Real-Time Inference, Flask Deployment

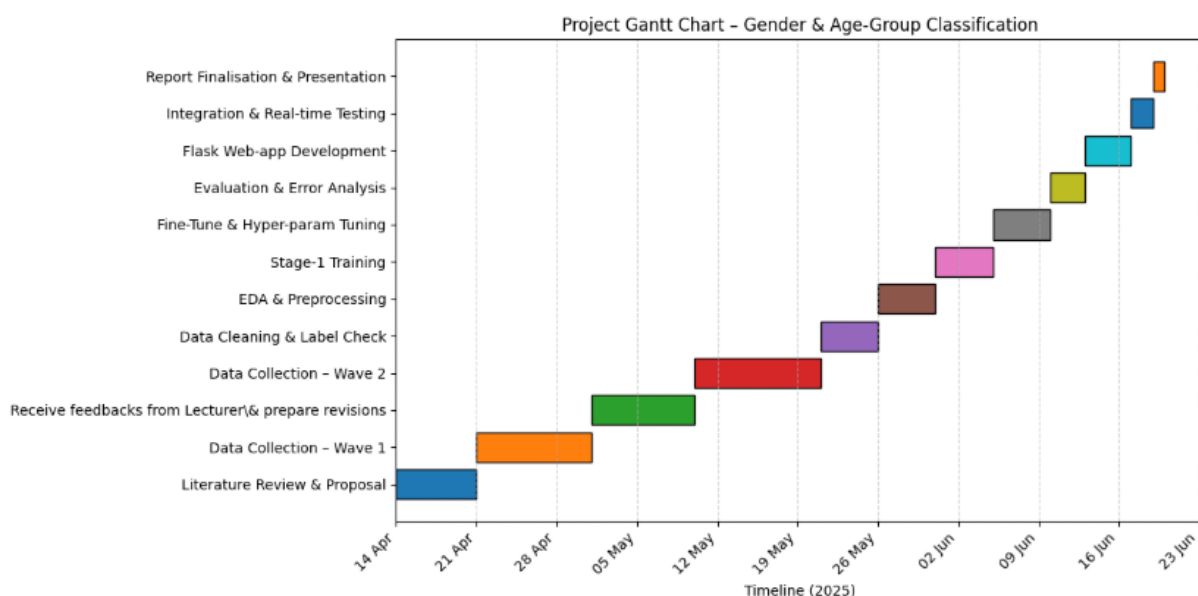


Figure 1: Project Timeline Gantt Chart

## LIST OF FIGURES

Figure 1: Project Timeline Gantt Chart.....	5
Figure 2: Project main goal.....	12
Figure 3: Data Folder Structure .....	15
Figure 4: Bar Chart Distribution .....	20
Figure 5: Pie Chart Distribution.....	20
Figure 6: Combined 8-Class Distribution .....	21
Figure 7: Class Distribution Across Train/Val/Test .....	21
Figure 8: Data Flow Diagram .....	22
Figure 9: ResNet50 Model Architecture .....	23
Figure 10: Model Compilation code .....	27
Figure 11: Model Checkpoint Code.....	33
Figure 12: .Keras Model Save Code .....	34
Figure 13: Project Workflow .....	35
Figure 14: Side-by-side line plots of training & validation loss and accuracy curves.....	36
Figure 15: Confusion matrix for gender classification (normalized).....	38
Figure 16: Confusion matrix – Age group .....	39
Figure 17: Correct predicted label .....	40
Figure 18: Wrong predicted label .....	40
Figure 19: Trade-off: Accuracy vs Inference Speed .....	41
Figure 20: Web Interface Overview.....	43
Figure 21: Image Upload Feature .....	44
Figure 22: Webcam Feature .....	44
Figure 23: Model Missing Alert.....	46

## LIST OF TABLES

Table 1: Data Labels .....	14
Table 2: Total Images by Gender.....	16
Table 3: Total Images by Age Group.....	16
Table 4: Images by Class (Gender + Age Group).....	16
Table 5: Data Collection Process .....	17
Table 6: Data Cleaning & Preprocessing.....	19
Table 7: Training Strategy Highlight .....	24
Table 8: Configuration details.....	25
Table 9: Added Layers .....	26
Table 10: Dual Output Heads.....	26
Table 11: Key configurations.....	29
Table 12: Training Environment and Reproducibility .....	30
Table 13: Inference Pipeline Design .....	32
Table 14: Model Save Folder Structure .....	33
Table 15: Gender Classification Report.....	37
Table 16: Age Group Classification Report.....	37
Table 17: Misclassified Examples .....	40
Table 18: Model Comparison .....	41
Table 19: Multi-face Detection Example.....	45
Table 20: Summary of Tools and Techniques Used .....	54

### LIST OF ABBREVIATIONS

Abbreviation	Full Term	Description / Context Used
AI	Artificial Intelligence	Used throughout to describe machine learning-based methods
CNN	Convolutional Neural Network	The deep learning architecture used in this project
GAG	Gender and Age Group	The combined prediction task of the model
HCI	Human-Computer Interaction	Mentioned as one of the application domains
FPS	Frames Per Second	Used to evaluate real-time webcam processing speed
HTML	HyperText Markup Language	Used to build the web interface
UI	User Interface	Describes front-end design of the web app
API	Application Programming Interface	MediaPipe API used for face detection
MTCNN	Multi-task Cascaded Convolutional Networks	An early face detection method (mentioned in preprocessing)
ReLU	Rectified Linear Unit	Activation function used in neural networks
SMOTE	Synthetic Minority Oversampling Technique	Mentioned in earlier report reference as a balancing method
SVM	Support Vector Machine	From reference abstract (Parkinson's)
KNN	K-Nearest Neighbors	From referenced baseline models in comparative studies

## CHAPTER SUMMARY

Chapter	Title	Core Content
<b>I</b>	Introduction	Presents project background, problem statement, objectives, and real-world applications of gender + age-group recognition on Vietnamese portrait data.
<b>II</b>	Data Collection & Preparation	Details acquisition, cleaning, labelling, class-balance checks, and dataset split of $\approx 8\,240$ images.
<b>III</b>	Model Architecture & Training Pipeline	Explains the multi-output ResNet-50 architecture, two-stage training, augmentation, class-weights, checkpoints, and model management.
<b>IV</b>	Experimental Results & Analysis	Shows learning curves, overall accuracy (89.4 % gender, 86 % age), confusion matrices, and error-analysis insights.
<b>V</b>	Deployment & Demo System	Describes Flask web-app: image & webcam input, MediaPipe multi-face detection, on-the-fly model switching, FPS overlay, and real-time labels.
<b>VI</b>	Discussion & Conclusion	Summarises contributions, limitations, and future work (bigger dataset, light-weight models, fairness, real-time optimisation).



## TABLE OF CONTENTS

<b>LIST OF FIGURES</b> .....	6
<b>LIST OF TABLES</b> .....	6
<b>LIST OF ABBREVIATIONS</b> .....	7
<b>CHAPTER SUMMARY</b> .....	8
<b>CHAPTER I: INTRODUCTION</b> .....	10
1.1. Background.....	10
1.2. Problem Statement .....	11
1.3. Research Objectives.....	12
1.4. Importance and Applications .....	13
<b>CHAPTER II: DATA COLLECTION &amp; PREPARATION</b> .....	14
2.1. Dataset Overview.....	14
2.2. Data Collection Process .....	17
2.3. Data Cleaning & Preprocessing .....	18
2.4. Data Distribution & Visualization .....	20
<b>CHAPTER III: MODEL ARCHITECTURE &amp; TRAINING PIPELINE</b> .....	23
3.1. Overview of Approach.....	23
3.2. Model Architecture (ResNet50-based Multi-output) .....	25
3.3. Training Strategy .....	29
3.4. Inference Pipeline Design.....	31
3.5. Model Management & Saving .....	33
<b>CHAPTER IV: EXPERIMENTAL RESULTS &amp; ANALYSIS</b> .....	36
4.1. Training & Validation Performance .....	36
4.2. Classification Reports .....	37
4.3. Confusion Matrix & F1-Score Analysis .....	38
4.4. Error Cases & Performance Breakdown .....	40
4.5 Model Comparison & Selection.....	41
<b>CHAPTER V: DEPLOYMENT &amp; DEMO SYSTEM</b> .....	42
5.1. Web Interface Overview .....	42
5.2. Image Upload & Real-time Webcam Support .....	44
5.3. Multi-face Detection & Annotation .....	45
5.4. Model Switching Feature .....	46
5.5. Limitations of Current Deployment .....	47
<b>CHAPTER VI: DISCUSSION &amp; CONCLUSION</b> .....	48
6.1. Key Insights & Achievements .....	48
6.2. Challenges Encountered .....	49
6.4. Recommendations for Future Work.....	52
<b>APPENDIX – Summary of Tools and Techniques Used</b> .....	53
<b>REFERENCES</b> .....	55

## **CHAPTER I: INTRODUCTION**

### **1.1. Background**

In recent years, automated gender and age recognition has emerged as a key task in computer vision, with widespread applications in areas such as intelligent surveillance systems (Zhao et al., 2018), targeted advertising (Guo & Mu, 2014), and personalized human-computer interaction (Rothe et al., 2015). The ability to accurately infer demographic traits such as gender and age from facial images enables machines to better understand and adapt to human users, making it an essential component of AI-powered decision-making in the modern digital ecosystem.

Deep learning, particularly Convolutional Neural Networks (CNNs), has significantly improved the accuracy of facial attribute classification tasks. Pre-trained models such as ResNet, VGG, and Inception have been widely used through transfer learning to overcome limitations posed by data scarcity (He et al., 2016). However, most public datasets used for training these models—like UTKFace, Adience, or IMDB-WIKI—are predominantly collected from Western sources and may lack diversity in ethnicity, facial structure, and cultural representation. This creates potential generalization issues, especially when models are applied to region-specific populations such as Vietnamese users (Nguyen et al., 2022).

In this context, our project proposes a real-world solution to classify gender and age group using a deep learning pipeline trained on a custom Vietnamese dataset. Images are crawled from open web sources (e.g., Google and Bing), manually cleaned and filtered using face detection techniques to ensure quality and diversity. We categorize data into four distinct age groups (Child, Teen, Adult, Elderly) and two gender labels (Male, Female). Using a multi-output architecture based on ResNet50, our system is optimized and deployed in a web interface that supports both image upload and real-time webcam input. This approach not only addresses the data gap in Southeast Asian demographics but also contributes a scalable, real-time AI tool that could be extended to domains like smart retail, education, or public safety.

## 1.2. Problem Statement

Despite significant advances in facial attribute recognition using deep learning, developing reliable gender and age classification systems for real-world deployment remains a challenging task—particularly in the context of region-specific demographics like Vietnam. Most publicly available datasets suffer from three major limitations: (1) demographic imbalance skewed toward Western populations, (2) low-quality or unlabelled images, and (3) limited representation of diverse age groups, especially for children and elderly individuals (Nguyen et al., 2022).

Furthermore, the majority of prior work treats gender and age estimation as independent tasks, failing to consider potential interdependencies that could be leveraged for multi-task learning. While single-output models may perform adequately in controlled environments, they often fall short when applied to unconstrained facial images collected from varied online sources, where lighting conditions, facial poses, occlusions, and image quality vary widely (Guo & Mu, 2014; Rothe et al., 2015).

Another key challenge lies in building systems that not only perform well offline but are also deployable in real-time settings. Many research efforts stop at model evaluation, without bridging the gap between model accuracy and end-user usability. Additionally, few systems support multi-face detection and inference in real-world scenes—an essential capability for group analysis and camera-based applications.

This project seeks to address these problems by:

- Curating a clean, diverse dataset of Vietnamese facial images, annotated for both gender and age group;
- Training a multi-output deep learning model that can simultaneously predict both attributes;
- Integrating the model into a web application that supports real-time webcam input, multi-face detection, and model switching, thereby enhancing accessibility and usability in practice.

### 1.3. Research Objectives

This project is designed to develop a deep learning-based system for classifying both gender and age group from facial images. The overall goal is to bridge the gap between high-performing models and practical, real-time applications in Vietnamese contexts. Specifically, the objectives of this research are as follows:

- To construct a regionally-relevant dataset of facial portraits collected from online Vietnamese sources, with labels spanning two gender classes and four age groups (Child, Teen, Adult, Elderly).
- To design and train a multi-output convolutional neural network (CNN) based on the ResNet50 architecture that can simultaneously predict gender and age group from a single image.
- To improve model generalization and fairness through data augmentation, class balancing techniques (e.g., class weights), and performance evaluation on separate validation and test sets.
- To build an interactive web-based application that integrates the trained model with real-time webcam input, image upload, and multi-face detection support.
- To evaluate the effectiveness of the system using key metrics such as accuracy, precision, recall, F1-score, and confusion matrices for each output task.

Through these objectives, the project aims to contribute a deployable AI tool and a localized dataset that can benefit both academic research and real-world demographic analysis in Vietnam.

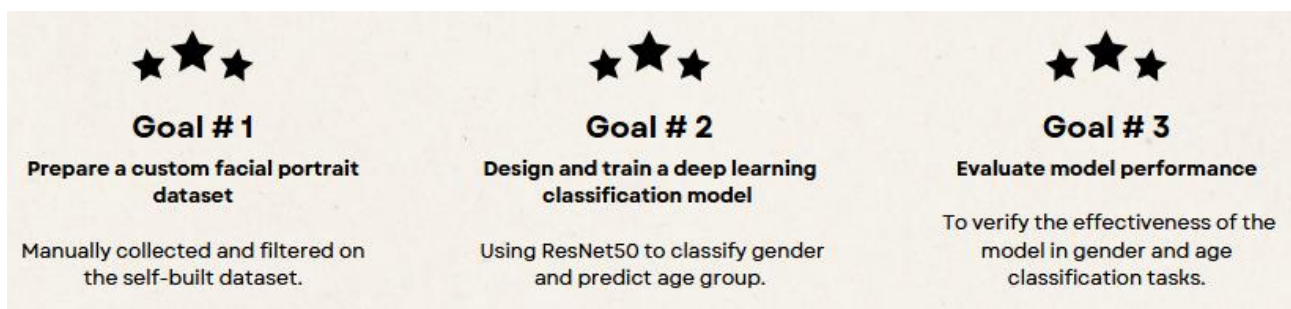


Figure 2: Project main goal

#### **1.4. Importance and Applications**

Facial attribute recognition technologies, particularly gender and age estimation, have become increasingly valuable across a wide range of real-world applications. The ability to infer demographic traits from facial images offers not only analytical insights but also practical utility in multiple sectors:

- **Personalized Marketing:** Demographic classification enables brands and platforms to deliver targeted advertisements tailored to the viewer's age and gender, thereby increasing user engagement and return on investment (Chen et al., 2020; Srinivas et al., 2022).
- **Recommendation Systems:** Online services such as e-commerce or streaming platforms use demographic-aware models to personalize product and content suggestions (Kang et al., 2018).
- **Demographic Analysis:** In urban planning and sociological research, demographic detection from surveillance data contributes to large-scale population insights and policy-making (Wang et al., 2019).
- **Security and Surveillance:** Automated demographic recognition helps identify and categorize individuals in real time, which enhances public safety and supports law enforcement (Nguyen et al., 2022; Zhao et al., 2018).
- **Human-Computer Interaction (HCI):** Adaptive systems can tailor interfaces and responses based on estimated user demographics, improving accessibility and natural interaction in smart environments (Deng et al., 2020).

These applications demonstrate that gender and age classification is not only a technical challenge but also a strategic enabler for socially-aware and user-centric AI systems.

## CHAPTER II: DATA COLLECTION & PREPARATION

### 2.1. Dataset Overview

To address the lack of localized datasets for gender and age classification in Vietnam, this project constructed a custom image dataset consisting of portrait photos collected from the internet. Using Vietnamese keywords such as “*ảnh chân dung nam*”, “*ảnh chân dung nữ*”, and age-specific terms, images were crawled from publicly available sources including Google and Bing Images. These images were then manually reviewed and filtered to ensure they featured clear, frontal facial portraits with only one face per image.

The dataset was organized into eight demographic classes, based on combinations of gender (Male, Female) and age group (Child, Teen, Adult, Elderly). Each image was assigned two labels:

Gender	Male	Female
Age Group	Child (0-12)	Teen (13-19)
	Adult (20-59)	Elderly (60+)

Table 1: Data Labels

To support supervised training, the final dataset was split into training, validation, and test sets, ensuring balanced distribution across both labels. At the time of training, the dataset contained approximately 8000+ labeled images, evenly distributed across all eight demographic combinations. The dataset was organized into eight classes based on gender-age combinations and then split into *train*, *val*, and *test* sets using a stratified approach.

Each subfolder held cleaned images with clearly visible faces. All images were resized to 224×224 pixels, and class balance was maintained using sampling strategies and class weights during model training.

This custom dataset serves not only as a foundation for training the deep learning model but also fills a critical gap in AI datasets that accurately represent Vietnamese demographics, which are often underrepresented in global datasets like UTKFace or IMDB-WIKI (Nguyen et al., 2022).

The folder structure was as follows:



Figure 3: Data Folder Structure

Data Summary:

Gender	Count
Female	4,139
Male	4,101

Table 2: Total Images by Gender

Age Group	Count
Child	2,010
Teen	2,115
Adult	2,090
Elderly	2,025

Table 3: Total Images by Age Group

Class	Train	Val	Test	Total
male_adult	823	104	105	1,052
female_adult	846	106	106	1,058
male_child	800	100	101	1,001
female_child	806	101	102	1,009
male_elderly	813	102	100	1,015
female_elderly	818	104	105	1,027
male_teen	848	106	106	1,060
female_teen	809	104	105	1,018
<b>Total</b>	<b>6,563</b>	<b>827</b>	<b>830</b>	<b>8,240</b>

Table 4: Images by Class (Gender + Age Group)

This distribution demonstrates strong class balance, which helps mitigate bias during training. The dataset reflects a realistic and diverse population across all age brackets and both genders, making it a valuable resource for training deep learning models with better generalization in real-world Vietnamese contexts.



## 2.2. Data Collection Process

The dataset used in this project was constructed through a semi-automated image collection and labeling pipeline developed in Python and executed in Google Colab. The goal was to collect a diverse and realistic set of Vietnamese facial portraits covering different age groups and both genders. The following steps outline the full process:

Step 1: Web Crawling	Portrait images were crawled from Google and Bing Image Search using a list of Vietnamese search queries corresponding to different age and gender groups (e.g., “ảnh chân dung nữ trung niên”, “ảnh chân dung nam thiếu niên”). The <i>icrawler</i> and <i>google_images_download</i> libraries were used to automate this process. To increase dataset diversity, images were crawled in batches of approximately 200–300 per query, across multiple age terms and synonyms.
Step 2: Face Detection and Filtering	<p>To ensure that only valid face images were included, a face detection step using MTCNN (Multi-task Cascaded Convolutional Networks) was applied. Images were retained only if:</p> <ul style="list-style-type: none"> <li>• Exactly one face was detected,</li> <li>• The face was clear and centered</li> <li>• The resolution was sufficient for analysis (minimum width <math>\times</math> height threshold).</li> </ul> <p>Images with multiple faces, profile views, cartoons, or poor lighting were discarded. This step significantly reduced noise in the dataset.</p>
Step 3: Folder-Based Labeling	The dataset was organized into subfolders with the format: [gender]_[age_group]. This hierarchical folder structure served as a simple yet effective labeling method. During preprocessing, labels were automatically inferred from the folder names using scripts.
Step 4: Dataset Splitting	<p>After filtering, a stratified split was applied to divide the dataset into: Training set (80%), Validation set (10%), Test set (10%)</p> <p>This ensured that all eight demographic classes (male_child, female_teen, etc.) were proportionally represented across the splits. A custom Python script iterated through all folders and performed the split while preserving class balance.</p>
Step 5: Statistics Generation	<p>To validate the dataset structure and class balance, custom pandas-based summaries were generated showing:</p> <ul style="list-style-type: none"> <li>• Total images per class</li> <li>• Distribution across train, val, test folders</li> <li>• Gender and age group breakdowns</li> </ul> <p>This data was later used for visualization and to inform training parameters (e.g., class weights).</p>

Table 5: Data Collection Process

### 2.3. Data Cleaning & Preprocessing

To ensure the effectiveness and stability of the deep learning model, several preprocessing steps were applied to the raw image dataset. These steps aimed to normalize image properties, augment the training set for better generalization, and address class imbalance in both gender and age group labels.

1. Image Resizing and Normalization	<p>All images were resized to 224×224 pixels, matching the input shape required by the pre-trained ResNet50 architecture. This also standardized the input dimensions across the entire dataset.</p> <p>Following resizing, pixel values were normalized to the range [0, 1] by dividing each pixel intensity by 255. This helped stabilize the training process and allowed faster convergence during optimization.</p>
2. Data Augmentation (Training Set Only)	<p>To increase model robustness and prevent overfitting, real-time augmentation was applied to the training set using TensorFlow's <i>ImageDataGenerator</i> or custom <i>layers.Random*</i> augmentations. The augmentation techniques included:</p> <ul style="list-style-type: none"> <li>• Random horizontal flip</li> <li>• Random rotation (<math>\pm 15</math> degrees)</li> <li>• Random zoom (<math>\pm 10\%</math>)</li> <li>• Random brightness adjustments</li> <li>• Minor shifts in height/width (translation)</li> </ul> <p>These transformations simulated real-world conditions such as varied camera angles, lighting changes, and image imperfections.</p>
3. One-Hot Encoding and Label Formatting	<p>Although each image carried two labels (gender and age group), these were processed using separate label encoders:</p> <ul style="list-style-type: none"> <li>• Gender encoded as: 0 (Male); 1 (Female)</li> <li>• Age group encoded as: <ul style="list-style-type: none"> <li>○ 0: Child</li> <li>○ 1: Teen</li> <li>○ 2: Adult</li> <li>○ 3: Elderly</li> </ul> </li> </ul>

	<p>During training, each image was paired with two outputs using a multi-output setup, and both were treated as categorical cross-entropy classification problems.</p>
4. Handling Class Imbalance	<p>While the dataset was relatively balanced across the eight combined classes, minor differences remained. To address this:</p> <ul style="list-style-type: none"><li>• Class weights were computed separately for both the gender and age group labels using <code>sklearn.utils.class_weight.compute_class_weight()</code>.</li><li>• These weights were passed to the loss function during model training to give higher importance to underrepresented classes, especially for age categories like Child and Elderly.</li></ul>
5. Data Pipeline Integration	<p>All preprocessing steps were integrated into a TensorFlow data pipeline using <code>image_dataset_from_directory()</code> for loading and batching, and <code>prefetch()</code> for GPU acceleration. This ensured efficiency and scalability during model training on Colab.</p>

Table 6: Data Cleaning &amp; Preprocessing

## 2.4. Data Distribution & Visualization

A total of 4,139 images are labeled as Female, while 4,101 are labeled as Male. This near-equal distribution is clearly visualized in both bar and pie charts. The dataset is also well-balanced across the four age groups:

- Child: 2,010 images (24.4%)
- Teen: 2,115 images (25.7%)
- Adult: 2,090 images (25.4%)
- Elderly: 2,025 images (24.6%)

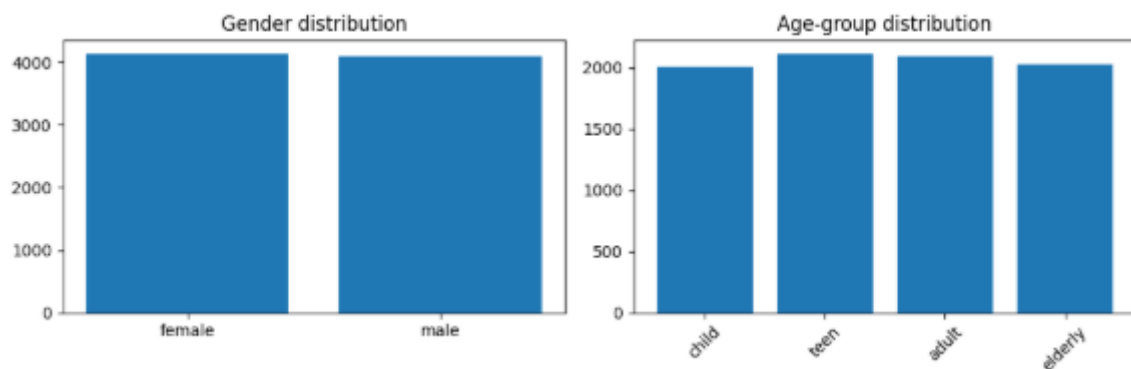


Figure 4: Bar Chart Distribution

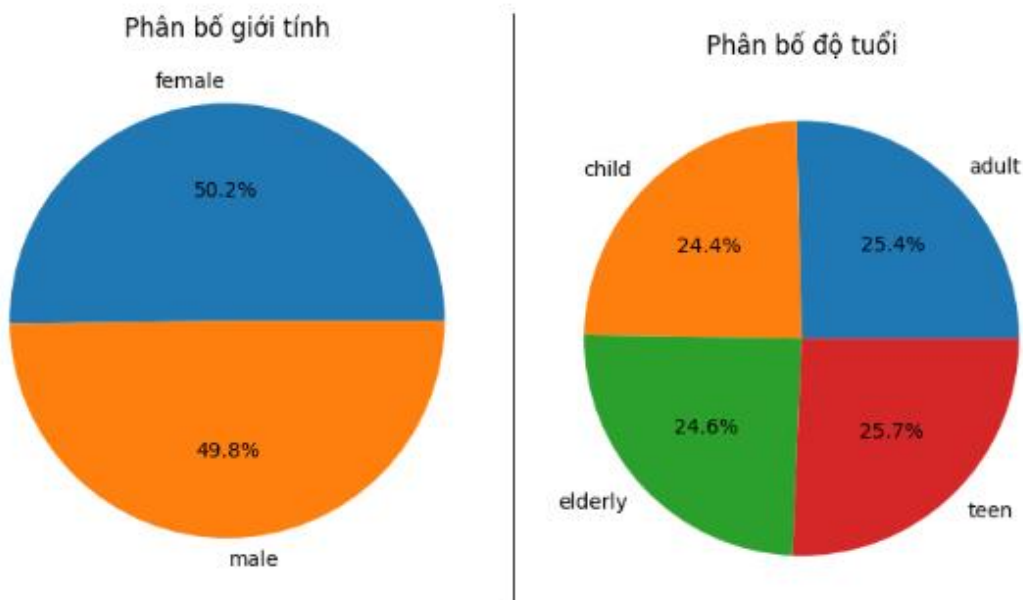


Figure 5: Pie Chart Distribution

Each of the eight demographic classes (e.g., male\_child, female\_adult) contains between 1,001 and 1,060 samples. This strong class balance supports stable model learning without heavy reliance on oversampling or regularization.

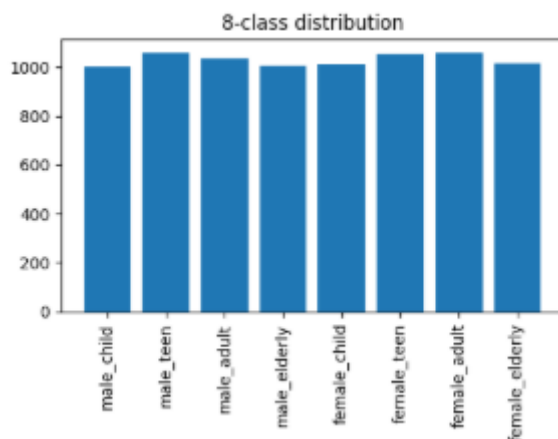


Figure 6: Combined 8-Class Distribution

The bar chart below shows the number of images for each class split across training, validation, and test. All classes are well represented across all three splits, ensuring fair evaluation. sets.

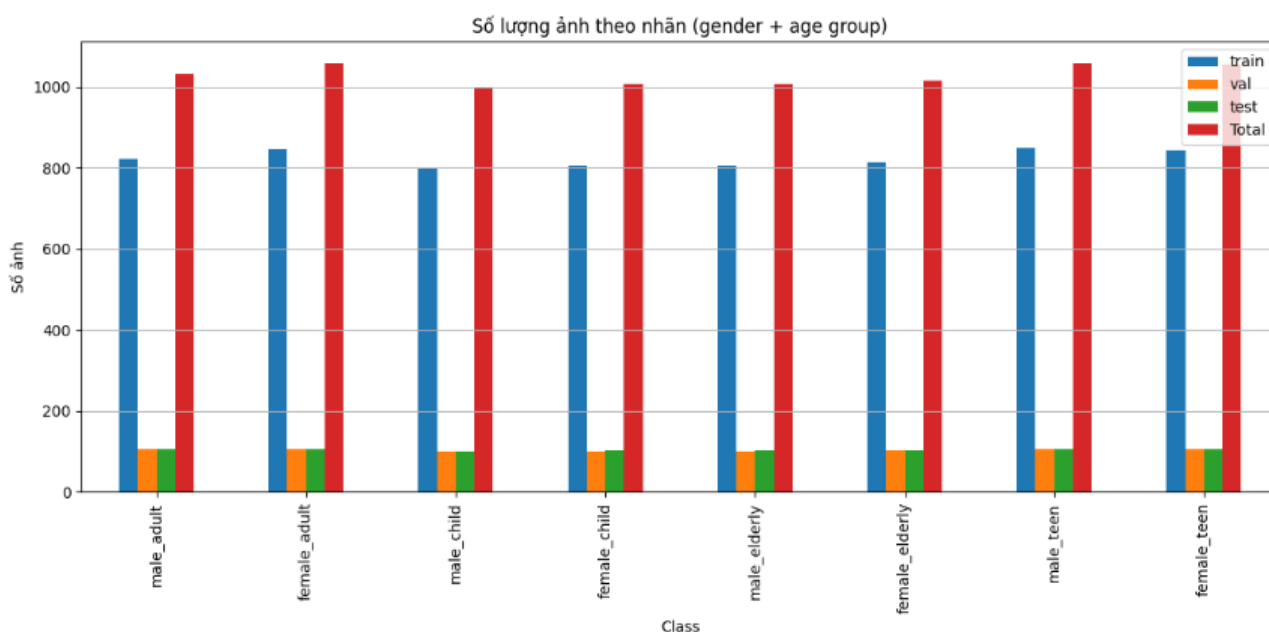


Figure 7: Class Distribution Across Train/Val/Test

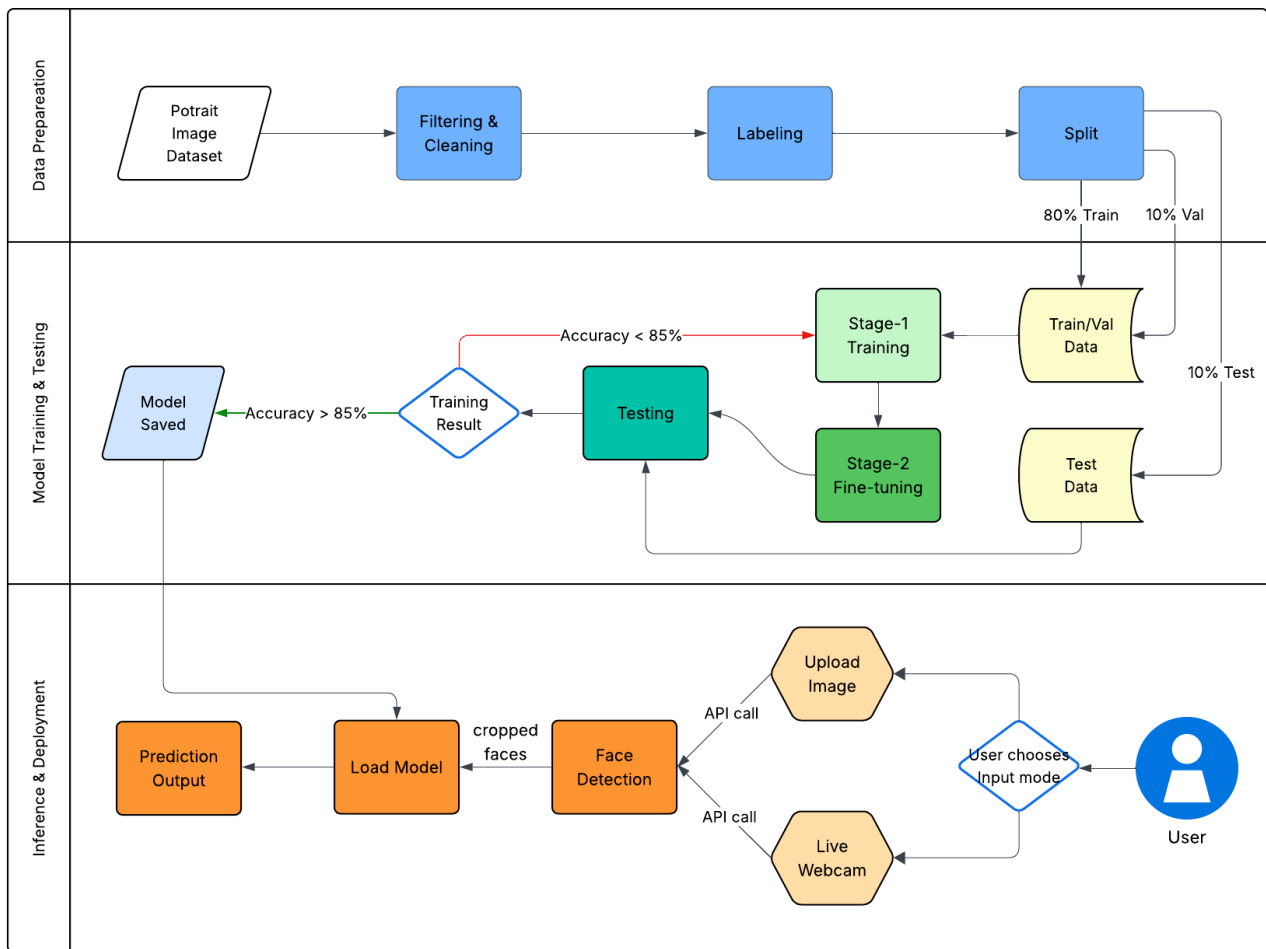


Figure 8: Data Flow Diagram

To complement the detailed description of the dataset pipeline, the diagram below outlines the full data flow across the system. It starts from the raw Vietnamese portrait image dataset and shows how the data is filtered, labeled, and used for model training. The flow also includes the inference stage, where user input is processed in real-time and visual predictions are returned via the web application. This visualization enhances understanding of how data transitions through both training and deployment environments.

## CHAPTER III: MODEL ARCHITECTURE & TRAINING PIPELINE

### 3.1. Overview of Approach

This project adopts a multi-output deep learning approach to simultaneously classify both gender and age group from a single facial image. Given the dual-label nature of the task, a shared feature extraction backbone is combined with two parallel classification heads—one for each output. This architecture offers several advantages:

- **Parameter efficiency:** Shared convolutional layers reduce redundancy.
- **Joint optimization:** Training both tasks together may improve overall feature learning via inductive transfer.
- **Real-time readiness:** A unified model reduces inference latency compared to running two separate models.

#### 3.1.1. Model Selection

To leverage the benefits of transfer learning, ResNet50 was chosen as the base model. It is a widely adopted convolutional neural network (CNN) architecture known for its residual connections and strong performance in image classification tasks (He et al., 2016). The ResNet50 backbone was initialized with ImageNet pre-trained weights and modified as follows:

- The top layer (include\_top=False) was removed.
- A GlobalAveragePooling2D layer was applied.
- The output was branched into:
  - A Dense layer for gender classification (binary softmax)
  - A Dense layer for age group classification (4-class softmax)

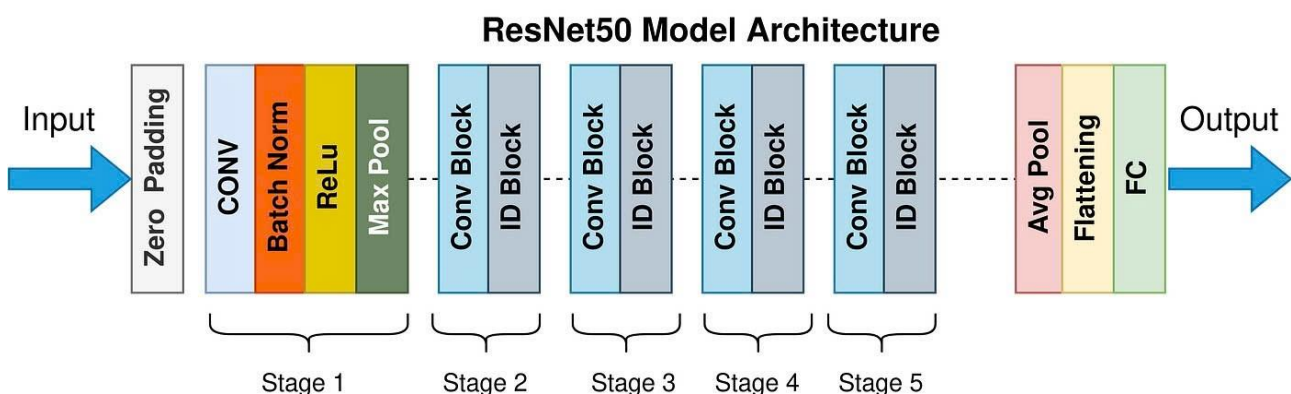


Figure 9: ResNet50 Model Architecture

**3.1.2. Multi-Output Setup**

This dual-head model outputs two probability distributions:

output_gender	→	shape: (None, 2)
output_age	→	shape: (None, 4)

Each output is trained using categorical cross-entropy loss, and both losses are combined using equal weighting:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{gender}} + \mathcal{L}_{\text{age}}$$

The final compiled model is optimized using Adam, and accuracy is monitored separately for each output.

**3.1.3. Training Strategy Highlight**

The model is trained in two stages:	Stage 1: Freeze the ResNet50 backbone and train only the classification heads.
	Stage 2: Unfreeze selected layers of the backbone and fine-tune the entire model.
Callbacks include:	Early stopping (per output)
	ReduceLROnPlateau
	ModelCheckpoint
Class imbalance is addressed using class weights per output.	
Input data is loaded via TensorFlow datasets with augmentation applied only to training batches.	

Table 7: Training Strategy Highlight



### 3.2. Model Architecture (ResNet50-based Multi-output)

This project implements a multi-output deep learning model designed to predict both gender and age group from a single facial portrait. The architecture is based on the widely adopted ResNet50 convolutional neural network (CNN), which serves as a feature extractor, followed by custom fully-connected layers that branch into two classification outputs. This design not only optimizes parameter sharing but also supports real-time deployment through efficient computation.

#### 3.2.1. Backbone Feature Extractor: ResNet50

At the core of the architecture lies the ResNet50 model, which was pre-trained on ImageNet and loaded with `include_top=False` to remove its original classification head. ResNet50 is composed of 50 convolutional layers and introduces residual connections that facilitate the training of deep networks by preserving gradients during backpropagation (He et al., 2016). The model was initialized with `weights="imagenet"` and kept frozen during the initial training phase, treating it as a fixed feature extractor. This allowed the model to leverage learned general-purpose image features while avoiding overfitting on a relatively small dataset.

The input to the model is a  $224 \times 224$  RGB image, consistent with standard ResNet input dimensions.

<code>include_top = False</code>	removes the default ImageNet classification head.
<code>input_shape = (224, 224, 3)</code>	
<code>weights = "imagenet"</code>	
<code>trainable = False</code>	during initial training; partially unfrozen in fine-tuning.

Table 8: Configuration details

#### 3.2.2. Global Average Pooling Layer

Following the ResNet50 base, a GlobalAveragePooling2D layer is applied to convert the final convolutional feature map (shape:  $7 \times 7 \times 2048$ ) into a 2048-dimensional vector. This technique averages each feature map across its spatial dimensions, significantly reducing the number of parameters compared to a Flatten operation while retaining essential information. It also improves generalization by suppressing spatial overfitting, making it a widely accepted choice in modern CNN architectures.

### 3.2.3. Custom Fully Connected Layers (Head Block)

To further process the extracted features, the model includes two dense layers with ReLU activation, interleaved with dropout layers for regularization:

Dense Layer 1	1024 units, ReLU activation
Dropout Layer	Rate = 0.5
Dense Layer 2	512 units, ReLU activation
Dropout Layer	Rate = 0.5

Table 9: Added Layers

These fully connected layers act as a shared "head" for both classification tasks. They allow the model to learn complex nonlinear combinations of the visual features extracted by ResNet50 and provide robust representations for downstream decision-making. The use of dropout with a relatively high rate (50%) helps mitigate overfitting, which is particularly important given the limited size of the dataset.

### 3.2.4. Output Heads

The model branches into two parallel classification heads:

(a) Gender Classification Head	Dense(2, activation='softmax')
	Output shape: (None, 2)
	Predicts probability distribution over two classes: <i>Male</i> and <i>Female</i> .
(b) Age Group Classification Head	Dense(4, activation='softmax')
	Output shape: (None, 4)
	Predicts probability distribution over four classes: <i>Child</i> , <i>Teen</i> , <i>Adult</i> , <i>Elderly</i> .

Table 10: Dual Output Heads

Both output layers produce probability distributions over their respective class sets and are trained using categorical cross-entropy loss functions. Each task is treated independently in terms of prediction but shares the same backbone and intermediate representation.

### 3.2.5. Model Compilation and Loss Function

The model was compiled using the Adam optimizer with a learning rate of  $1e-4$ , which is commonly used for fine-tuning pre-trained models. Two loss functions were defined—one for each output head:

- `categorical_crossentropy` for gender classification (2 classes)
- `categorical_crossentropy` for age group classification (4 classes)

Instead of assigning manual loss weights, the model was trained using default equal weighting (1.0 each). Accuracy metrics were also tracked individually for each output, allowing us to monitor the performance of both tasks during training.

The final compilation code is summarized below:

```
losses = {"gender_output": "categorical_crossentropy",
          "age_output": "categorical_crossentropy"}

metrics = {k: "accuracy" for k in losses}

model.compile(optimizer=tf.keras.optimizers.Adam(1e-4),
              loss=losses,
              metrics=metrics)
```

*Figure 10: Model Compilation code*

This compilation setup ensures that both outputs are optimized simultaneously without requiring explicit loss balancing. Monitoring separate accuracy metrics helps identify whether either output is underperforming during training.

### 3.2.6. Parameter Summary & Efficiency

- **Base model (ResNet50):** ~23 million parameters (frozen during Stage 1)
- **Head layers:** ~2.6 million parameters
  - 2048 → 1024: ~2.1M
  - 1024 → 512: ~524K
  - Output layers: ~3.5K total

Despite the relatively large backbone, the model remains efficient due to shared computation and the use of lightweight dense heads. This makes it feasible for deployment in web applications or edge devices with GPU support.

### 3.2.7. Integration of Data Augmentation

To improve generalization and make the model more robust to variations in pose and lighting, a custom augmentation block - defined as *build\_augmentation()* - was inserted directly into the model pipeline. This layer includes random transformations such as:

- Horizontal flipping
- Rotation
- Zooming
- Shifting (height/width)

By integrating augmentation into the model graph itself, the system ensures consistent preprocessing and allows for seamless experimentation, especially when deployed using TensorFlow's high-level APIs.

### 3.2.8. Model Name & Modularity

The complete model is referred to as "GAG" and is saved in .keras format for reuse in inference pipelines. Its modular construction allows for easy extensions, such as swapping out the ResNet50 backbone for alternatives like ResNet101 or InceptionV3, or adding new branches for emotion or ethnicity classification. The dual-output architecture also ensures reduced inference time compared to running two separate models.

### 3.3. Training Strategy

To ensure effective optimization while avoiding overfitting, a two-phase training strategy was adopted. This approach is widely used when fine-tuning pre-trained convolutional neural networks, such as ResNet50, and balances the benefits of transfer learning with task-specific adaptation.

#### 3.3.1. Phase 1: Transfer Learning with Frozen Base

In the initial phase, the ResNet50 backbone was frozen (`base_model.trainable = False`), and only the custom head layers were trained. These included the dense layers, dropout layers, and the two softmax output heads for gender and age classification. This allowed the model to leverage general visual representations learned from the ImageNet dataset while adapting the new layers to the specific dual-task classification problem.

Optimizer	Adam with learning rate = $1e-4$
Loss functions	<i>categorical_crossentropy</i> for both outputs No explicit <i>loss_weights</i> were used; each output was equally optimized by default.
Metrics	Accuracy for both <i>gender_output</i> and <i>age_output</i>
Batch size	32
Epochs	20

Table 11: Key configurations

The compilation setup was as Figure 6. Training was conducted using TensorFlow in a GPU-enabled Google Colab environment. After each epoch, both training and validation accuracy/loss were monitored for each output.

To monitor training and enhance stability, three key callbacks were applied:

- **ModelCheckpoint:** Saved the model whenever the combined validation loss improved. The best model (`val_loss` minimum) was retained to prevent overfitting.
- **EarlyStopping:** Monitored `val_loss` with patience = 10, stopping training early if no improvement was observed. The callback restored the best weights automatically.
- **ReduceLROnPlateau:** Reduced the learning rate by a factor of 0.2 if validation loss plateaued for 5 epochs, with a minimum learning rate (`min_lr`) of  $1e-6$ .

These callbacks significantly improved training efficiency and generalization, especially when convergence slowed or validation performance began to degrade.

### 3.3.2. Fine-Tuning

After initial training, the model was optionally fine-tuned by unfreezing part or all of the ResNet50 backbone. This allowed the pre-trained filters to adapt further to the specific patterns in the dataset, particularly subtle visual cues related to aging or facial differences across genders.

Fine-tuning strategy:

- Unfreeze selected layers (e.g., from layer index 140 onward) or the entire base.
- Recompile the model with a lower learning rate (e.g., 1e-5).
- Continue training for up to **30 epochs** with the same callbacks (patience counters reset).

In this project, the fine-tuning phase was considered optional. Due to time constraints and already satisfactory performance, it may not have been fully executed. However, the code and infrastructure for fine-tuning were prepared and included.

A custom data augmentation layer—defined via the `build_augmentation()` function—was incorporated directly into the model graph. This augmentation block included:

- Random horizontal flipping
- Random rotation
- Random zooming and shifting

By integrating augmentation inside the model pipeline, the approach ensured real-time, on-GPU preprocessing during training, leading to more robust learning without external augmentation logic.

Framework	TensorFlow 2.x with Keras API
Runtime	Google Colab with GPU acceleration
Model saving	Best model saved to .keras format in Google Drive
Random seed	Fixed <code>random_state = 42</code> for consistent data splits

*Table 12: Training Environment and Reproducibility*

All training procedures, data pipelines and callbacks were implemented in a modular fashion for reusability and extension.

### 3.4. Inference Pipeline Design

Face Detection and Cropping	<p>The pipeline begins by detecting faces in the input image using MTCNN (Multi-task Cascaded Convolutional Networks). MTCNN is a robust and lightweight detector that performs well on frontal and semi-profile faces under varying lighting conditions.</p> <p>For each detected face:</p> <ul style="list-style-type: none"> <li>• The bounding box is adjusted with a margin to capture facial context.</li> <li>• The cropped face is resized to 224×224 pixels to match the model's input size.</li> </ul> <p>In multi-face scenarios (e.g., group photos), the pipeline extracts each face individually and stores their bounding boxes for later annotation.</p>
Image Preprocessing	<p>Each cropped face image undergoes the same normalization as during training:</p> <ul style="list-style-type: none"> <li>• Pixel values are scaled to [0, 1] by dividing by 255.</li> <li>• Images are optionally expanded in batch dimension (batch_size=1) for inference.</li> <li>• Augmentation is not applied during inference to ensure consistency.</li> </ul>
Model Inference (Multi-Output Prediction)	<p>The preprocessed face image is passed into the trained GAG model using the <i>predict()</i> method. The model outputs a pair of softmax probability vectors—one for gender and one for age group.</p> <pre>pred = model.predict(image) gender_label = np.argmax(pred[0]) age_label = np.argmax(pred[1])</pre> <p>The predicted labels (gender_label, age_label) are then mapped to their corresponding class names:</p> <ul style="list-style-type: none"> <li>• Gender: “Male” or “Female”</li> <li>• Age group: “Child”, “Teen”, “Adult”, or “Elderly”</li> </ul> <p>Confidence scores (maximum softmax probability) may also be retained and displayed alongside the prediction for transparency.</p>

Annotating and Returning Results	<p>The original input image is annotated using OpenCV or PIL by:</p> <ul style="list-style-type: none"> <li>• Drawing a bounding box around each detected face.</li> <li>• Placing the predicted label (e.g., "Male – Teen") near the face region.</li> <li>• Using distinct colors for gender and age categories to enhance readability.</li> </ul> <p>The annotated image is then:</p> <ul style="list-style-type: none"> <li>• <b>Returned as a NumPy array</b> for real-time streaming, or</li> <li>• <b>Encoded to base64</b> for rendering in a web interface.</li> </ul>
Application Modes	<p>The inference pipeline supports two main modes:</p> <p><b>(a) Static Image Upload</b></p> <p>Users upload an image file via the web interface. The image is processed and the annotated result is displayed on the same page.</p> <p><b>(b) Real-time Webcam Processing</b></p> <p>A separate loop captures frames directly from the webcam, processes each frame in real time, and displays annotated predictions. The system is capable of detecting and predicting multiple faces simultaneously.</p>
Model Flexibility and Switching	<p>The backend also supports <b>model switching</b>, allowing users to select from different pre-trained model architectures (e.g., ResNet50, ResNet101, InceptionV3). If a selected model is unavailable, a fallback message ("Model not found") is returned. This modular design supports future experimentation with different architectures.</p>

Table 13: Inference Pipeline Design



### 3.5. Model Management & Saving

#### 3.5.1. Model Saving Strategy

During training, the model was saved using Keras's native .keras format. The ModelCheckpoint callback was configured to monitor the validation loss and save the best-performing model weights automatically:

```
tf.keras.callbacks.ModelCheckpoint(os.path.join(MODEL_DIR, "resnet50_best.keras"),  
                                  monitor="val_loss", save_best_only=True),
```

Figure 11: Model Checkpoint Code

This ensured that the model with the lowest validation loss across all epochs was preserved. The .keras format retains not only the architecture and weights, but also optimizer states—making it ideal for resuming training or deploying without manual reconfiguration.

#### 3.5.2. Model Versioning and Folder Structure

```
GAG_Models/  
|  
├── resnet50_best.keras  
├── Resnet101_best.keras # (optional)  
├── inceptionv3_best.keras # (optional)  
└── ...
```

Table 14: Model Save Folder Structure

This allows flexible switching between architectures during inference or comparative evaluation. In the web application, a dropdown menu or backend logic can dynamically load the appropriate model based on user selection.

### 3.5.3. Model Loading for Inference

Since the models were saved in .keras format using ModelCheckpoint, they can be reloaded for inference using:

```
model = tf.keras.models.load_model("GAG_Models/resnet50_best.keras")
```

*Figure 12: .Keras Model Save Code*

This step is essential in the deployment phase, allowing the trained model to be used in separate applications without retraining.

### 3.5.4. Reusability and Expansion

Thanks to modular model construction (e.g., build\_model() function) and versioned saving:

- The same pipeline supports multiple architectures without duplicating logic.
- Newly trained models can be added to the GAG\_Models/ directory and become instantly available for inference.
- The saved models are compatible with platforms such as TensorFlow Lite, ONNX, or TensorFlow.js, supporting future deployment to edge devices or browsers.

### 3.5.5. Backup and Hosting

All trained models are backed up to Google Drive during training sessions on Google Colab. This avoids data loss due to runtime resets and facilitates sharing or downloading for local use.

The model management workflow in this project reflects best practices in modern ML engineering: checkpointing, structured saving, modular loading, and scalability for future updates. This ensures that the trained models are not only effective but also production-ready.

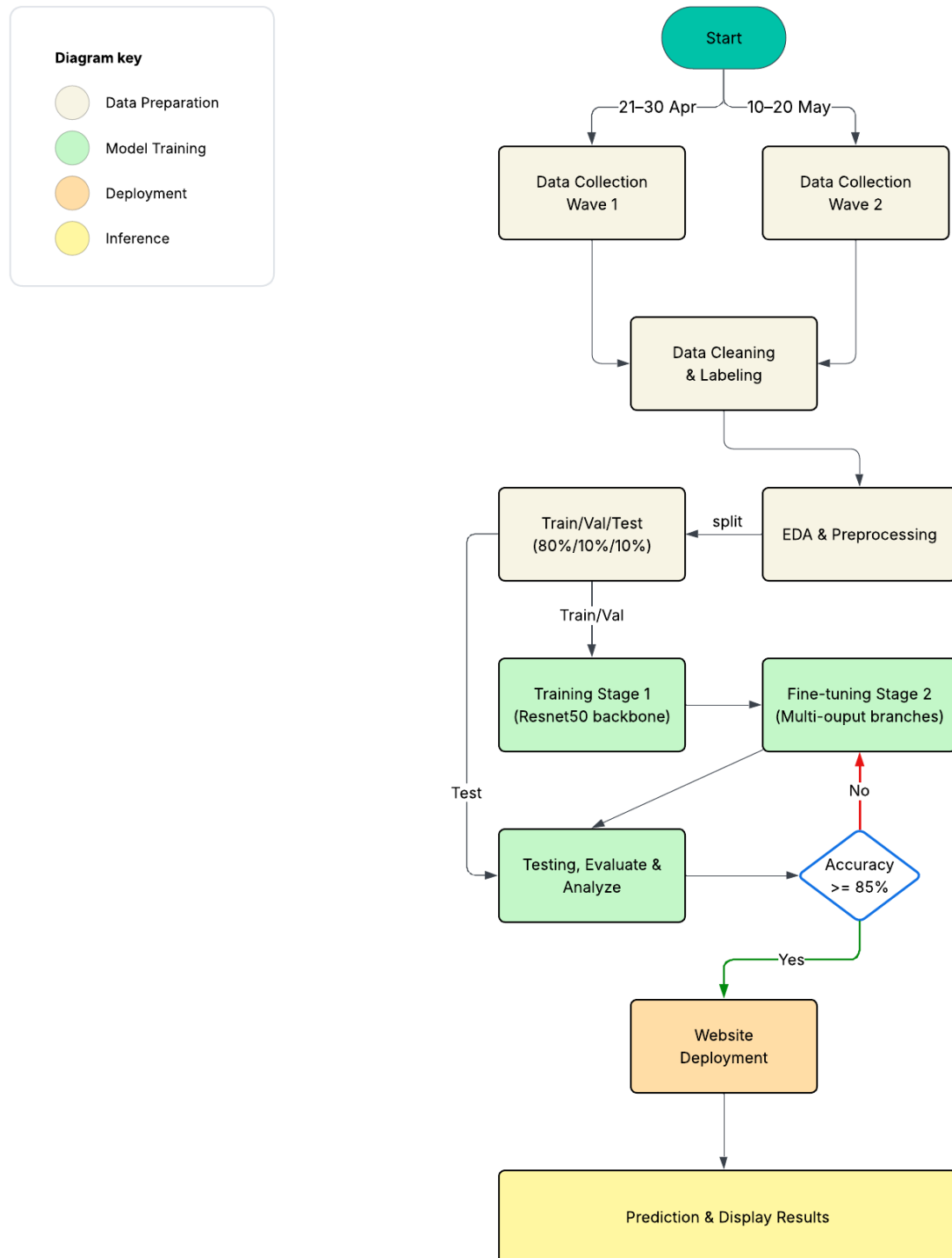


Figure 13: Project Workflow

To provide a consolidated overview of the development process, the following diagram illustrates the entire workflow of our gender and age classification project. It captures all major phases, including data preparation, model training, evaluation, deployment, and inference integration. This visual serves to highlight how different components—such as the training pipeline, model management, and real-time web application—are interconnected across stages of implementation.

## CHAPTER IV: EXPERIMENTAL RESULTS & ANALYSIS

### 4.1. Training & Validation Performance

To assess the model's learning capability, we conducted 30 training epochs using the fine-tuned ResNet50 architecture on the curated dataset. The model was compiled with the Adam optimizer (learning rate =  $1e-5$ ) and trained using a multi-output setup for both gender and age-group classification. Equal loss weights of 0.5 were assigned to both outputs to maintain balanced learning.

Throughout training, performance metrics—including categorical cross-entropy loss and classification accuracy—were monitored for both training and validation sets. As shown in the training log, both `gender_output_accuracy` and `age_output_accuracy` exhibited stable upward trends across epochs, with the final validation accuracies reaching approximately 0.924 for gender classification and 0.88 for age-group classification. Correspondingly, the loss values for both branches steadily decreased, indicating effective convergence of the model.

The training process was further stabilized by two callbacks: `ReduceLROnPlateau`, which dynamically adjusted the learning rate, and `EarlyStopping`, which prevented overfitting by halting training if no significant improvement was detected.

The performance trends are visually illustrated in the following learning curves (see Figure 14), where the loss and accuracy of both tasks are plotted over the training epochs.

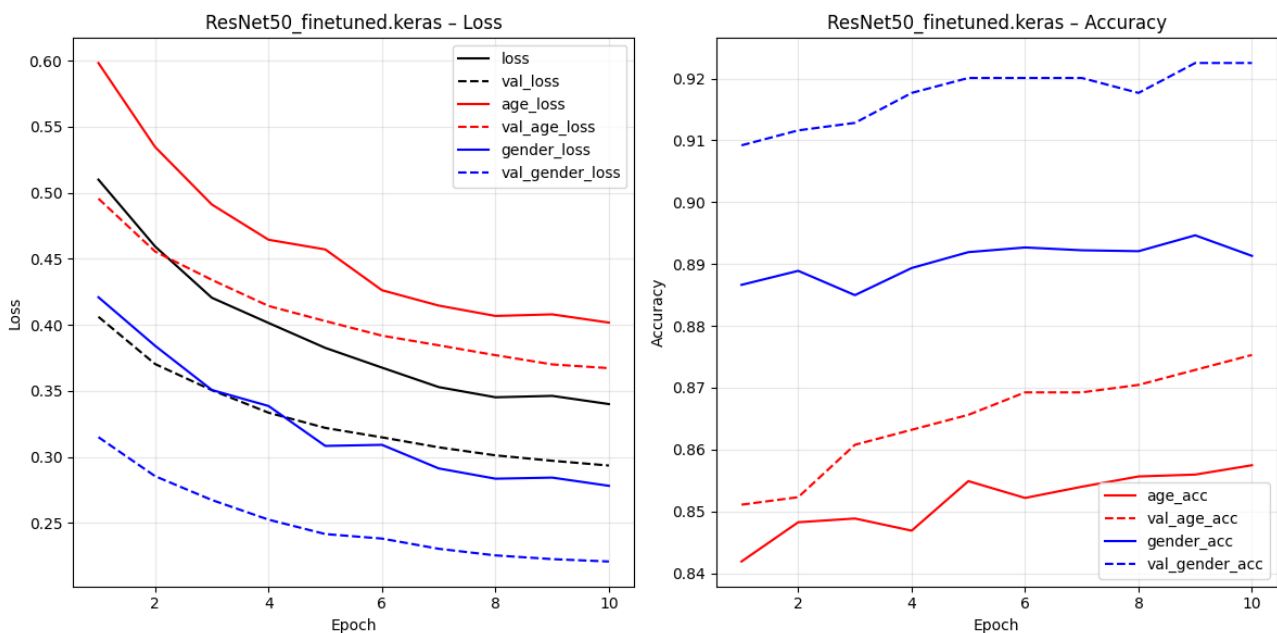


Figure 14: Side-by-side line plots of training & validation loss and accuracy curves.

## 4.2. Classification Reports

To evaluate the model's generalization on unseen data, we generated classification reports for both output branches using the test set. The reports present precision, recall, and F1-score for each class in the gender and age-group categories.

For the gender classification task, the model achieved high and balanced performance across both classes. Specifically, the precision, recall, and F1-score for female and male classes were all approximately 0.89–0.90, yielding an overall accuracy of 0.89. This consistency across metrics suggests that the model is not biased toward any particular gender class and is well-calibrated.

In the age-group classification task, the performance varied slightly across different age groups. The child and elderly categories achieved strong F1-scores of 0.91 and 0.88, respectively, while the teen category had a slightly lower recall (0.83), indicating some misclassifications with adjacent age groups. The adult class exhibited the lowest F1-score (0.77), which may stem from overlapping features shared with neighboring age groups such as teen and elderly. The overall accuracy for the age branch reached 0.86, with macro and weighted F1-scores maintaining consistency.

These classification reports confirm that the model performs robustly in both classification tasks, with higher consistency in gender prediction and slightly more variance in age classification.

Class	Precision	Recall	F1-Score	Support
Female	0.89	0.90	0.90	418
Male	0.90	0.89	0.89	414
<b>Accuracy</b>	—	—	<b>0.89</b>	832
<b>Macro Avg</b>	0.89	0.89	0.89	832
<b>Weighted Avg</b>	0.89	0.89	0.89	832

Table 15: Gender Classification Report

Class	Precision	Recall	F1-Score	Support
Child	0.92	0.90	0.91	203
Teen	0.94	0.83	0.89	212
Adult	0.73	0.81	0.77	213
Elderly	0.87	0.89	0.88	204
<b>Accuracy</b>	—	—	<b>0.86</b>	832
<b>Macro Avg</b>	0.87	0.86	0.86	832
<b>Weighted Avg</b>	0.87	0.86	0.86	832

Table 16: Age Group Classification Report

### 4.3. Confusion Matrix & F1-Score Analysis

To further analyze the model's performance beyond standard metrics, we examined the confusion matrices and macro F1-scores for both gender and age-group classification.

The confusion matrix for gender prediction shows that the model achieved highly balanced predictions across both classes. Specifically, 90% of female and 89% of male instances were correctly classified, with minimal confusion between the two. This result aligns with the classification report and confirms the model's strong discriminative power in binary classification tasks.

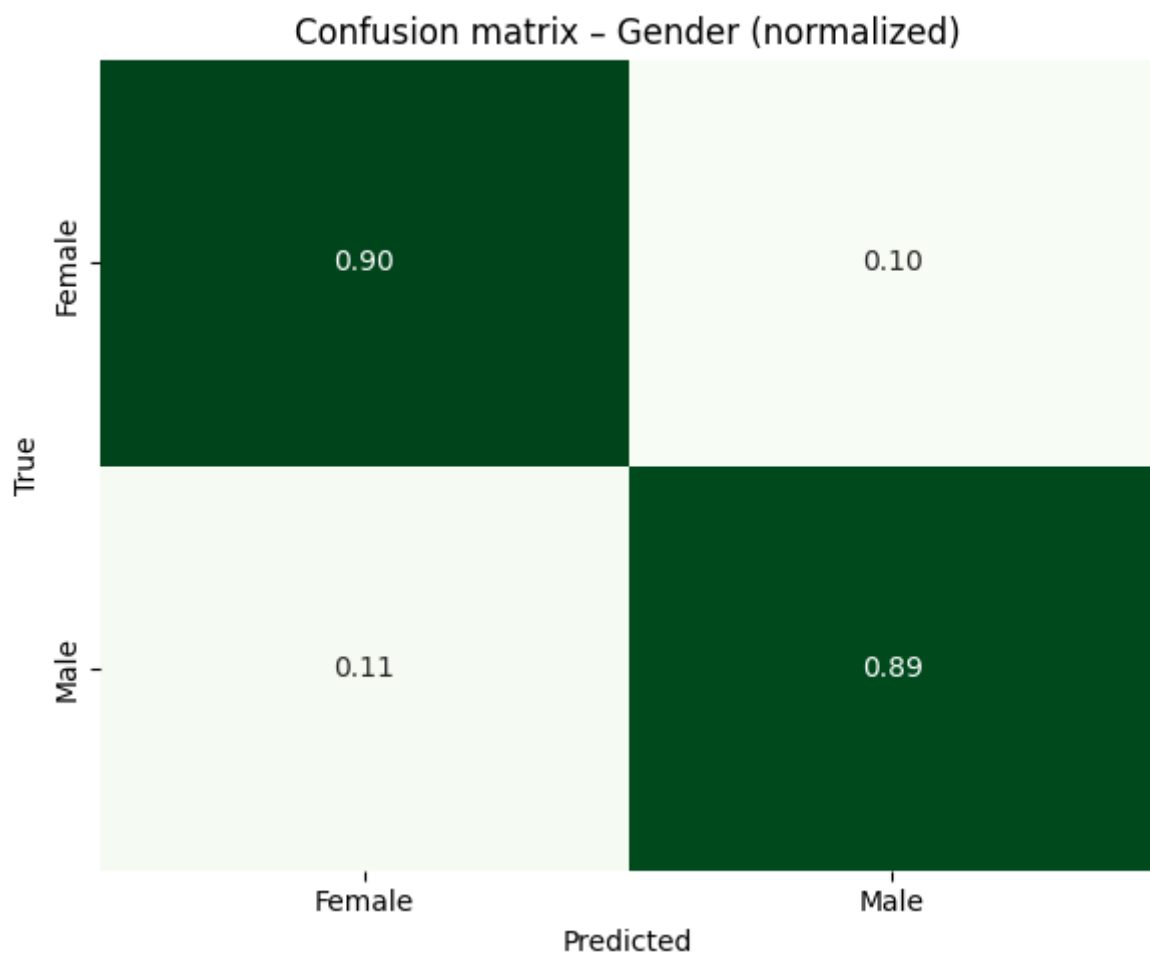


Figure 15: Confusion matrix for gender classification (normalized)

In contrast, the age-group confusion matrix reveals several insights. While the child (90%) and elderly (89%) groups showed high correct prediction rates, there was notable confusion between adult and teen classes. For example, 14% of teen samples were misclassified as adults, and 10% of adult samples were misclassified as elderly. This behavior is expected due to the overlapping facial features and transitional characteristics of individuals in adjacent age ranges.

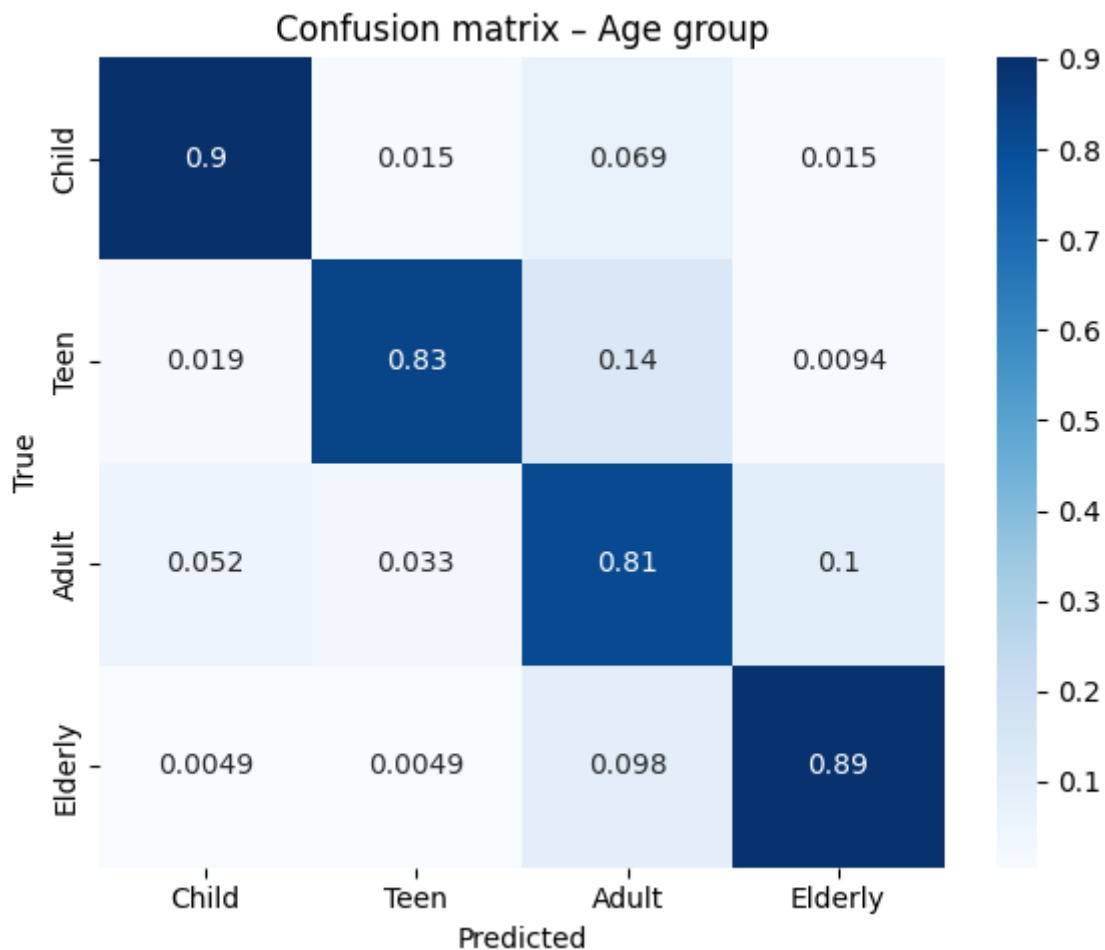


Figure 16: Confusion matrix – Age group

Overall, these matrices support the findings from the F1-score analysis: gender prediction remains consistently high, while age-group classification presents more challenges, particularly in middle-aged transitions.

#### 4.4. Error Cases & Performance Breakdown

While the model achieved promising results overall, a closer inspection of the misclassified samples reveals several limitations and areas for improvement.

For the gender classification task, most errors occurred in images where facial features were obscured by shadows, accessories (e.g., glasses or masks), or low-resolution quality. In rare cases, androgynous appearances or non-standard head poses also led to incorrect predictions. Nevertheless, these cases were relatively infrequent and did not significantly skew the overall distribution of results.

In the age-group classification task, errors were more pronounced. A significant proportion of misclassifications occurred between teen and adult, and between adult and elderly. These errors can be attributed to the subjective nature of age estimation, particularly in individuals with ambiguous facial features or inconsistent age appearances due to makeup, hairstyle, or expression. Moreover, visual overlaps among transitional age groups (e.g., late teens and young adults) pose a challenge even for human annotators.

Class imbalance also played a subtle role in performance variation. Despite efforts to curate a balanced dataset, some residual imbalance may have persisted at the fine-grained level of visual diversity (e.g., lighting or ethnic representation within each age class). This could have influenced the relatively lower F1-score observed for the adult group.


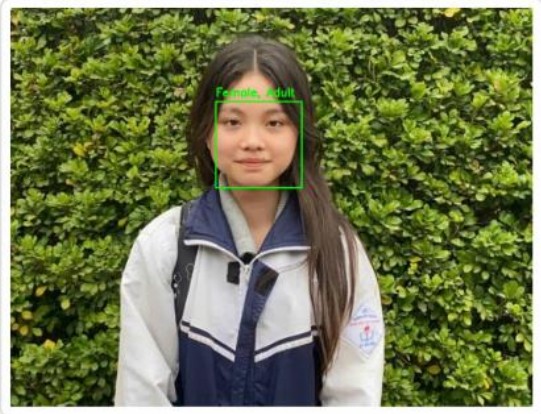
<p style="text-align: center; color: green;">Male, Adult</p>  <p style="text-align: center;"><i>Figure 17: Correct predicted label</i></p>	<p style="text-align: center; color: green;">Female, Adult</p>  <p style="text-align: center;"><i>Figure 18: Wrong predicted label</i></p>
---	--

Table 17: Misclassified Examples



#### 4.5 Model Comparison & Selection

Backbone	Best Gender Acc (%)	Best Age Acc (%)	Latency (ms/image)
<b>ResNet50</b>	92.4	88	183.6
<b>ResNet101</b>	93.3	87.5	231.0
<b>EfficientNetB0</b>	93.0	87.0	376.3
<b>InceptionV3</b>	76.6	66.7	430.6

Table 18: Model Comparison

Accuracy values represent the validation performance achieved during training. Latencies are averages over 100 inference runs, including preprocessing overhead.

Trade-off: Accuracy vs Inference Speed

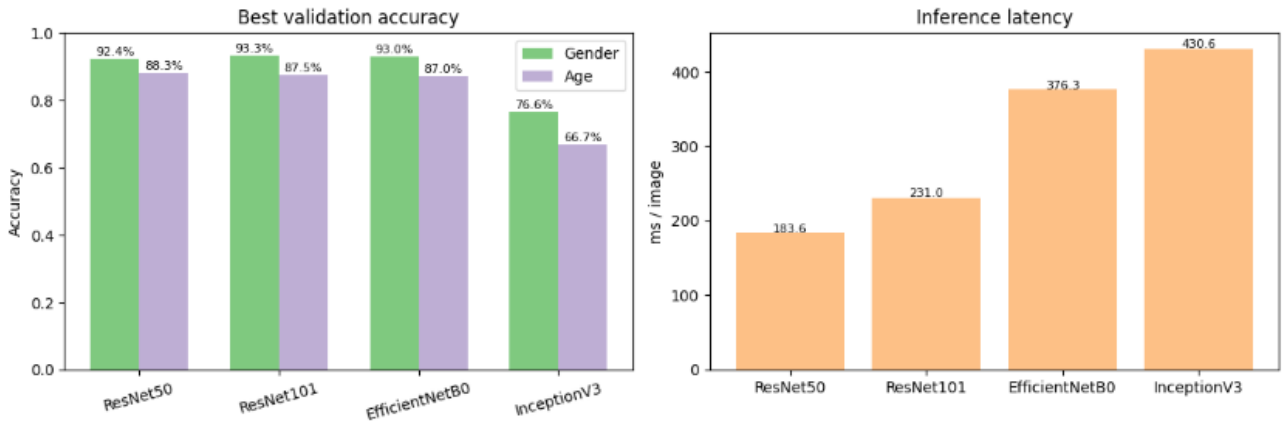


Figure 19: Trade-off: Accuracy vs Inference Speed

In our comparative evaluation of four backbone architectures (Table 18; Figure 19), under this criterion, ResNet50 remains the most balanced choice for real-time deployment: it achieves 92.4 % gender accuracy and 88.3 % age accuracy with only 183.6 ms per inference. Although ResNet101 slightly outperforms ResNet50 on gender (93.3 %), its latency jumps to 231 ms, which is suboptimal for web applications. EfficientNetB0 posts comparable accuracies (93.0 % / 87.0 %) but incurs a much higher delay (376.3 ms), while InceptionV3 fails to converge effectively in our setup (76.6 % / 66.7 %) and exhibits the slowest inference (430.6 ms). Consequently, we selected ResNet50 as the production backbone, since it consistently surpasses 90 % gender accuracy and 88 % age accuracy within a sub-200 ms latency budget.

## CHAPTER V: DEPLOYMENT & DEMO SYSTEM

### 5.1. Web Interface Overview

To enhance the usability and accessibility of the gender–age classification system, a fully functional web application was developed using the Flask framework. The application allows users to interact with the trained AI models directly via a browser without requiring technical knowledge or code interaction.

The landing page introduces the project and provides two main interaction modes:

- **Image Upload:** Users can upload a portrait image from their device to receive predictions.
- **Webcam Mode:** Users can activate their webcam, capture a frame, and obtain real-time predictions.

The web interface was designed with simplicity and responsiveness in mind using Bootstrap 5, ensuring compatibility across devices. It offers:

- Drop-down selection of available models (ResNet50, ResNet101, InceptionV3, EfficientNetB0)
- Real-time display of results, including annotated image and label
- FPS indicator for performance awareness during webcam usage

The UI is complemented with helpful tooltips and a footer section for user education about age estimation and the capabilities of the AI system.

### Giới thiệu Dự án

Dự án này được phát triển như một phần bài tập cuối kỳ trong học phần Trí tuệ Nhân tạo (AI) - INS3080 - Trường Quốc tế – ĐHQGHN.

Mục tiêu của dự án là xây dựng mô hình học sâu có khả năng **phân loại giới tính** và **dự đoán nhóm tuổi** từ ảnh chân dung, áp dụng mô hình **ResNet50** kết hợp kỹ thuật **Transfer Learning**. Ứng dụng được triển khai bằng **Flask**, giúp người dùng có thể tương tác trực tiếp thông qua giao diện web đơn giản, dễ sử dụng.

Mô hình đã được huấn luyện và tinh chỉnh trên tập dữ liệu 8000 ảnh chân dung được gán nhãn giới tính (Nam/Nữ) và nhóm tuổi (Child, Teen, Adult, Elderly). Kết quả dự đoán hiển thị cùng xác suất giúp người dùng có cái nhìn rõ hơn về hiệu quả mô hình.

## Dự đoán Giới tính & Nhóm tuổi từ ảnh chân dung

Tải ảnh hoặc dùng webcam để trải nghiệm.

Chọn ảnh chân dung

Chọn tệp

Không tệp nào được chọn

Chọn mô hình AI

ResNet50

Đoán thử xem


Hoặc sử dụng Webcam Realtime

Bắt đầu Webcam

Dừng Webcam


Chụp & Dự đoán

✦ Đây là một số mẹo dành cho bạn




#### Tại sao một số người trông trẻ hơn tuổi thật?

Tuổi khuôn mặt khác tuổi thật; lối sống, biểu cảm, di truyền đều ảnh hưởng.



#### Làm thế nào để nhìn trẻ hơn?

Giữ sức khỏe, giảm căng thẳng, chăm sóc da, sống lành mạnh...



#### AI đoán tuổi như thế nào?

Mô hình học sâu phân tích đặc trưng khuôn mặt để đưa ra dự đoán.

© 2025 GAG Project – Nhóm 5 | INS3080 – Lecturer: Dr Ha Manh Hung  
Thành viên nhóm: Nguyen Minh Thuy – Hoang Phuc Nhan – Tran Thi Khanh Huyen – Vi Thi Thuy Tien

Figure 20: Web Interface Overview

43

## 5.2. Image Upload & Real-time Webcam Support

Users can select an image file (JPEG or PNG) from their local device and submit it through a structured HTML form. Upon submission, the Flask route `@app.route('/predict', methods=['POST'])` processes the request. The image is temporarily saved using the `secure_filename()` function to prevent file injection issues.

Internally, the backend leverages the `predict.py` module, where the image is read using OpenCV (`cv2.imread`), converted to RGB, and passed through the chosen model for prediction. Detected faces are annotated with bounding boxes and labels, and the final annotated image is encoded into a base64 string and rendered directly on the web interface.

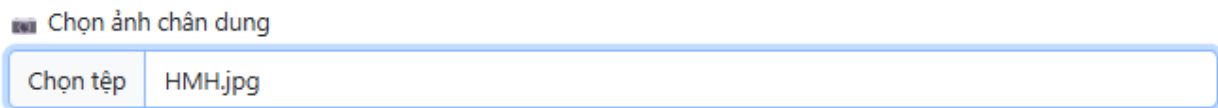


Figure 21: Image Upload Feature

In addition to file upload, the system integrates webcam-based detection using the `@app.route('/realtime', methods=['POST'])` endpoint. Users can activate their camera via JavaScript in the frontend. A single frame is captured and sent as a JPEG image (base64-encoded) to the backend.

The frame is then decoded using `cv2.imdecode()`, processed by the same face detection and prediction pipeline, and annotated with gender and age group predictions. The response includes: A base64 image with bounding boxes and labels; A list of predicted labels; An FPS (frames per second) value calculated using Python's `time` module. This setup allows users to receive instant visual feedback, making the experience interactive and engaging.



Figure 22: Webcam Feature

### 5.3. Multi-face Detection & Annotation

A key enhancement in the deployment phase of GAGNet is the ability to detect and classify multiple faces in a single frame or uploaded image. This feature increases the system's real-world applicability, particularly in group scenarios such as classroom settings, public monitoring, or family applications.

The implementation leverages the MediaPipe Face Detection API, accessed through the custom wrapper in `mp_detector.py`. Compared to traditional detectors like MTCNN or Haar cascades, MediaPipe offers superior speed and robustness, especially in real-time applications.

Upon receiving an input image (either from upload or webcam), the following pipeline is executed:

1. The image is first resized and converted to RGB.
2. MediaPipe detects all visible faces in the frame and returns bounding boxes.
3. For each bounding box:
  - A cropped, margin-padded version of the face is extracted.
  - The cropped face is resized to 224×224 and passed through the classification model.
  - The predicted gender and age group are stored.
4. The original image is annotated with:
  - A bounding box for each detected face
  - A label containing both predictions (e.g., “Female – Teen”)

The annotated image is returned in base64 format and displayed directly on the interface.

This multi-face detection capability has been thoroughly tested in both image upload and webcam modes, showing robust performance under varied conditions such as lighting changes and background complexity.

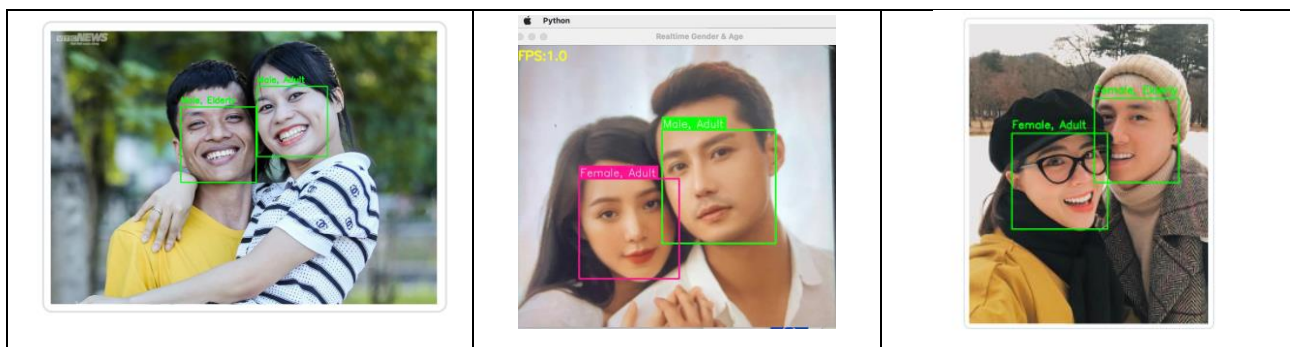
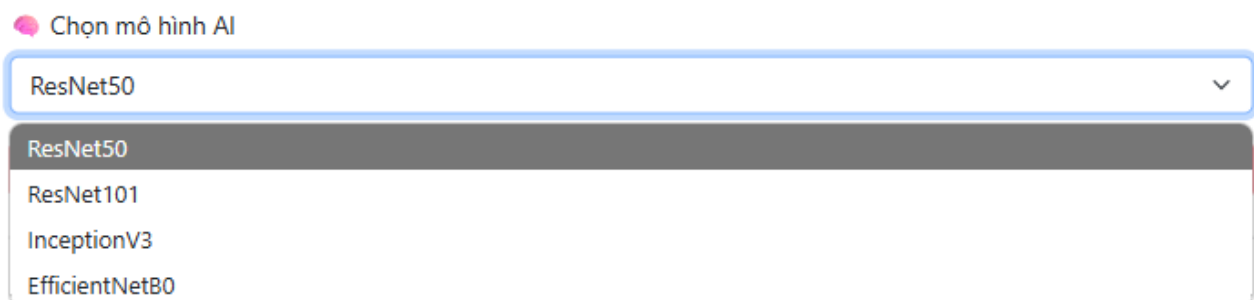


Table 19: Multi-face Detection Example

## 5.4. Model Switching Feature

To allow flexible experimentation and real-time performance comparison between different deep learning architectures, the application integrates a dynamic model switching feature. Users can select their preferred model from a drop-down menu on the interface, choosing among options such as: ResNet50, ResNet101, InceptionV3, EfficientNetB0



This functionality is supported by the Flask backend logic defined in the `load_selected_model(model_name)` function within `app.py`. When a request is received, the server performs the following:

1. Extracts the selected model name from the form (`model = request.form.get('model', current_model_name)`).
2. Attempts to load the corresponding .keras model file from the `./GAG_Models` directory using:  
`model = tf.keras.models.load_model(model_path)`
3. If the model file is missing or invalid, an error is returned to alert the user.

**Model not found**



*Figure 23: Model Missing Alert*

4. Otherwise, the model is cached in memory and used for subsequent predictions.

This design supports on-demand switching without requiring server restarts. Users can test how different architectures behave on the same input, enabling side-by-side evaluation of:

- Prediction accuracy
- Processing speed (FPS)
- Visual confidence

### 5.5. Limitations of Current Deployment

Limited Real-Time Speed on CPU	The current system runs inference using TensorFlow on a local CPU. As a result, real-time webcam processing can experience delays, particularly when handling multiple faces in high-resolution frames. Integrating GPU acceleration or converting models to TensorFlow Lite or ONNX format could significantly improve responsiveness.
Model Loading Overhead	Each model is dynamically loaded on user request, which can introduce latency if the selected model is large (e.g., ResNet101 or InceptionV3). Although the system supports switching, caching strategies for preloading frequently used models could improve responsiveness and reduce cold-start delay.
No Confidence Score Display	The current interface does not visualize prediction confidence scores or softmax probabilities. This limits transparency for borderline cases and reduces user trust in ambiguous predictions. Future iterations should display confidence thresholds or visual indicators (e.g., color-coded bounding boxes).
Lack of Image Quality Checks	There is no pre-validation to reject low-quality, blurry, or face-absent images. As a result, invalid inputs may lead to failed predictions or unhandled errors. Integrating basic quality control or feedback messages would improve robustness.
Browser and Security Constraints	Webcam access may be restricted or blocked depending on browser permissions or HTTPS enforcement. Additionally, the application currently stores uploaded images temporarily without automatic deletion, which may raise data privacy concerns if used in production.

In summary, the current system functions well for demonstration purposes but would require several technical improvements—particularly in inference speed, UI transparency, and robustness—to support broader real-world deployment.

## CHAPTER VI: DISCUSSION & CONCLUSION

### 6.1. Key Insights & Achievements

Effectiveness of Multi-Output Learning	The multi-branch architecture successfully enabled simultaneous training for both gender and age classification, improving computational efficiency and fostering shared feature learning. This strategy led to robust performance on both tasks, with final test accuracies of <b>89% for gender</b> and <b>86% for age group</b> .
Importance of Model Fine-Tuning	Fine-tuning the ResNet50 backbone rather than using it solely as a frozen feature extractor significantly enhanced performance. The addition of GlobalAveragePooling, Dropout, and two Dense layers further improved generalization and reduced overfitting.
Practical Deployment via Flask	Deploying the model in a web application using Flask enabled real-time interaction through both image upload and webcam input. The inclusion of MediaPipe-based multi-face detection and dynamic model switching enriched user control and flexibility.
Real-World Adaptability	The system demonstrated the ability to generalize across various image conditions and perform inference in diverse scenarios. Although some misclassifications occurred, particularly among adjacent age groups, the predictions remained stable and interpretable.
Interdisciplinary Integration	This project combined techniques from deep learning, computer vision, web development, and human-computer interaction. It showcased the feasibility of translating AI models into accessible applications that non-experts can use, bridging the gap between academic research and real-world adoption.



## **6.2. Challenges Encountered**

During the development of the GAG system, several technical and operational challenges were encountered. These challenges arose throughout different stages of the project—ranging from data collection to training, evaluation, and deployment.

### **6.2.1. Data Quality and Filtering**

Despite implementing face detection via MTCNN and performing manual review, the team encountered several issues in data quality. Many crawled images were either low-resolution, poorly lit, captured from side angles, or contained multiple faces. In some cases, non-human or cartoon-like faces passed through automated filters. Cleaning such noisy data proved to be a time-consuming process and, inevitably, a small portion of these inconsistencies may have persisted in the final dataset, potentially impacting model performance.

### **6.2.2. Class Label Ambiguity**

Age group labeling—particularly distinguishing between the Teen and Adult categories—posed inherent subjectivity due to the lack of metadata in the image sources. The decision to classify based solely on facial appearance led to ambiguity in cases where individuals were visually on the boundary between age groups. This ambiguity may have introduced labeling noise, which can hinder the learning process, especially for age group classification.

### **6.2.3. Model Training Limitations**

Training deep learning models on Google Colab introduced constraints, including: Limited GPU runtime and frequent disconnections; Slow fine-tuning stages when unfreezing deeper layers; Difficulty resuming interrupted training phases without consistent checkpoints. These issues required careful planning, frequent saving of progress, and constant monitoring during long training sessions.

### **6.2.4. Multi-Face Detection Integration**

Enabling multi-face detection and classification within the same inference pipeline required additional architectural considerations. Extracting multiple faces from a single frame, processing them efficiently, and annotating each with accurate labels—without visual overlap or latency—demanded careful design and multiple rounds of optimization. Balancing performance with accuracy in both image upload and real-time webcam modes was particularly challenging.

### **6.2.5. UI–Backend Integration**

Connecting the trained model to a functioning web interface introduced typical backend engineering issues. These included ensuring quick model loading, handling user input validation (e.g., unsupported file types), and converting prediction results into browser-viewable formats. Maintaining a responsive and smooth user experience—especially during real-time webcam use—required tight coordination between model inference, image rendering, and server logic.

### **6.2.6. Performance Monitoring & Maintenance**

After deployment, it is essential to maintain and observe the model’s effectiveness over time. First, implement drift detection to continuously monitor incoming data distributions—such as age-group proportions, lighting conditions, and image angles—and key performance metrics (accuracy, confidence scores). Any deviation beyond a set threshold (e.g., a drop in accuracy of more than 5 percentage points) should automatically trigger an alert via a monitoring dashboard (e.g. Grafana with Prometheus). Next, record and analyze inference latency by measuring average processing time and latency percentiles (p50, p95, p99) on both CPU and GPU environments. If p95 latency exceeds the service-level objective (for example, 200 ms), the operations team should investigate resource allocation or apply optimizations such as TensorRT conversion or reduced-precision inference. Finally, establish an automated retraining pipeline on a weekly or monthly schedule: collect user-submitted images—especially those misclassified—append them to an incremental “retraining dataset,” and rerun the full training workflow. Only models that meet or exceed a predefined validation accuracy should be promoted to production.

### **6.2.7. Security & Privacy Considerations**

Handling user images in a live web application introduces important security and privacy challenges. To protect user privacy, ensure uploaded images reside solely in memory and are never written to disk; any temporary files stored in /tmp must be securely deleted immediately after inference, and no personally identifiable information should be logged. For API security, require all endpoints to use HTTPS and short-lived authentication tokens, enforce rate limiting (e.g. 100 requests/minute per IP), and validate incoming data to guard against denial-of-service or injection attacks. To safeguard model integrity, verify the checksum of each .keras model file against a trusted value in a secure vault before loading; if the checksum does not match, abort the load and issue an alert rather than risk using a tampered model. Finally, conduct periodic compliance audits (e.g. GDPR or local data protection regulations) to confirm that image processing, data retention, and logging practices remain transparent and legally sound.

### 6.3. Conclusion

This project successfully demonstrated the feasibility and impact of building a real-time gender and age group classification system tailored for Vietnamese demographics using deep learning. By employing a multi-output Convolutional Neural Network architecture based on ResNet50, the system achieved strong classification performance on both tasks—reaching test accuracies of 89% for gender and 86% for age group classification.

Throughout the project, we tackled challenges from dataset construction to model training and system deployment. A carefully curated dataset of over 8,000 Vietnamese facial images was developed, emphasizing diversity across age and gender while maintaining class balance. The training pipeline incorporated advanced techniques such as data augmentation, class-weighting, and fine-tuning of a pre-trained model, which collectively enhanced the model's generalization capabilities.

Importantly, the trained model was deployed in a user-friendly Flask-based web application supporting both image upload and real-time webcam detection, along with multi-face inference and model switching capabilities. This deployment bridged the gap between research and real-world application, providing an interactive and educational tool accessible to non-technical users.

Despite certain limitations—such as subjectivity in age labeling, real-time performance bottlenecks on CPU, and occasional misclassifications—the overall system performed reliably and delivered consistent, interpretable outputs. These results affirm the potential of AI-powered demographic analysis tools in domains like marketing, security, and smart interaction systems.

In conclusion, the GAG system serves as both a proof-of-concept and a deployable prototype, contributing not only to academic knowledge but also to practical solutions in the field of computer vision and human-centered AI.

#### 6.4. Recommendations for Future Work

As our multi-task gender and age classification system reaches a stable prototype stage, there remain several avenues to enhance its robustness, efficiency, and applicability. The following table summarizes six key recommendations for future work, spanning data enrichment, model innovation, system optimization, fairness assessment, and expanded functionality. Implementing these steps will not only improve overall performance and reduce bias, but also prepare the pipeline for deployment on resource-constrained devices and broader real-world scenarios.

No.	Recommendation	Description
1	Expand and Diversify the Dataset	Collect more varied images in terms of age, ethnicity, lighting, and facial angle. Use public datasets like FG-NET or UTKFace to improve label reliability and reduce bias across demographic groups.
2	Apply Advanced Data Augmentation	Integrate techniques such as MixUp, CutMix, and AutoAugment to introduce more variability and improve generalization, especially in edge cases and underrepresented classes.
3	Explore Alternative Model Architectures	Experiment with lightweight or higher-performing architectures such as MobileNetV2, EfficientNet, or Vision Transformers (ViTs) to balance accuracy and computational efficiency.
4	Optimize Real-time Performance	Convert models to TensorFlow Lite or ONNX formats and optimize inference pipelines with multi-threading or GPU/CPU acceleration to support smooth real-time deployment.
5	Extend to Multi-Attribute Prediction	Expand the system to include emotion recognition, ethnicity classification, or facial expressions through a multi-task learning setup with additional datasets.
6	Conduct Fairness & Bias Evaluation	Analyze model performance across subgroups (e.g., gender-age combinations) to detect potential biases and address them via loss weighting or dataset balancing.

**APPENDIX – Summary of Tools and Techniques Used**

Category	Name / Concept	Description / Purpose
<b>Programming Language</b>	Python	Main language used for data processing, model training, and web deployment
<b>Deep Learning Framework</b>	TensorFlow / Keras	Used to build and fine-tune the CNN model (ResNet50 + custom head)
<b>Model Architecture</b>	ResNet50 (pre-trained)	Backbone CNN used for multi-output classification via transfer learning
<b>Model Training Concepts</b>	Multi-output Learning	Predicts both gender and age group from a single image
	Transfer Learning	Reuses pre-trained weights on ImageNet to improve training speed and generalization
	Class Weighting	Adjusts training emphasis on underrepresented classes to handle slight imbalance
	Data Augmentation	Random flipping, zoom, brightness, etc., to improve generalization
	Early Stopping	Stops training when no improvement is observed in validation loss
	ReduceLROnPlateau	Reduces learning rate when plateauing is detected
<b>Image Processing</b>	OpenCV	Used for image reading, resizing, color conversion, and annotation
	NumPy	Array operations and tensor manipulation
	Matplotlib	Visualization of training curves and confusion matrices

<b>Face Detection</b>	MediaPipe (BlazeFace)	Real-time multi-face detection for both upload and webcam images
<b>Web Deployment</b>	Flask	Backend framework used to serve predictions and handle image uploads
	HTML/CSS (Bootstrap 5)	Front-end web interface for user interaction
	JavaScript	Handles real-time webcam capture and submission
<b>Evaluation Metrics</b>	Accuracy, Precision, Recall, F1	Used to assess performance of both output branches
	Confusion Matrix	Visual tool to analyze misclassification patterns
<b>Tools &amp; Platforms</b>	Google Colab / Local Jupyter	Platforms used for model training and development
	VS Code	Main development environment

Table 20: Summary of Tools and Techniques Used

## REFERENCES

1. He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep Residual Learning for Image Recognition*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778. <https://doi.org/10.1109/CVPR.2016.90>
2. Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). *Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks*. IEEE Signal Processing Letters, 23(10), 1499–1503. <https://doi.org/10.1109/LSP.2016.2603342>
3. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). *ImageNet: A large-scale hierarchical image database*. 2009 IEEE Conference on Computer Vision and Pattern Recognition, 248–255. <https://doi.org/10.1109/CVPR.2009.5206848>
4. Tan, M., & Le, Q. V. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. Proceedings of the 36th International Conference on Machine Learning (ICML), 6105–6114. <https://proceedings.mlr.press/v97/tan19a.html>
5. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv preprint arXiv:1704.04861. <https://arxiv.org/abs/1704.04861>
6. Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018). *mixup: Beyond Empirical Risk Minimization*. arXiv preprint arXiv:1710.09412. <https://arxiv.org/abs/1710.09412>
7. Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 6023–6032. <https://doi.org/10.1109/ICCV.2019.00612>
8. Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2019). *AutoAugment: Learning Augmentation Policies from Data*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 113–123. <https://doi.org/10.1109/CVPR.2019.00020>
9. UTKFace dataset. (n.d.). *UTKFace Large-scale Face Dataset*. Retrieved from <https://susanqq.github.io/UTKFace/>
10. FG-NET. (n.d.). *FG-NET Aging Database*. Retrieved from <https://yanweifu.github.io/fgnet-aging-database.html>
11. TensorFlow. (n.d.). *Keras Model Saving and Serialization*. Retrieved June 2025, from [https://www.tensorflow.org/guide/keras/save\\_and\\_serialize](https://www.tensorflow.org/guide/keras/save_and_serialize)
12. TensorFlow. (n.d.). *Callbacks API Guide*. Retrieved June 2025, from [https://www.tensorflow.org/api\\_docs/python/tf/keras/callbacks](https://www.tensorflow.org/api_docs/python/tf/keras/callbacks)
13. Chollet, F. (2017). *Deep learning with Python*. Manning Publications.