# Machine Learning - Final Project

Michael Dadush 206917908
Shay Gali 315202242

March 2025

## 1 Introduction

Gender and age classification in not controlled settings shows big challenges due to many imaging conditions and problems in the data. In our project, we deal with these issues by making a framework that includes data exploration, preprocessing, feature extraction, model training, and checking performance. All our code and implementation details can be seen on our github repository [1].

We start our project with deep data exploration to understand main characteristics and challenges of Adience Benchmark dataset. This first analysis is in the `data_exploration.ipynb` notebook, which shows data distribution, imaging conditions, and what preprocessing we need. You can see notebook and follow our exploration by visiting data_exploration.ipynb.

This introduction part prepare for next detailed discussions about challenges and the strategies we use to overcome them, making sure we have good approach to solve the gender and age classification problem.

After data exploration, we try several models we learn in our course, like Softmax regression, Support Vector Machines (SVM), and other machine learning methods. To make feature representation better, we try different image transformations like grayscale and RGB. Also, we use Principal Component Analysis (PCA) for dimensionality reduction, so we can check if compact representations improve classification performance. The classical models we implement are in the `old_models` directory on GitHub: old_models folder.

After we check individual models, we want to combine our findings by using multiple models trained on different image representations. We develop two pipelines: one for grayscale images and another for RGB images. These approaches are in separate notebooks:

- **Grayscale Models:** We train various models on grayscale images and check their performance. The complete implementation is in grayscale_models.ipynb.
- **RGB Models:** We do similar experiments on RGB images, comparing if color information give better classification accuracy. This notebook is at rgb_models.ipynb.

With models trained and checked, we answer key research questions about gender and age classification. Our analysis and findings for first question are in q1.ipynb.

To show practical use of our model, we create real-time camera application that predicts gender and age from live video. This application, in the `camera-viewer-app` folder, shows how model works in real situations. The source code for this application is at camera-viewer-app.

# 2 Description of the Database

We chose to work with the Adience Benchmark dataset for Gender and Age Classification[3]. This dataset has real-world images of faces, which makes it good for testing different machine learning methods. The dataset includes:

### Labels in Dataset

The dataset provides many labels for each data point, we used the followinglabels for our analysis:

- **gender**: Male / Female
- **age**: The dataset categorizes images into the following age ranges: 0–2 years, 4–6 years, 8–12 years, 15–20 years, 25–32 years, 38–43 years, 48–53 years, 60 and above
- **user_id, face_id, original_image**: These labels are used to identify the image and the person in the image.
- The dataset includes more labels that we did not use in our analysis, but we explain what each label means in the dataset in the data exploration notebook[1].

# 3 Analysis of the Database

## 3.1 Data Exploration and Preprocessing

When we explore the Adience Benchmark dataset, we find several important characteristics and preprocessing needs:

1. **Duplicate Directories**: At first look, we find two directories with name "faces" that contain image files. After we check more, we see that one directory (with 40 files) is just small part of the other one (with 168 files). We choose the big directory with all 168 user folders for our work.
2. **Data Organization**: The dataset is organize in 5-fold cross-validation way, with data in files named "fold_0_data.txt" to "fold_4_data.txt". We use Folds 0-3 (with 15,554 entries) for training, and fold 4 (with 3,816

entries) for testing, so totally we have 19,370 entries before we do preprocessing.

3. **Missing Values**: The original dataset have some entries with missing values. After we remove these entries, we get 18,551 complete records, which mean only small part (about 4.2%) of data is incomplete.

## 3.2 Age Classification Standardization

One of big challenges when we work with Adience dataset was the not consistent representation of age information:

1. **Overlapping Age Ranges**: Dataset had many overlapping age ranges that we need to fix:
   - The ranges (38, 42), (38, 43), and (38, 48) we merge into (38, 43) based on frequency analysis (46, 2293, and 6 instances)
   - One weird (8, 23) age range with only one instance we remove
   - The less common (27, 32) range (77 instances) we merge into more frequent (25, 32) range (4,953 instances)
2. **Single Age Values**: Dataset had 1,094 entries with single age values not ranges. We map these to right age ranges based on how close they are:
   - Ages 2, 3 $\rightarrow$ (0, 2), (4, 6)
   - Age 13 $\rightarrow$ (8, 12)
   - Ages 22, 23, 29, 34 $\rightarrow$ (25, 32)
   - Ages 35, 36, 42 $\rightarrow$ (38, 43)
   - Ages 45, 46, 55 $\rightarrow$ (48, 53)
   - Ages 57, 58 $\rightarrow$ (60, 100)

   This standardization give us consistent set of eight age ranges: (0, 2), (4, 6), (8, 12), (15, 20), (25, 32), (38, 43), (48, 53), and (60, 100).

## 3.3 Gender Distribution Analysis

- Female: 9,332 samples (50.3%)
- Male: 9,216 samples (49.7%)

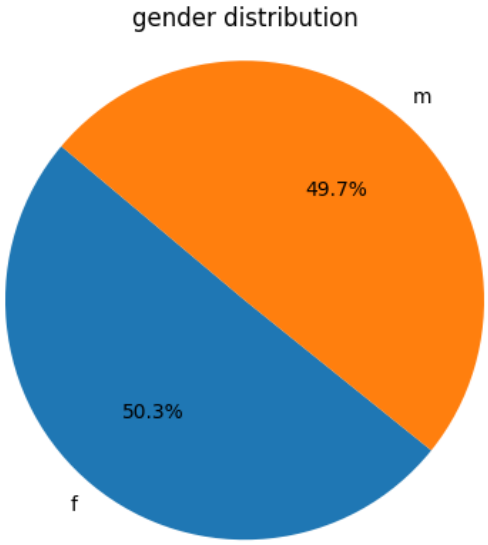



Figure 1: Gender Distribution in the Adience Benchmark Dataset

### 3.4 Age Distribution Analysis

After standardization, the age distribution across the eight defined ranges was:

- (0, 2): 2,491 samples (13.4%)
- (4, 6): 2,158 samples (11.6%)
- (8, 12): 2,287 samples (12.3%)
- (15, 20): 1,642 samples (8.9%)
- (25, 32): 5,391 samples (29.1%)
- (38, 43): 2,695 samples (14.5%)
- (48, 53): 990 samples (5.3%)
- (60, 100): 896 samples (4.8%)

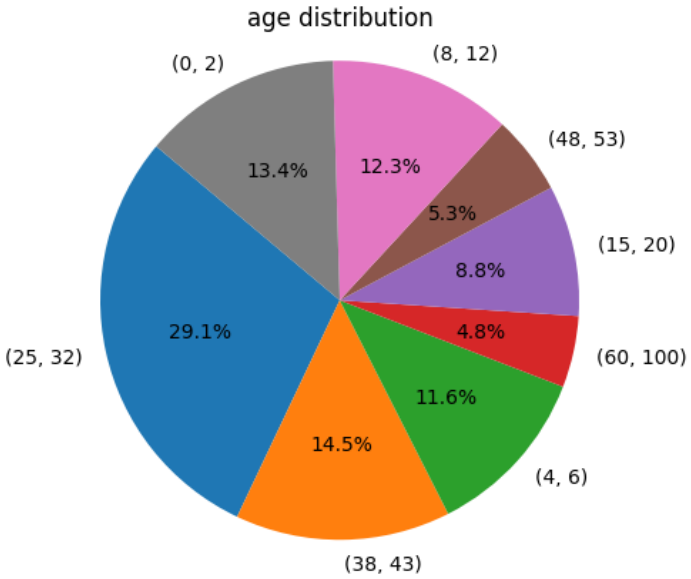



Figure 2: Age Distribution in the Adience Benchmark Dataset

### 3.5 Combined Age-Gender Analysis

The intersection of age and gender categories revealed interesting patterns:

- The (25, 32) age range showed the highest representation for both genders, with 3,026 female and 2,363 male samples.
- Gender distribution within age groups varied significantly:
  - (0, 2): 684 female vs. 1,807 male
  - (8, 12): 1,353 female vs. 932 male
  - (25, 32): 3,026 female vs. 2,363 male
  - (48, 53): 458 female vs. 532 male
  - (60, 100): 428 female vs. 467 male

This combined analysis provides valuable insights into potential intersectional biases that might emerge during model training, particularly in the youngest age group where gender assignment was artificial rather than determined from image features.

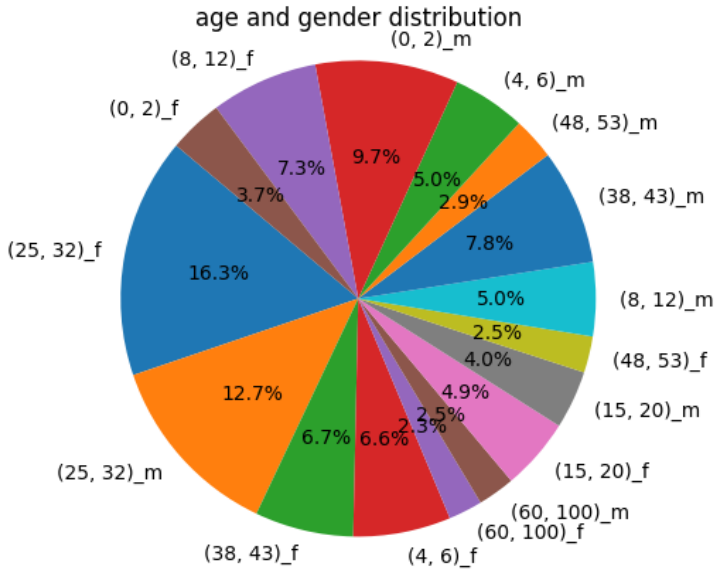



Figure 3: Combined Age and Gender Distribution in the Adience Benchmark Dataset

### 3.6 Data Partitioning and Integrity

To make sure we have good evaluation way, we split the dataset into three parts:

- Training set: 11,850 samples (63.9%)
- Validation set: 2,963 samples (16.0%)
- Test set: 3,729 samples (20.1%)

We check very carefully that no overlap exist between these sets by checking if any image identifiers and face IDs appear in more than one set. This check confirm complete separation, which is very important for fair performance evaluation.

## Tools used in the project

The following tools and techniques were used in this project :

- Softmax
- SVM
- Random Forest
- Adaboost
- KNN
- Multi-layer Perceptron (MLP)
- ResNet50- as feature extractor and as a model
- PCA - as a feature extractor

# 4 Training Methodology

## 4.1 Feature Extraction

For most of our models, we used pre-trained networks to extract meaningful features from face images rather than training directly on raw pixel values.

- **ResNet50 Feature Extraction**: We utilized the ResNet50 architecture pre-trained on ImageNet to transform raw images into 2048-dimensional feature vectors. This was implemented for both RGB and grayscale processing pipelines.
- **Image Preprocessing**: Before feature extraction, images were resized to 224×224 pixels and normalized according to ImageNet statistics. For grayscale processing, single-channel images were duplicated across three channels to match ResNet50's input requirements.
- **Batch Processing**: To manage memory efficiently, images were processed in batches of 64, with separate feature sets created for training, validation, and test partitions.

We also try PCA as a feature extraction method, but the results were not as good as the ResNet50 feature extraction method, so we decided to use the ResNet50 feature extraction method for our models.

## 4.2 Data Preprocessing

After feature extraction, we did several preprocessing steps to prepare the data for model training:

- **Feature Standardization**: We standardize features to have zero mean and unit variance using scikit-learn's StandardScaler, which we fit on training set and apply to validation and test sets.
- **Label Encoding**: We encode the combined age and gender labels (e.g., "(25, 32)_f" for female in 25-32 age range) into numeric indices using LabelEncoder.
- **Class Weights**: To fix class imbalance in dataset, we compute balanced class weights that are inverse to class frequencies, so minority classes get right attention during training.

## 4.3 Model Training Approaches

We train two parallel sets of models—one with RGB features and another with grayscale features—to compare which is better:

- **Traditional Machine Learning**: For Softmax (Logistic Regression), SVM, Random Forest, AdaBoost, and KNN models, we use scikit-learn implementations.

- **Neural Network Models**: For MLP model, we implement architectures in TensorFlow with dropout and batch normalization to stop overfitting. Training use Adam optimizer with categorical cross-entropy loss and early stopping based on validation set performance.
- **CNN Training**: These models we train directly on image data instead of extracted features.

### 4.4 Evaluation Methodology

To ensure robust evaluation of model performance:

- **Metrics**: We primarily used accuracy as our evaluation metric, calculated on both training and test sets to assess overfitting.
- **Confusion Matrix Analysis**: We analyzed confusion matrices to understand the types of errors made by each model, particularly focusing on whether errors occurred between adjacent age groups or across gender boundaries.
- **Cross-Comparison**: Performance was compared across RGB and grayscale processing pipelines to determine whether color information provided meaningful advantages.

All models were saved after training for later deployment and testing in real-world scenarios, which informed our analysis of practical applications detailed in the research questions section.

# 5 Model Performance

## 5.1 Grayscale scores

### 5.1.1 Grayscale Model Training and Testing Accuracy

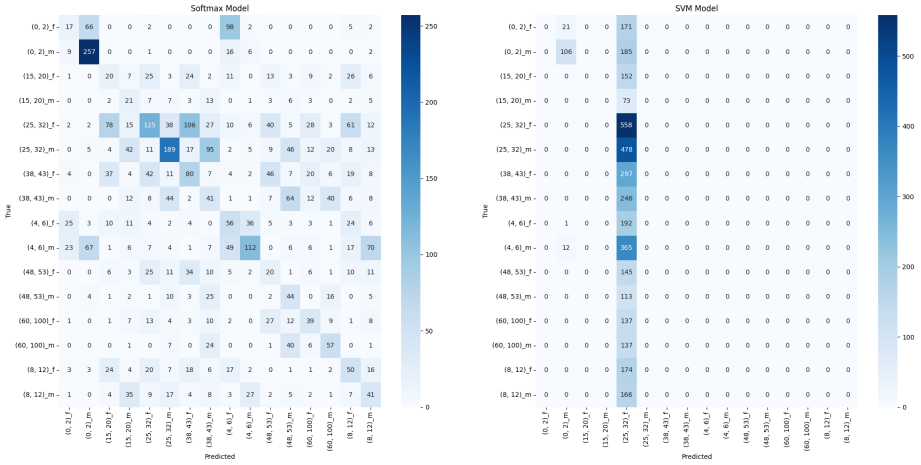
Table 1 summarizes the accuracy results of different models on the dataset using grayscale image.

| Model | Train Accuracy | Test Accuracy | Notes |
|---|---|---|---|
| Softmax | 0.7095 | 0.3135 | most of the error in the confusion matrix was in the adjacent age group. |
| SVM | 0.2046 | 0.1781 | the model predict for the most of the data to 25-32 age group, and the gender was female. |
| Random Forest | 0.7730 | 0.3167 | the model predict for the most of the data to 25-32 age group, but he got the gender right. |
| AdaBoost | 0.3467 | 0.2650 | the have berrly predict classes that he dont have a lot of data. |
| K-NN (k=1) | 0.9998 | 0.2202 | we can see the overfitting - the train accuracy is very high, but the test accuracy is very low. |
| K-NN (k=3) | 0.7525 | 0.2258 | we can see start of overfitting - most of the error in the confusion matrix was near the (25,32) age group - the majority class. |
| K-NN (k=5) | 0.7001 | 0.2314 | the over fitting is less significant |
| K-NN (k=7) | 0.6631 | 0.2422 | the over fitting is less significant |
| K-NN (k=9) | 0.6331 | 0.2440 | the over fitting is less significant |
| MLP | 0.7521 | 0.3229 | the model with the less overfitting, and when the model got wrong predictions, it was not by much (the most of the error in the confusion matrix was in the adjacent age group). |
| CNN | 0.4371 | 0.3153 | this model have overffiting - it predict most of the data to the (25,32) age group. |

Table 1: Train and Test Accuracy of Various Models with grayscale method

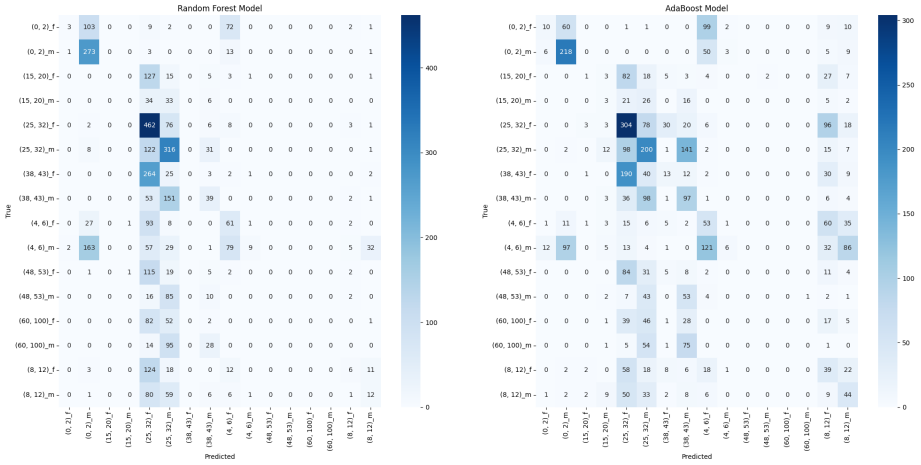
### 5.1.2 Grayscale Model Confusion Matrices

Here are confusion matrices for grayscale models, they show where model predictions are right (on diagonal) and where they are wrong (not on diagonal).



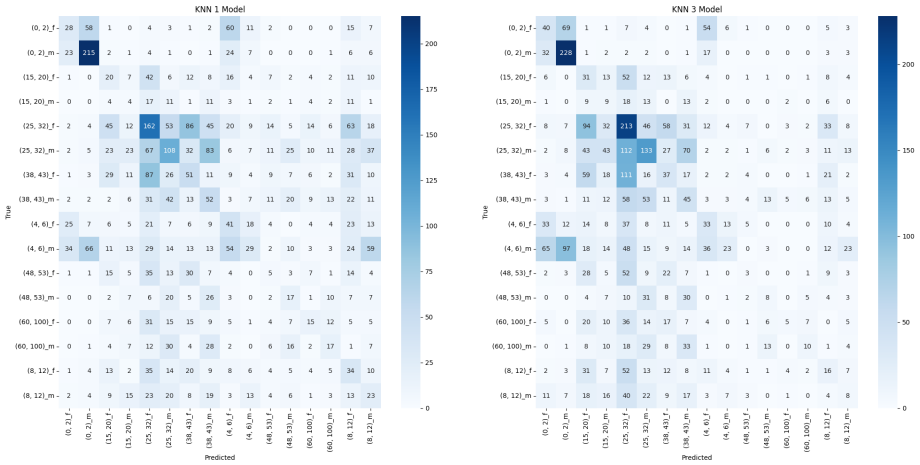

(a) Softmax (b) SVM

Figure 4: Confusion matrices for Softmax and SVM models (grayscale)



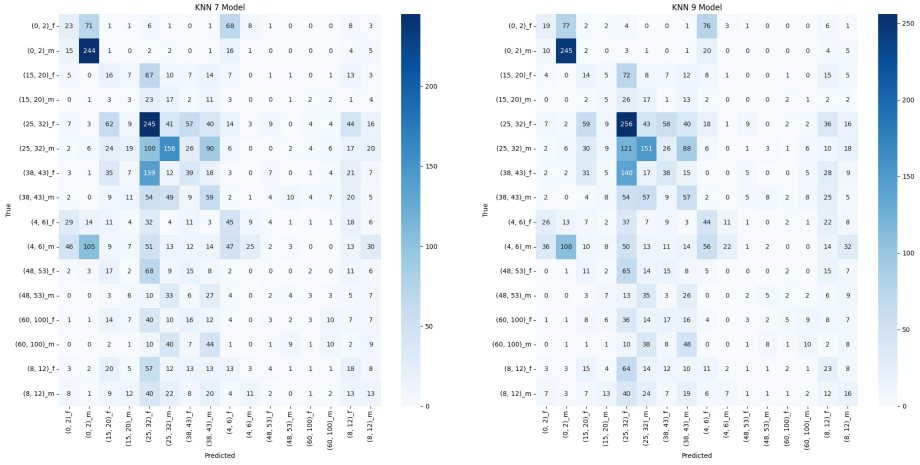

(a) Random Forest (b) AdaBoost

Figure 5: Confusion matrices for Random Forest and AdaBoost models (grayscale)



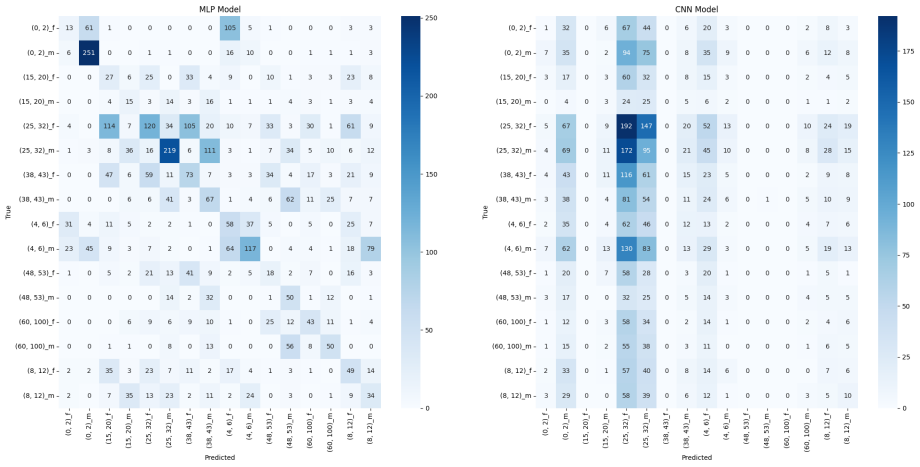

(a) KNN (k=1) (b) KNN (k=3)

Figure 6: Confusion matrices for KNN models with k=1 and k=3 (grayscale)



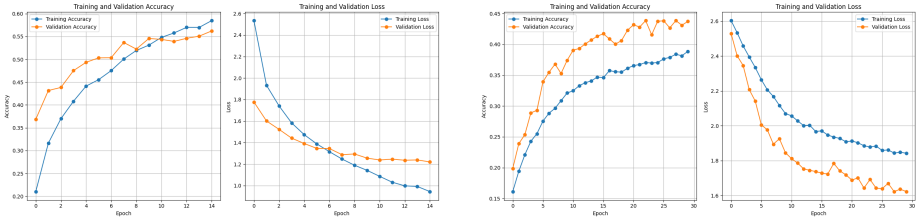

(a) KNN (k=7) (b) KNN (k=9)

Figure 7: Confusion matrices for KNN models with k=7 and k=9 (grayscale)




(a) MLP (b) CNN

Figure 8: Confusion matrices for MLP and CNN models (grayscale)




(a) MLP training history (b) CNN training history

Figure 9: Taining history for MLP and CNN models (grayscale)

## 5.2 RGB scores

### 5.2.1 RGB Model Training and Testing Accuracy

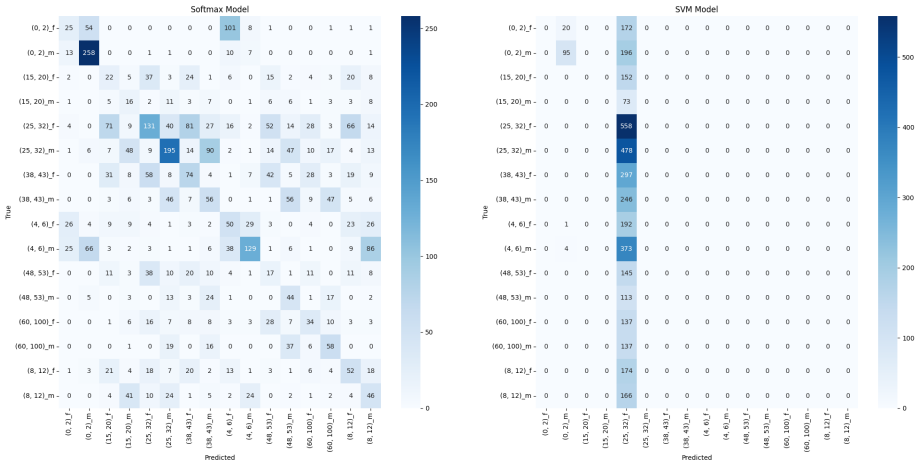
Table 2 summarizes the accuracy results of different models on the dataset using RGB image. (we dont add a Note column because the results are similar to the grayscale image)

| Model | Train Accuracy | Test Accuracy |
|---|---|---|
| Softmax | 0.7331 | 0.3237 |
| SVM | 0.2045 | 0.1751 |
| Random Forest | 0.7728 | 0.3218 |
| AdaBoost | 0.3716 | 0.3162 |
| K-NN (k=1) | 0.9998 | 0.1917 |
| K-NN (k=3) | 0.7778 | 0.2092 |
| K-NN (k=7) | 0.6886 | 0.2279 |
| K-NN (k=9) | 0.6658 | 0.2333 |
| MLP | 0.7950 | 0.3478 |
| CNN | 0.4718 | 0.3298 |

Table 2: Train and Test Accuracy of Various Models with RGB image

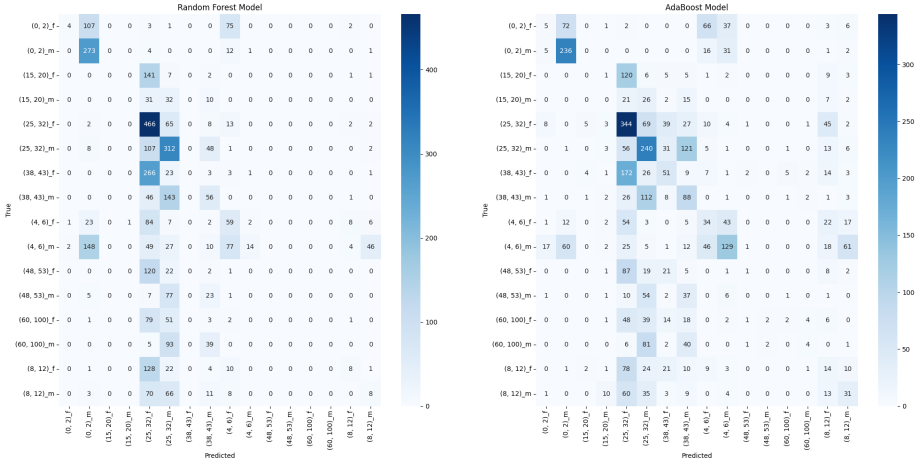
### 5.2.2 RGB Model Confusion Matrices

Here are confusion matrices for rgb models, they show where model predictions are right (on diagonal) and where they are wrong (not on diagonal).



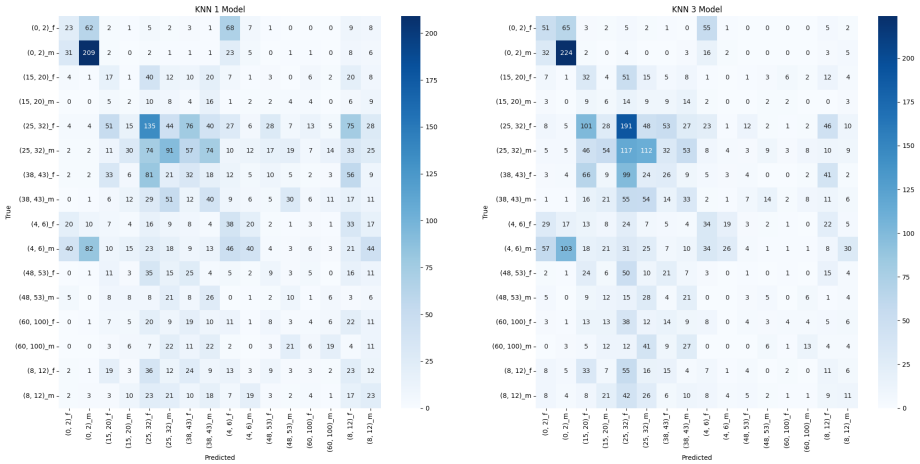

(a) Softmax (b) SVM

Figure 10: Confusion matrices for Softmax and SVM models (rgb)



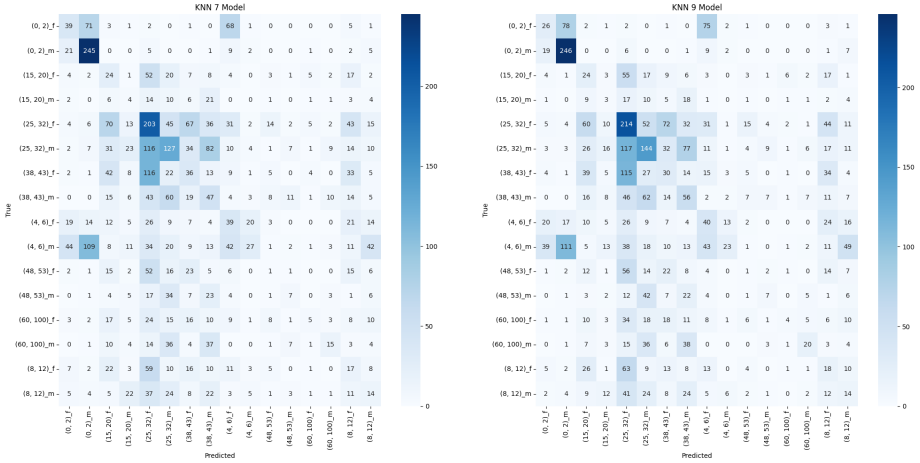

(a) Random Forest (b) AdaBoost

Figure 11: Confusion matrices for Random Forest and AdaBoost models (rgb)




(a) KNN (k=1) (b) KNN (k=3)

Figure 12: Confusion matrices for KNN models with k=1 and k=3 (rgb)

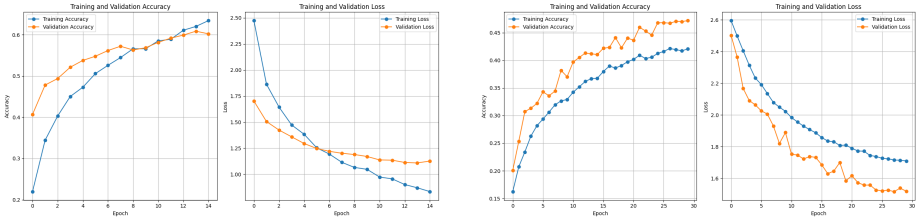



(a) KNN (k=7) (b) KNN (k=9)

Figure 13: Confusion matrices for KNN models with k=7 and k=9 (rgb)


(a) MLP (b) CNN

Figure 14: Confusion matrices for MLP and CNN models (rgb)




(a) MLP training history (b) CNN training history

Figure 15: Taining history for MLP and CNN models (rgb)

## 5.3 Analysis of Results

Our results show important patterns across different models and processing methods. Here we analyze what we found.

### 5.3.1 General Performance Trends

We see that models trained on RGB features perform little better than grayscale ones, which mean color information helps for age and gender classification. The best model was MLP with RGB features (34.78% test accuracy), and worst was SVM (17.51% test accuracy on RGB).

The test accuracies (17.51% to 34.78%) are not very high because this is hard task with 16 classes (age+gender). Random guessing would give about 6.25% accuracy, so our models are 3-5 times better than random.

### 5.3.2 Model-Specific Analysis

**Softmax Regression:** Softmax got good performance (32.37% test accuracy with RGB) even though it is simple model. This mean our feature space have some linear separability. The confusion matrix show it was best on big classes like (0, 2)_m and (25, 32)_m, with most errors between close age groups not gender mistakes.

**Support Vector Machine (SVM):** SVM perform badly (17.51% test accuracy), which suggest the high-dimensional feature space was not good for finding hyperplanes. SVM mostly predict the (25, 32)_f class—our biggest class—showing it bias toward majority classes even with class weighting.

**Random Forest:** Random Forest perform well (32.18% test accuracy with RGB), showing it can catch complex patterns with many decision trees. The model learn feature importance patterns from ResNet50 embeddings, and depth-limited trees (max_depth=10) help control overfitting. But like other models, it have trouble with small classes.

**AdaBoost:** AdaBoost show medium performance (31.62% with RGB), focusing on hard-to-classify examples. This work well for our imbalanced dataset, and we see smaller gap between train and test accuracy compared to Random Forest.

**K-Nearest Neighbors (KNN):** KNN show clear overfitting, especially with small k. With k=1, model got almost perfect training accuracy (99.98%) but bad test performance (19.17

**Multi-Layer Perceptron (MLP):** MLP got best performance (34.78% test accuracy with RGB), showing deep learning better for complex patterns. The dense layers with dropout (0.5) and batch normalization control overfitting while keeping good representation. Early stopping also help generalization by stopping training at right time.

**Convolutional Neural Network (CNN):** CNN perform well (32.98% with RGB) even when trained on raw images not extracted features. This show end-to-end learning can be good as transfer learning for this task. The smaller gap between training (47.18%) and testing (32.98%) accuracy suggest CNN architecture have some natural regularization compared to other models.

#### 5.3.3 Overfitting Analysis

All models had some overfitting, we see in gap between training and test accuracy. This happen because:

- **Limited data diversity:** Even with thousands of images, we don't have enough samples per class for good generalization across all 16 classes.
- **Class imbalance:** The uneven distribution of samples across age and gender categories make it hard for models to learn good representations of small classes.
- **High-dimensional feature space:** The 2048-dimensional ResNet50 features create very big feature space compared to training examples, which make overfitting more likely.

#### 5.3.4 RGB vs. Grayscale Performance

RGB models always perform better than grayscale, but not by much (usually 1-3 percentage points). This mean color information is helpful but not essential for age-gender classification. Color information maybe help with things like skin tone changes with age or makeup differences between genders.

#### 5.3.5 Error Pattern Analysis

Confusion matrices show consistent error patterns in all models:

- Most wrong predictions happen between close age groups (like (25, 32) and (38, 43)), which show it hard to put ages in exact groups based only on face features.
- Gender mistakes happen less than age group mistakes, which mean gender features are more clear than age features in face images.
- Some age-gender combinations, especially ones with few training examples, always get poor classification results in all models.

This analysis show that while our models give useful age-gender classification, there is still lot of room for improvement with better architectures, data augmentation, and training strategies for this multi-class problem.

# 6 Research Questions

## Question 1: Different Types of People

- Does it work the same for people from different places?
- Can we use the same model everywhere or do we need different ones?

## Answer to Question 1

To explore these questions, we used the UTKFace dataset [2], which contains images of people from diverse ethnic backgrounds with labeled age, gender, and ethnicity information. Our analysis aimed to determine whether the model performs consistently between different groups.

1. **Image Path Construction**: Image paths were constructed to facilitate loading - the dataset was diffrent, and the way that we ajust the data to work with our model can see in q1.ipynb.
2. **Label Encoding**: Age and gender were combined and encoded for model training.

A pie chart of race distribution illustrates the diversity within the dataset. Age and gender distributions were plotted for each racial group to provide context for analysis.

Multiple models, including CNNs and various machine learning models, were loaded for evaluation:

- **CNN Models**: Trained on RGB images and grayscale images.
- **Sklearn Models**: Includes classifiers like AdaBoost, KNN, Random Forest, Softmax, and SVM.

Preprocessing of both RGB and grayscale images was performed. Features were stored in `.npy` files to prevent redundant processing.

Models were evaluated across different racial groups, with results summarized as follows:

**Model Accuracy by Race**

| Model | White | Black | Asian | Indian | Other |
|---|---|---|---|---|---|
| CNN | 0.17 | 0.11 | 0.17 | 0.14 | 0.22 |
| AdaBoost | 0.18 | 0.13 | 0.17 | 0.15 | 0.20 |
| KNN | 0.17 | 0.15 | 0.19 | 0.14 | 0.17 |
| Random Forest | 0.19 | 0.17 | 0.20 | 0.17 | 0.23 |
| Softmax | 0.23 | 0.18 | 0.25 | 0.22 | 0.23 |
| SVM | 0.07 | 0.07 | 0.07 | 0.06 | 0.08 |

Table 3: Model accuracy across different racial groups.

The accuracy rates suggest variability in model performance across different races, which could indicate biases in model training or dataset representation. Models tended to perform better on the "Other" category, suggesting that there is a more even distribution of features in this group, or it could be less represented in some training aspects. But we can see that the prediction on "Black" was the worst, which could indicate that the model was not trained well on this group.

## Question 2: Can We Use This in Real Life?

- Will it work fast enough for apps or security cameras?
- What about when there is bad lighting or people wear glasses?

## Answer Question 2

To evaluate real-time performance, we built a face recognition system that detects age and gender when people appear in front of the camera. Our results show that most models perform well in real-time scenarios, even with poor lighting or glasses, making it suitable for applications such as mobile applications and security cameras.
Just one model - the SVM - showed a significant decrease in performance in real-time scenarios, indicating that it may not be suitable for such applications.

## Question 3: About Age and gender prediction

- Which model is best at dealing with different age and gender?

## Answer to Question 3

From the analysis of the models, we can see that the model with the less overfitting, and when the model got wrong predictions, it was not by much (the most of the error in the confusion matrix was in the adjacent age group), it was the MLP model.

## Question 4: How Models Work with Different Images

- What happens when images are not clear or blurry?
- Do facial expressions (such as smiling or not smiling) affect the results?

## Answer Question 4

To investigate the impact of facial expressions on the predictions of the model, we examined four different emotional states: smiling, serious, mad and regular. Our analysis revealed that all models produced consistent predictions in these

expressions, indicating that facial expressions do not significantly influence the results.
Also from the real-time analysis, we can see that the model works well with different images, even with blurry images, which makes it suitable for real-world applications.

# 7 Challenges in Gender and Age Classification using Adience Benchmark Dataset

## 7.1 Data Structure and Organization Challenges

When we explore the Adience Benchmark dataset, we find several structure problems:

- **Duplicate folders:** The dataset have two `faces` folders with same content, so we need to check which folder to use without losing important data.
- **Files in many places:** Files were spread in many different fold directories, so we need special code to combine all dataset together.
- **Keeping people separate:** We need to make sure images from same person don't appear in both training and testing sets by checking the `user_id` across different folds.

## 7.2 Data Quality and Preprocessing Challenges

The dataset have many quality and preprocessing problems we need to fix:

- **Age labels not consistent:** Some labels have specific ages (like '22', '35', '58') and some have age ranges (like '(0, 2)', '(25, 32)'), so we need process to put single ages into right ranges.
- **Age ranges that overlap:** We find many overlapping age groups:
  - Same range with different numbers: (38, 42), (38, 43), and (38, 48)
  - Similar ranges with different start/end: (25, 32) and (27, 32)
  - Ranges that don't make sense together: (8, 12), (8, 23), and (15, 20)
- **Missing gender labels:** Many images (especially in (0, 2) age range) have 'u' for gender, so we need to remove them or guess gender.
- **Missing data:** Some records have NaN values that we need to handle by filling in values or removing them.

## 7.3 Class Imbalance Challenges

When we check the dataset, we find big problems with balance:

- **Age groups not equal:** The (25, 32) age range had too many samples (5391), but other ranges like (60, 100) had very few (just 896).
- **Gender not balanced:** In some age groups, one gender has much more samples than other gender.
- **Unknown gender problem:** Most images with 'u' gender label were in (0, 2) age range, this make bias in the data.

## 7.4 Feature Engineering and Extraction Challenges

When we try to convert images to features, we face many problems:

- **ResNet50 is too big:** Using ResNet50 need a lot of computer power and we need to optimize it for our task.
- **Features too many or too few:** It's hard to decide how many features from ResNet50 is good - too many make slow model, too few lose information.

## 7.5 Early Stopping Problems

When we try to use early stopping to stop overfitting, we face many challenges:

- **Which metric to check:** Hard to choose right metrics to watch:
  - Should we watch both age and gender accuracy or just one?
  - Sometimes loss goes down but accuracy not improve
  - Not sure if we need special metrics for imbalanced classes
- **Patience value problem:** Hard to choose how many epochs to wait before stopping:
  - Too little waiting: Might stop too early before model learn good weights
  - Too much waiting: Waste time and might overfit
- **Learning rate changes:** Hard to use with learning rate scheduler:
  - Not sure if model stuck or just need learning rate change
  - Should we reset patience after change learning rate?
- **Saving models:** Need to decide about checkpoints:
  - Need lot of disk space to save models during training
  - Not sure if save model with best validation loss or best accuracy
  - How to load best model when training finish
- **Planning compute time:** Hard to know how long training will take:
  - Different models stop at different times
  - Need balance between trying many parameters and saving time

## 7.6 Combined Age-Gender Prediction Problems

Predicting both age and gender together was hard:

- **How to setup problem:** We need to decide:
  - Make two separate models (one for age, one for gender)
  - Make one model that predicts combined age-gender classes
  - Make one model with shared features but separate outputs
  - From our observation, the method that work the best for us was to make one model that predicts in combined lable.
- **Loss function problems:** Hard to balance how much age errors and gender errors matter.
- **Evaluating results:** Need special way to measure if model good at both age and gender.

## 7.7 Training and Testing Split Problems

Making good train and test splits had many problems:

- **How to split data:** Dataset already had folds, but we need validation set too.
- **No data leaking:** Need make sure same person not in both train and test set.
- **Keeping fold info:** Need track which images come from which original folds.

# 8 Real time application

We make camera application that can detect faces and predict age and gender in real time. We use our best model - MLP with RGB features - for this app. We build this app with OpenCV for finding faces, TensorFlow for running our model, and Tkinter for making simple window interface.

## 8.1 How the app works

- The app open window that show live video from camera.
- When app see face in video, it predict age and gender. We make prediction every 3 frames to save computer power.
- App show age range and gender on top of each face, also show how confident model is about prediction.

## 8.2 Image of the app

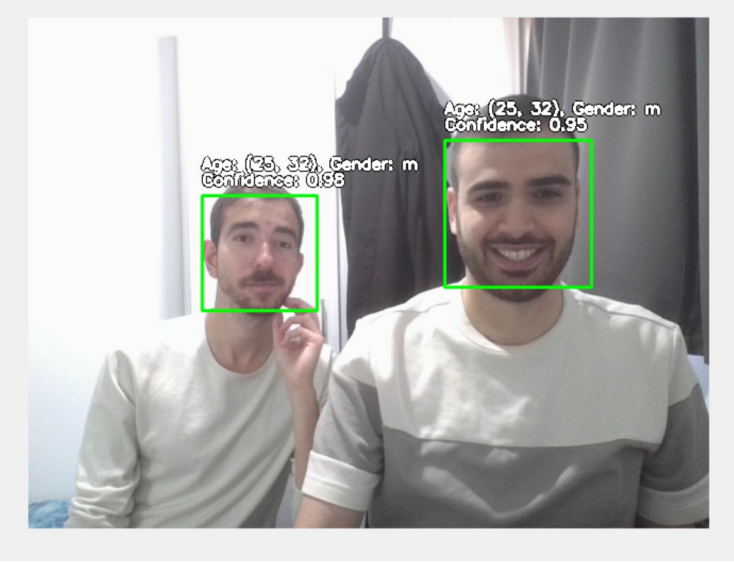



Figure 16: Real-time Age and Gender Prediction App

# References

[1] Michael Dadush Shay Gali. *GitHub Repository*. GitHub. Available at: `https://github.com/ShayGali/VisAge`. 2025.

[2] jangedoo. *UTKFace Dataset*. Kaggle. Available at: `https://www.kaggle.com/jangedoo/utkface-new`. 2024.

[3] Tung Le. *Adience Benchmark Gender and Age Classification Dataset*. Kaggle. Available at: `https://www.kaggle.com/datasets/ttungl/adience-benchmark-gender-and-age-classification`. 2024.