Gender, Age, and Ethnicity Classification Using Multi-Task Learning

Utpalraj Kemprai

December 9, 2024

Abstract

This project presents a multitask deep learning approach for predicting gender, age group, and ethnicity from facial images. Utilizing the FairFace dataset, we trained a Convolutional Neural Network (CNN) capable of performing these classifications simultaneously. The pipeline begins with face detection and cropping using Haar cascades, followed by a lightweight multitask learning architecture. To optimize computational efficiency without compromising accuracy, we incorporated a Depthwise Separable CNN into the architecture, reducing the number of parameters and enhancing inference speed. Additionally, the model was rigorously evaluated to check for potential biases toward any specific gender, age group, or ethnicity, ensuring fairness and inclusivity in predictions. The model is deployed as a user-friendly application through Streamlit, showcasing its potential for real-world applications in security, marketing, and social research.

Contents

List of Figures

List of Tables

1. Introduction

The increasing demand for automated systems in human profiling has driven significant advancements in computer vision and deep learning. Predicting demographic attributes such as gender, age, and ethnicity from facial images has diverse applications in fields like marketing, healthcare, and security. This project focuses on developing a multi-task learning framework using convolutional neural networks (CNNs) to achieve accurate predictions for these attributes.

Multitask learning offers distinct advantages over traditional single-task models by leveraging shared features across tasks, which not only enhances prediction accuracy but also reduces computational overhead and improves generalizability.

The scope of this project encompasses face detection, demographic prediction, and deployment of the trained model using the FairFace dataset. Broader tasks, such as emotion recognition or face recognition, lie beyond the defined boundaries of this work.

2. Literature Review

2.1. Metrics for Evaluating Data Imbalance

The challenge of data imbalance has been extensively studied in the field of machine learning, as it can significantly impact model performance, particularly for classification tasks. To address this issue, various metrics have been proposed to quantitatively assess the degree of imbalance in datasets. This section reviews key metrics commonly used in literature:

2.1.1. Max Ratio and Min Ratio

These metrics quantify the imbalance by comparing the maximum and minimum proportions of class frequencies. The **max ratio** is the proportion of the largest class relative to the total dataset size, while the **min ratio** measures the smallest class's proportion. Together, they highlight the spread of class distributions and are simple yet effective for identifying extreme imbalance scenarios.

Let the dataset have C classes, and let N_c be the number of instances in class c. The total number of instances is denoted as $N = \sum_{c=1}^{C} N_c$. Then the **max ratio** and **min ratio** can be defined as:

$$\text{Max Ratio} = \frac{\max(N_c)}{N}$$

$$\text{Min Ratio} = \frac{\min(N_c)}{N}$$

2.1.2. Ratio Range

The ratio range, computed as the difference between the max and min ratios, provides a straightforward measure of the variability in class proportions. A higher ratio range indicates greater disparity among classes, which could lead to biased model predictions.

$$\text{Ratio Range} = \text{Max Ratio} - \text{Min Ratio}$$

Reference: He and Garcia (2009) explored this measure in their survey of learning from imbalanced data.

2.1.3. Coefficient of Variation (CV)

The **coefficient of variation** (CV) [15] measures the relative variability of class frequencies, calculated as the standard deviation normalized by the mean class frequency.

It captures the extent of imbalance relative to the average class size, with higher values indicating more pronounced imbalance.

Let the frequencies of the C classes be $N_1, N_2, \ldots, N_C$. The mean class frequency μ and the standard deviation σ are defined as:

$$\mu = \frac{1}{C} \sum_{c=1}^{C} N_c$$

$$\sigma = \sqrt{\frac{1}{C} \sum_{c=1}^{C} (N_c - \mu)^2}$$

The **coefficient of variation** is then:

$$\text{CV} = \frac{\sigma}{\mu}$$

2.1.4. Entropy

Entropy [16] quantifies the uncertainty or diversity in class distributions. A perfectly balanced dataset has maximum entropy, while a highly imbalanced dataset has lower entropy. Entropy is particularly useful for multi-class problems, where it provides an aggregated view of imbalance.

Entropy H for a multi-class dataset is defined as:

$$H = -\sum_{c=1}^{C} p_c \log(p_c)$$

where $p_c = \frac{N_c}{N}$ is the proportion of class c in the dataset.

Reference: Shannon's entropy theory (1948) serves as the foundational work for this metric, applied extensively in imbalance analysis by Chawla et al. (2004).

2.1.5. Gini Index

The **Gini index** [17], commonly used in economics to measure inequality, has been adapted to evaluate imbalance in machine learning. It ranges from 0 (perfect balance) to 1 (complete imbalance), capturing the inequality in class frequencies. The Gini index G is defined as:

$$G = 1 - \sum_{c=1}^{C} p_c^2$$

where $p_c = \frac{N_c}{N}$ is the proportion of class c in the dataset.

2.2. Face Analysis Techniques

Face analysis has been a significant research area in computer vision, evolving from traditional methods to state-of-the-art deep learning-based approaches. Early techniques

relied heavily on handcrafted features, such as Haar cascades [9] and Local Binary Patterns (LBP) [10] , which were effective for simple tasks like face detection and basic feature extraction. Haar cascades, introduced by Viola and Jones (2001), revolutionized real-time face detection by using an integral image representation and a cascaded classifier structure. LBP further contributed to texture-based face representation by encoding local image patterns.

With the advent of deep learning, Convolutional Neural Networks (CNNs) have emerged as the standard for face analysis tasks due to their ability to automatically learn hierarchical feature representations from raw pixel data. CNN-based architectures like AlexNet, VGG, and ResNet have demonstrated superior performance in tasks such as face detection, recognition, and attribute classification. Advanced face analysis frameworks now integrate facial landmarks for alignment and preprocessing to improve accuracy. Techniques like the Single Shot Multibox Detector (SSD) and Multi-Task Cascaded Convolutional Networks (MTCNN) have also been adopted for robust face detection under varying conditions of lighting, pose, and occlusion.

Recent trends emphasize fairness and explainability in face analysis, as traditional models have often shown bias against underrepresented demographic groups. The shift toward datasets and methods that account for demographic diversity has become crucial in reducing these disparities.

2.3. Multitask Learning

Multitask learning (MTL) is a paradigm in which a model is trained to optimize multiple objectives simultaneously. By sharing a common feature representation across tasks, MTL improves generalization and reduces computational overhead. This approach has shown remarkable success in domains like facial attribute analysis, emotion detection, and biometric verification.

Caruana (1997) [13] introduced MTL as a method to improve generalization by leveraging the domain-specific information contained in the training signals of related tasks. In the context of face analysis, MTL is particularly advantageous because facial features such as eyes, nose, and mouth provide shared information relevant to multiple tasks like gender classification, age estimation, and ethnicity prediction.

Research has demonstrated that MTL can mitigate overfitting, especially when labeled data is limited. Studies by Ranjan et al. (2017)[14] highlight that MTL frameworks achieve state-of-the-art performance by balancing task-specific losses with a shared feature extractor. Moreover, task interdependence plays a critical role; for instance, learning age and gender together can improve the prediction accuracy of both tasks, as they share biological and social correlations.

Advanced implementations of MTL employ attention mechanisms and task-specific adapters to manage conflicts between tasks with varying levels of complexity or data availability. These methods further refine the shared representation to maximize task performance without sacrificing generalizability.

2.4. FairFace Dataset

The FairFace dataset [1] represents a significant advancement in addressing bias in face analysis datasets. Traditional datasets like LFW (Labeled Faces in the Wild) and CelebA,

while widely used, often exhibit skewed distributions, overrepresenting certain demographic groups (e.g., white males) while underrepresenting others. Such biases lead to models that perform poorly on underrepresented groups, raising concerns about fairness and equity in automated systems.

FairFace, introduced by Karkkainen and Joo (2021) [1], addresses these challenges by providing a balanced dataset annotated for gender, age, and ethnicity across seven racial groups. This diversity ensures that models trained on FairFace are better equipped to generalize across different demographic groups, reducing performance disparities.

The dataset contains over 108,000 high-quality face images with labels for attributes such as race (White, Black, Indian, East Asian, Southeast Asian, Middle Eastern, and Latino), gender (Male and Female), and age (seven age groups). It has been widely adopted in research and commercial applications for training fair and unbiased models. Studies using FairFace have shown improved fairness metrics, such as equalized odds and demographic parity, in tasks ranging from facial recognition to demographic attribute classification.

By addressing the limitations of earlier datasets, FairFace not only promotes ethical AI practices but also enables researchers to build robust systems applicable to diverse populations. Its balanced representation has made it a benchmark for fairness in face analysis tasks.

3. Methodology

3.1. Dataset

The FairFace dataset [1], introduced by Kärkkäinen and Joo (2021), is a large-scale facial dataset designed to address biases that exist in many facial recognition and demographic classification datasets. Traditional datasets, such as CelebA and LFW, often suffer from biases in representation, which can lead to suboptimal performance on underrepresented demographic groups. The FairFace dataset was created to mitigate this issue by ensuring balanced representation across multiple demographic groups, including racial, gender, and age categories.

3.1.1. Dataset Overview

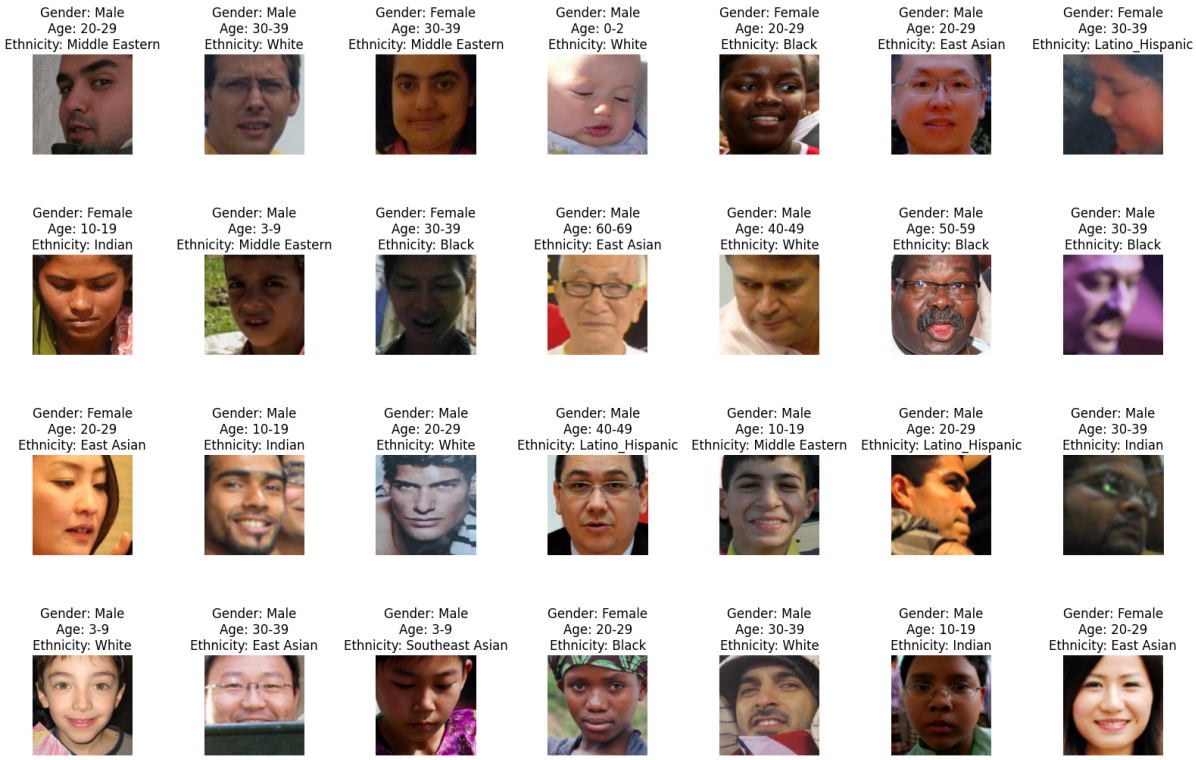


Figure 3.1: Sample Images from FairFace Dataset

The FairFace dataset contains around 100,000 images of faces, with annotations for gender, age group, and ethnicity. The images are sourced from the YFCC-100M dataset,

ensuring diversity in the facial characteristics represented. This dataset provides a critical resource for building AI systems that are not only accurate but also fair and ethical.

- Race/Ethnicity: Seven categories – White, Black, Indian, East Asian, Southeast Asian, Middle Eastern, and Latino.
- Gender: Male and Female.
- Age: Nine age groups – `"0-2"`, `"3-9"`, `"10-19"`, `"20-29"`, `"30-39"`, `"40-49"`, `"50-59"`, `"60-69"` and `"70+"`.

3.1.2. Key Features

- **Balanced Representation**: Unlike many other face datasets that have a skewed representation of certain groups, FairFace ensures an equal distribution of images across the seven racial/ethnic groups, promoting fairness in facial recognition and demographic prediction tasks.
- **High-Quality Annotations**: Each image is labeled with gender, age group, and ethnicity by multiple annotators to ensure accuracy and consistency in the data.
- **Focus on Fairness**: FairFace is specifically designed to help reduce the biases that often occur in machine learning models by providing a balanced and diverse dataset, making it an important tool for training more inclusive and fair AI systems.

3.1.3. Comparison with other datasets

Dataset	# Faces	In-the-wild?	Age	Gender	Ethnicity	Balanced?	White	Asian	Black
FairFace	108K	Yes	Yes	Yes	7 Categories	Yes	Yes	E, SE	Yes
UTKFace	20K	Yes	Yes	Yes	Merged	Yes	Yes	Merged	Yes
LFW+	15K	Yes	Yes	Yes	Merged	No	Yes	Merged	Yes
IMDB-WIKI	500K	Yes	Yes	Yes	No	No	Yes	No	No
MORPH	55K	Yes	Yes	Yes	Merged	No	Yes	No	Yes
FotW	25K	Yes	Yes	Yes	No	Yes	Yes	No	No
CACD	160K	Yes	Yes	No	No	No	Yes	No	No
CelebA	200K	Yes	No	Yes	Merged	No	Yes	No	No

Table 3.1: Comparison of FairFace Dataset with Other Public Face Datasets. Sources: FairFace [1], UTKFace [2], LFW+ [3], IMDB-WIKI [4], MORPH [5], FotW [6], CACD [7], CelebA [8].

3.2. Preprocessing

The preprocessing of data was an essential step in ensuring the FairFace dataset was in a suitable format for training the multitask deep learning model. The following steps outline the preprocessing procedure:

3.2.1. Data Loading and Preparation

The facial images were loaded using their file paths specified in the dataset's annotation file. The images were opened using the Python Imaging Library (PIL) and converted to the RGB color format to ensure consistency in the number of channels. This step was critical for accommodating deep learning models, which generally require RGB inputs.

3.2.2. Data Transformations

To prepare the images for training, a transformation pipeline was applied using PyTorch's `transforms` module. This pipeline consisted of the following:

- **Tensor Conversion:** The images were converted from their original PIL format into PyTorch tensors. This process scaled pixel values from their original range of $[0, 255]$ to $[0, 1]$, ensuring better numerical stability during training.
- **Normalization:** The pixel values of the images were standardized using channel-wise mean and standard deviation values derived from the ImageNet [11] dataset:
  - **Mean:** $[0.485, 0.456, 0.406]$
  - **Standard Deviation:** $[0.229, 0.224, 0.225]$

  Normalization ensured that the input data had zero mean and unit variance, optimizing the performance and convergence of the deep learning model. The mean and standard deviation values reflect typical statistics of natural images, making model initialization more robust.

Label Mapping

The dataset contained categorical labels for gender, age, and ethnicity, which were encoded into numerical values for compatibility with the model. The mappings were as follows:

- **Gender Mapping:**
  - `Male` $\rightarrow 0$, `Female` $\rightarrow 1$
- **Age Group Mapping:** The age groups (e.g., `"0-2"`, `"3-9"`, etc.) were mapped to integer values ranging from 0 to 8, effectively representing ordinal age groups.
- **Ethnicity Mapping:** Seven ethnicity categories (e.g., `"White"`, `"Black"`, etc.) were mapped to integers ranging from 0 to 6, ensuring compatibility with the model's output layer for ethnicity classification.

Output Format

Each preprocessed image, along with its corresponding labels for gender, age, and ethnicity, was returned as a tuple in the form:

$$(\text{image}, (\text{gender label}, \text{age group label}, \text{ethnicity label}))$$

This structure enabled efficient multitask learning, where the model shared input features but had separate outputs for each task.

Validation Data Preprocessing

For the validation dataset, a similar transformation pipeline was applied. The images underwent tensor conversion and normalization to ensure consistency between training and validation data.

3.3. Model Architecture

The proposed model, **LightweightMTLNet224**, is a compact and efficient multitask neural network designed to simultaneously predict *gender*, *age group*, and *ethnicity* from facial images. The architecture is tailored for resource-constrained environments by leveraging **Depthwise Separable Convolutions**, a computationally efficient alternative to standard convolution operations. Below is a detailed description of the model's components:

3.3.1. Depthwise Separable Convolution

The architecture incorporates **Depthwise Separable Convolution** [12] layers, which break down a standard convolution into two operations:

- **Depthwise Convolution**: Applies individual filters to each input channel independently, reducing computational overhead.
- **Pointwise Convolution**: Combines the outputs of the depthwise convolution across channels using a 1×1 kernel.

Each of these convolutional steps is followed by **Batch Normalization** for feature scaling and a **ReLU activation function** for introducing non-linearity. This efficient design allows for the extraction of spatial and channel-specific features at reduced computational cost.

3.3.2. Convolutional Layers

The model includes seven convolutional layers (Conv1 through Conv7), each progressively increasing feature depth and reducing spatial dimensions:

- **Conv1**: The input RGB image of size $224 \times 224 \times 3$ is processed into 64 feature maps.
- **Conv2 & Conv3**: These layers expand the feature maps to 128 channels.
- **Conv4 & Conv5**: Further increase the feature representation depth to 256 channels.
- **Conv6 & Conv7**: Extract high-level features with 512 channels.

These layers provide a hierarchical feature extraction pipeline, enabling the model to capture both low-level and high-level features from input images.

3.3.3. Pooling Layer

The model includes an **adaptive average pooling layer**, which reduces the spatial dimensions of the feature maps to 1×1. This layer provides a compact representation of the global features extracted by the convolutional layers, ensuring a fixed-size input for subsequent fully connected layers.

3.3.4. Fully Connected Layers

The output of the pooling layer is flattened and passed through:

- A **shared fully connected (FC) layer** with 256 neurons, which provides a common feature representation for all tasks. This layer uses a **ReLU activation function** for non-linearity.
- **Task-Specific Classifiers**:
  - A **gender classifier** predicting probabilities for two classes (male and female).
  - An **age group classifier** predicting probabilities for nine age groups.
  - An **ethnicity classifier** predicting probabilities for seven ethnic groups.

3.3.5. Multitask Outputs

The model's multitask learning framework ensures efficient feature sharing across tasks, while task-specific heads allow for specialization in gender, age group, and ethnicity predictions.

3.3.6. Pipeline Summary

1. The input image passes sequentially through seven convolutional layers (Conv1 to Conv7).
2. A global average pooling layer reduces the spatial dimensions of the feature maps.
3. The resulting features are passed through a shared fully connected layer.
4. Finally, task-specific classifiers output predictions for gender, age group, and ethnicity.

3.3.7. Advantages of the Architecture

- **Computational Efficiency**: The use of depthwise separable convolutions significantly reduces the number of parameters and computational complexity.
- **Compact Design**: The lightweight structure makes the model suitable for deployment on edge devices.
- **Multitask Learning**: Efficiently utilizes shared features while maintaining specialization across tasks.

3.4. Model Training

For our purpose of training the model for two gender classes, nine age groups and seven ethnicities the model had a total of 668077 trainable parameters.

The training procedure for the **LightweightMTLNet224** model involves training the model for a set number of epochs, during which the model learns to predict the three tasks: gender, age group, and ethnicity from facial images. The procedure is as follows:

3.4.1. Training and Validation Data

The training data has roughly 85,000 images and validation data has roughly 11,000 images.

Training Data

Figure 3.2: Distributions in the Training Data: Age, Ethnicity, and Gender.

The training data has significantly more people in the age group of `"20-29"` and `"30-39"`, the numbers are about the same for `"40-49"`, `"3-9"` and `"10-19"` (around 10000). For the age groups `"60-69"`, `"0-2"` and `"70+` the number of individuals is significantly less than 5000.

The distribution is fairly uniform across gender and Ethnicity. With both genders being almost equal and there is much significant variation in the number of people in different ethnicities.

	Max Ratio	Min Ratio	Ratio Range	Coefficient of Variation	Entropy	Gini Index
Age	0.295098	0.009707	0.285391	0.808729	2.695531	0.816217
Gender	0.530135	0.469865	0.060269	0.060269	0.997378	0.498184
Race	0.190526	0.106244	0.084282	0.169074	2.787104	0.853059

Table 3.2: Data Imbalance for Age, Gender, and Ethnicity in the training data

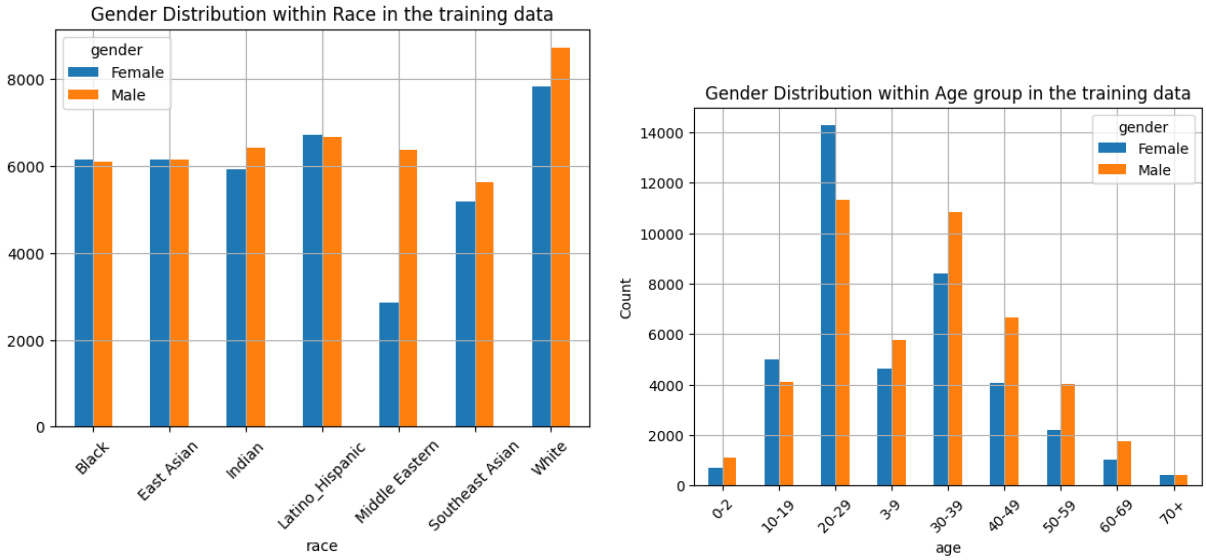


Figure 3.3: Distribution of Gender across age and ethnicity in training data

The distribution of gender is also fairly uniform across most age groups except 40-69 with significantly more males. It is also fairly even across all the ethnicity groups except Middle Eastern, where there are significantly more males than females.

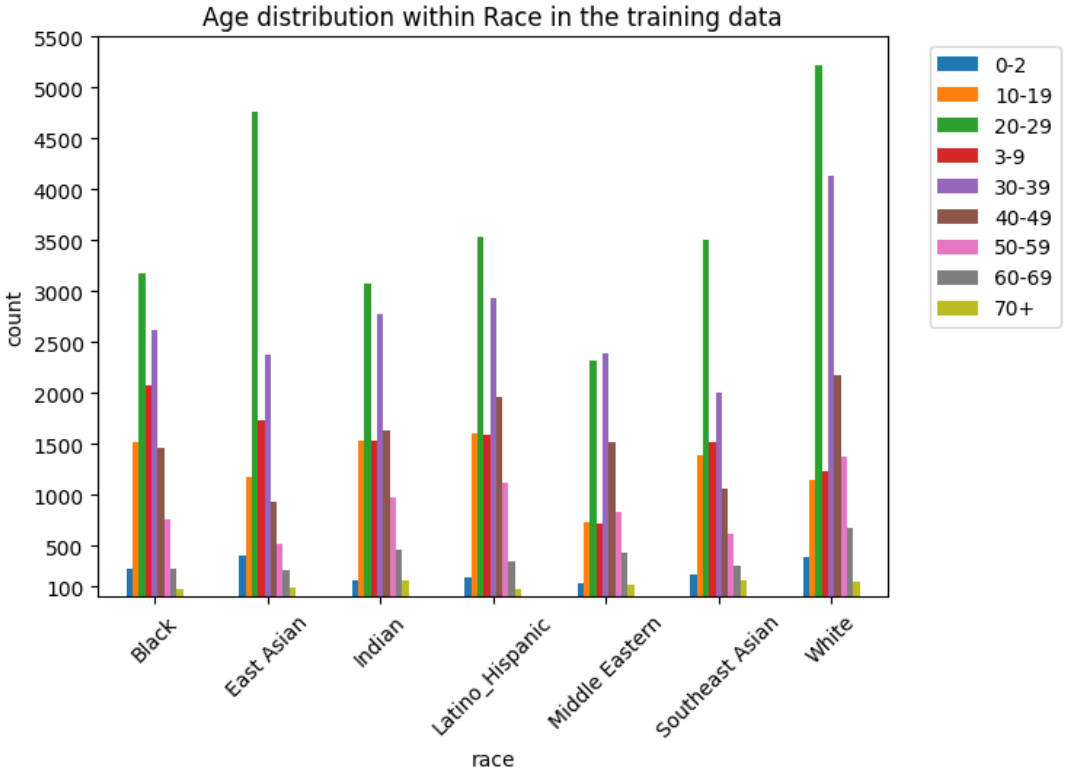


Figure 3.4: Distribution of age across ethnicity in the training data

The distribution of age groups between ethnicities is similar for all the ethnicity groups. Similar pattern is also observable in the distribution of ethnicities between age groups.

Validation Data

The distribution of gender, age groups and ethnicities in the validation data is almost identical to the training data. This is also easily observable for the distribution of gender across age groups and ethnicities as well as distribution of ethnicities across age groups.

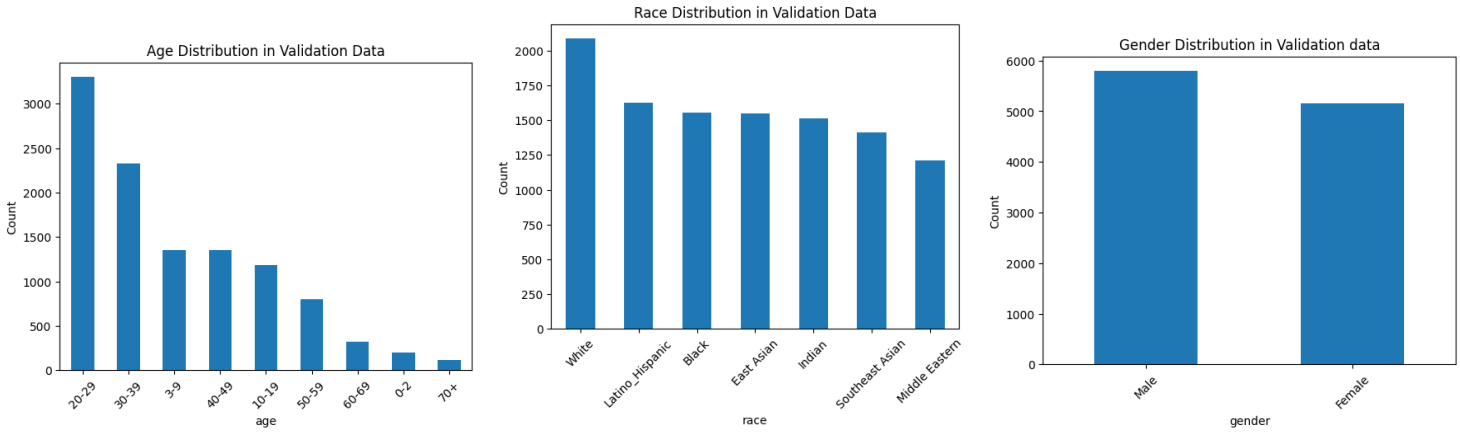


Figure 3.5: Distributions in the Validation Data: Age, Ethnicity, and Gender.

	Max Ratio	Min Ratio	Ratio Range	Coefficient of Variation	Entropy	Gini Index
Age	0.295098	0.009707	0.285391	0.808729	2.695531	0.816217
Gender	0.530135	0.469865	0.060269	0.060269	0.997378	0.498184
Race	0.190526	0.106244	0.084282	0.169074	2.787104	0.853059

Table 3.3: Data Imbalance for Age, Gender, and Ethnicity in the validation data

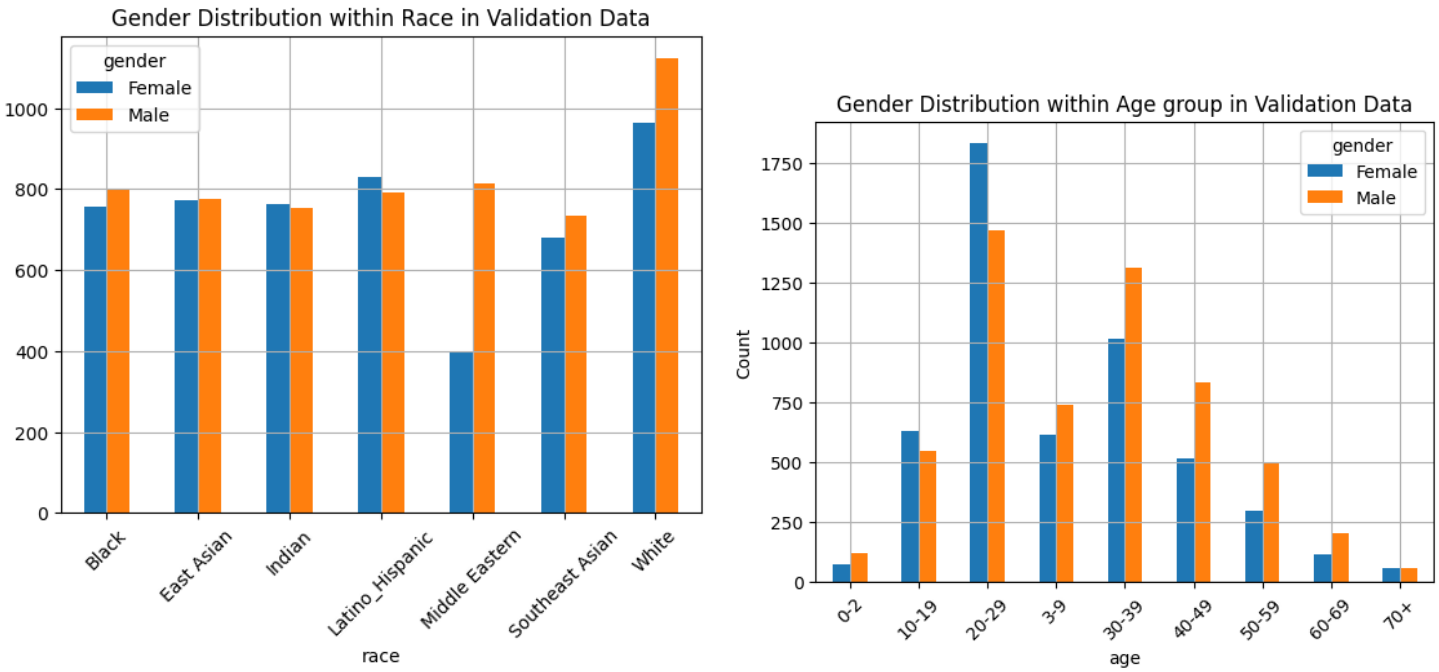


Figure 3.6: Distribution of Gender across age and ethnicity in validation data

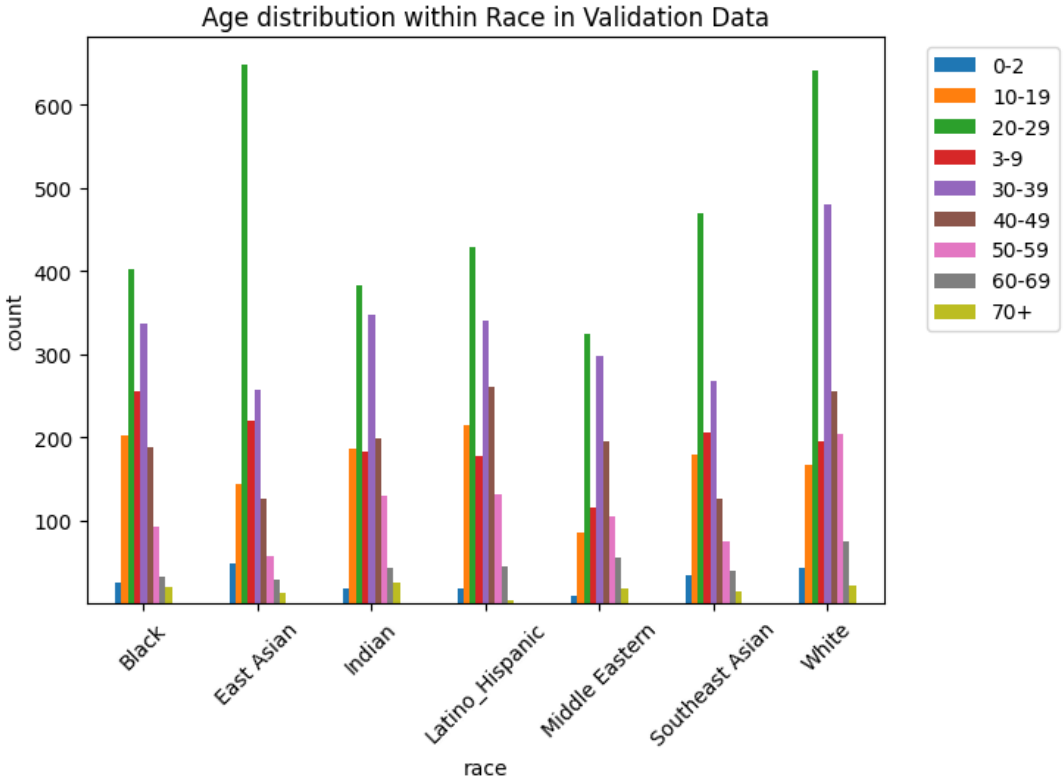


Figure 3.7: Distribution of age acorss ethnicity in validation data

3.4.2. Loss Functions

For each of the three tasks (gender, age, and ethnicity), we utilize the `CrossEntropyLoss` function. This loss function is appropriate for multi-class classification tasks, where the goal is to minimize the difference between the predicted and actual class labels. The loss functions for the three tasks are defined as follows:

- **Gender Loss:** A `CrossEntropyLoss` is used to minimize the error between the predicted and actual gender labels.
- **Age Loss:** A `CrossEntropyLoss` is employed to minimize the error between the predicted and actual age group labels.
- **Ethnicity Loss:** A `CrossEntropyLoss` is used to minimize the error between the predicted and actual ethnicity labels.

3.4.3. Optimizer

We use the Adam optimizer with a learning rate of 0.001 to update the model's weights during training. Adam is a popular choice due to its adaptive learning rate and efficient performance in training deep learning models.

- **Optimizer:** Adam
- **Learning Rate:** 0.001

3.4.4. Training Loop

The model is trained for a total of 10 epochs, and the following steps are executed during each epoch:

1. The model is set to training mode using `model.train()`.
2. For each mini-batch in the training set:
    - The images and corresponding labels for gender, age, and ethnicity are loaded and moved to the GPU (if available).
    - The optimizer gradients are reset using `optimizer.zero_grad()`.
    - The images are passed through the model to get the predictions for gender, age, and ethnicity.
    - The individual losses for gender, age, and ethnicity are computed using `CrossEntropyLoss`.
    - The total loss is calculated as the sum of the individual losses.
    - The gradients are backpropagated using `total_loss.backward()`.
    - The optimizer updates the model's weights using `optimizer.step()`.
3. After every 200 batches, the training loss and the individual task losses are printed to monitor the progress.

4. The running loss for each task is accumulated and averaged over all batches in the epoch.

5. After each epoch, the model is evaluated on the validation set.

6. The validation loss and individual task losses are computed similarly to the training phase.

3.4.5. Early Stopping

An early stopping criterion is employed to prevent overfitting. If the training loss does not improve significantly (i.e., if the loss does not decrease by at least 0.1 from the previous epoch), the training process is halted early. This is done to save computational resources and prevent the model from overfitting the training data.

- **Early Stopping Criterion:** If the difference between the current and previous epoch's loss is less than 0.1, training is stopped.

3.4.6. Monitoring and Evaluation

During training, the losses for each task (gender, age, and ethnicity) are tracked separately. At the end of each epoch, the following metrics are reported:

- **Training Loss:** The total loss averaged over all batches in the training set.
- **Gender Loss:** The loss associated with the gender prediction.
- **Age Loss:** The loss associated with the age group prediction.
- **Ethnicity Loss:** The loss associated with the ethnicity prediction.

Additionally, the model's performance on the validation set is evaluated at the end of each epoch, and metrics for each task are computed and logged.

3.5. Model Deployment

The deployment of the multi-task learning model for gender, age, and ethnicity classification was carried out using Streamlit, a popular open-source Python framework for building interactive web applications.

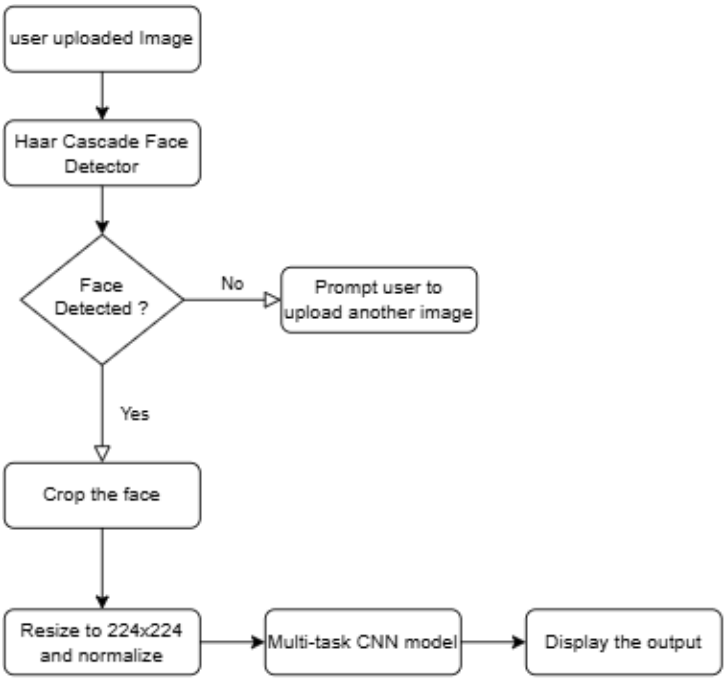


Figure 3.8: Model Pipeline

3.5.1. Model Pipeline

1. **Model Loading**:

   The first step in the pipeline involves loading the pre-trained model. Upon deployment, the model is loaded from a saved `.pth` file . The model is then set to evaluation mode to ensure it behaves correctly during inference.

2. **Face Detection**:

   Once an image is uploaded, the first task is to detect a face in the image. A pre-trained Haar Cascade classifier is used for face detection. This classifier identifies the face in the uploaded image and crops the image accordingly to focus on the face. If no face is detected, the user is prompted to upload another image.

3. **Image Preprocessing**:

   After face detection, the cropped image is transformed into the required format for input into the model. This includes resizing the image to 224x224 pixels, converting it into a tensor and normalizing the pixel values using standard ImageNet mean and standard deviation values.

4. **Prediction**:

   The processed image is passed through the model to generate predictions. The model outputs three sets of probabilities corresponding to the gender, age, and ethnicity classes. These outputs are then processed using the `torch.softmax()` function to convert the raw output scores into probabilities.

5. **Display Results**:

   The predicted labels (gender, age, and ethnicity) are displayed on the web interface along with their respective probabilities. The user is presented with the image

of the uploaded face and the corresponding predicted labels. Additionally, the probabilities for each class are visualized using bar charts generated by `matplotlib`.

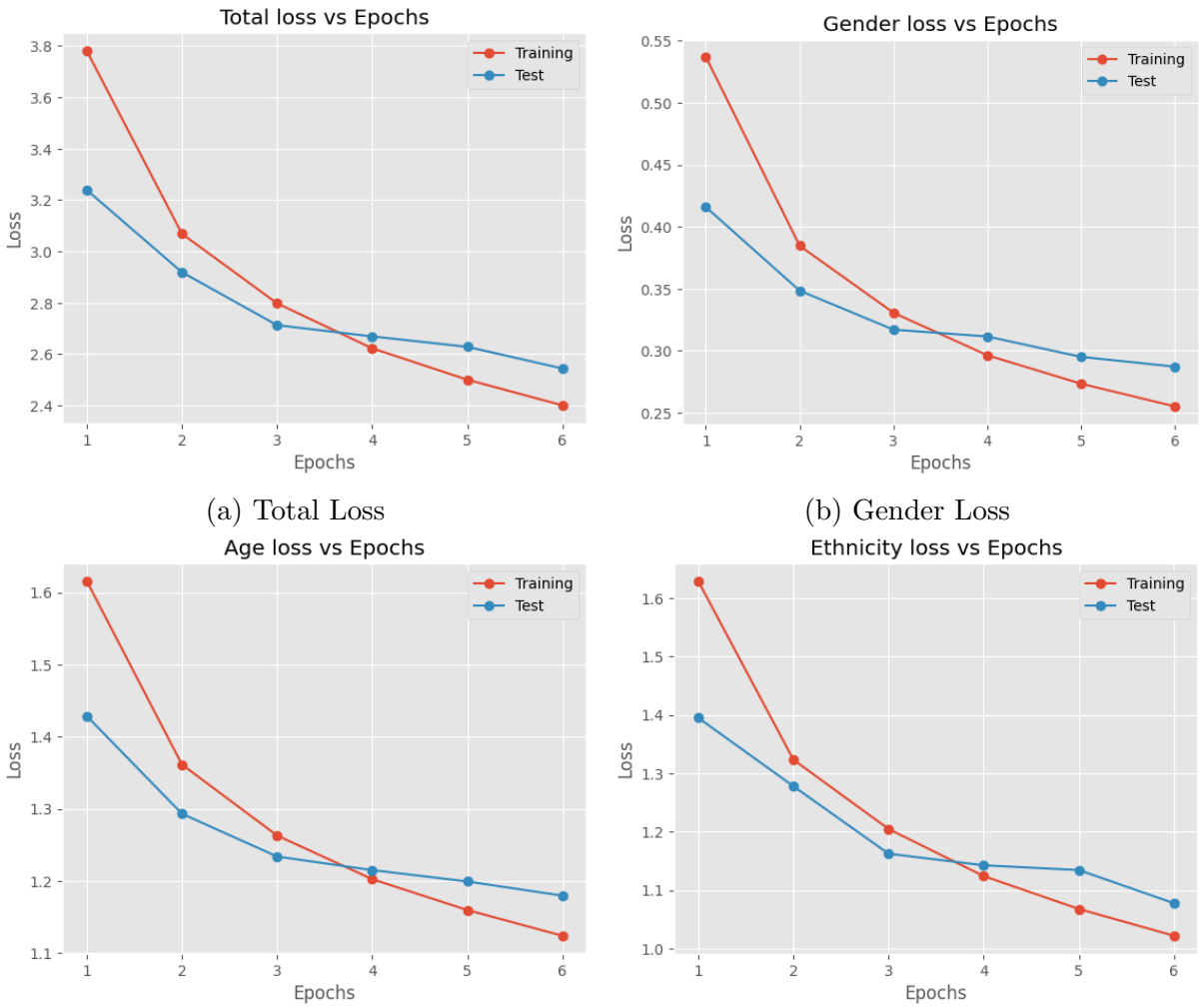
3.5.2. Technology Stack

The following technologies were used for the model deployment:

- **Streamlit**: For building the interactive web application interface.
- **PyTorch**: For implementing and deploying the multi-task learning model.
- **OpenCV**: For face detection using the Haar Cascade classifier.
- **Matplotlib**: For visualizing the predicted probabilities as bar charts.
- **NumPy and PIL**: For image manipulation and processing.

The deployment pipeline ensures an end-to-end solution that takes an uploaded image, processes it, applies the pre-trained model, and returns the predictions in an accessible and user-friendly manner.

4. Results and Discussion



(a) Total Loss

(b) Gender Loss

(c) Age Loss

(d) Race Loss

Figure 4.1: Training losses for different aspects of our model.

The training for the model terminated after 6 epochs. With both training and test loss having similar values at the end of the training. The gender loss was the lowest at the end of the training while age loss and race loss had similar values much higher than the gender loss.

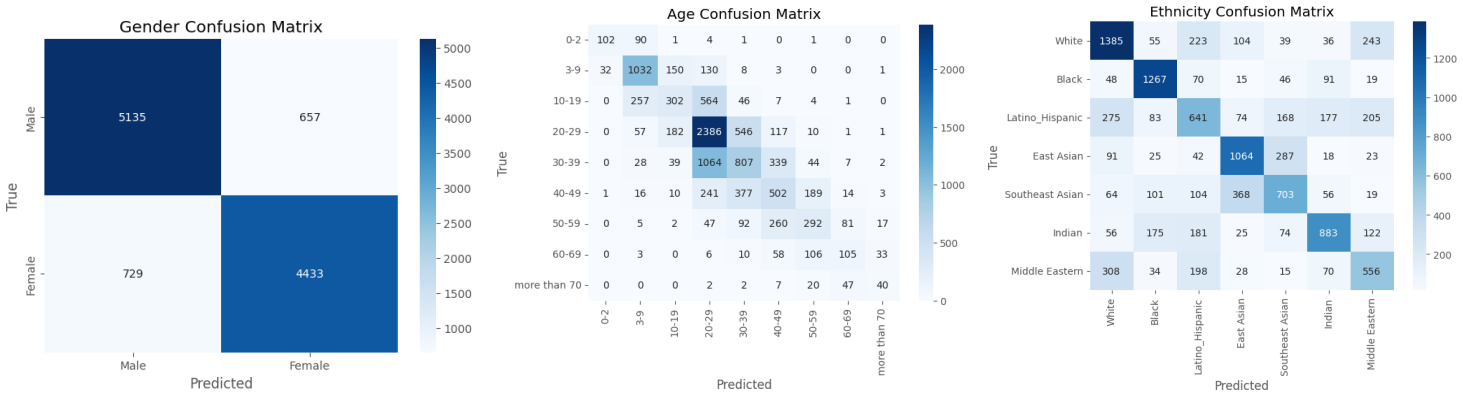


Figure 4.2: Confusion matrices for Gender, Age and Ethnicity

We also look at the confusion matrices for gender, age and ethnicity classification. The performance was the best for gender classification with an accuracy of 87.35%, followed by ethnicity and age group with 59.33% and 50.83% respectively.

We also observe that most of the cases when age group was incorrectly predicted it was predicted to be in the age group just smaller or bigger than the true age group. The model incorrectly predicted a lot of people in `"30-39"` to be in `"20-29"`, this was especially more true for female than male. Hinting a possible bias in the model for age group classification for female.

For ethnicity classification most of the misclassifications were between the groups "East Asian and Southeast Asia", "Middle Eastern and White", "Latino Hispanic and White", and "Indian and Latino Hispanic". There did not seem any significant difference in ethnicity classification for both the genders.

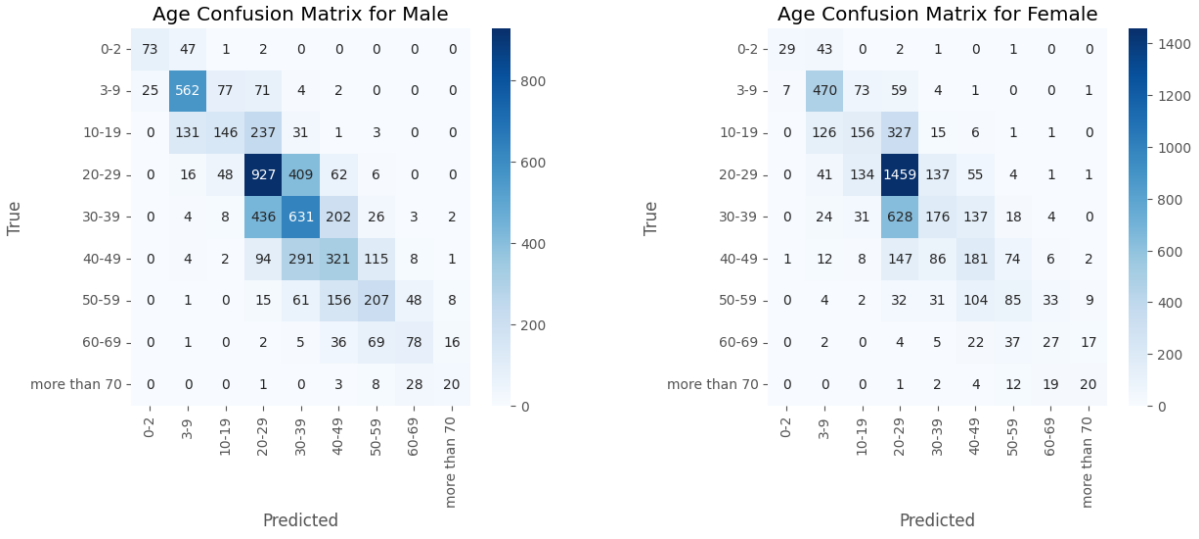


Figure 4.3: Age Confusion Matrices across gender

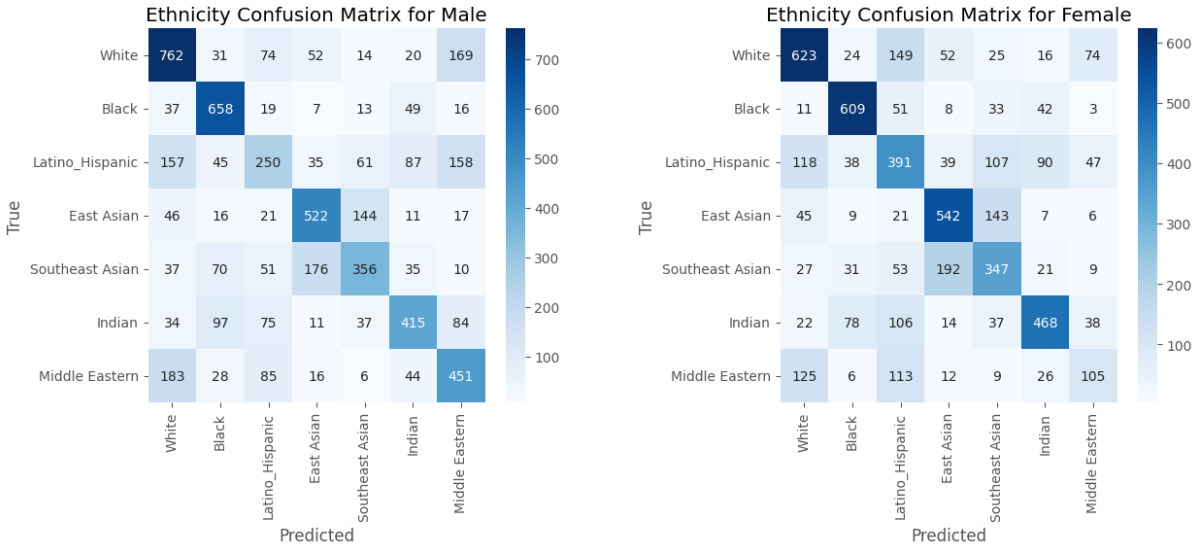


Figure 4.4: Race Confusion Matrices across gender

Age Group	White	Black	Latino/Hispanic	East Asian	Southeast Asian	Indian	Middle Eastern
0-2	0.66	0.68	0.84	0.76	0.56	0.61	1.00
3-9	0.73	0.69	0.70	0.83	0.84	0.72	0.72
10-19	0.74	0.68	0.87	0.84	0.83	0.75	0.82
20-29	0.92	0.83	0.93	0.92	0.90	0.95	0.95
30-39	0.92	0.87	0.96	0.89	0.89	0.93	0.95
40-49	0.93	0.87	0.93	0.85	0.89	0.94	0.95
50-59	0.90	0.79	0.95	0.90	0.84	0.90	0.94
60-69	0.84	0.81	0.91	0.80	0.77	0.82	0.95
70+	0.82	0.80	0.75	0.86	0.87	0.76	0.83

Table 4.1: Gender Accuracy Across Age and Ethnicity

Gender	White	Black	Latino/Hispanic	East Asian	Southeast Asian	Indian	Middle Eastern
Male	0.48	0.45	0.51	0.58	0.54	0.52	0.53
Female	0.51	0.43	0.49	0.59	0.50	0.48	0.55

Table 4.2: Age Accuracy Across Gender and Ethnicity

Gender	0-2	3-9	10-19	20-29	30-39	40-49	50-59	60-69	70+
Male	0.61	0.65	0.62	0.60	0.58	0.57	0.50	0.56	0.57
Female	0.55	0.58	0.64	0.59	0.59	0.58	0.60	0.61	0.60

Table 4.3: Ethnicity Accuracy Across Gender and Age

Inferences Based on the Tables

Gender Accuracy Across Age and Ethnicity

- **General Observations:**
  - Gender classification accuracy is highest in the 20-39 age groups across almost all ethnicities, indicating better performance for adults.
  - Accuracy declines significantly for the youngest (0-2 years) and oldest (70+) age groups.
- **Ethnic-Specific Trends:**
  - The accuracy is notably high for **Middle Eastern** individuals across most age groups, peaking at 20-49 years.
  - Most ethnicity groups show relatively lower accuracy for younger age groups (0-2 and 3-9), suggesting potential challenges in handling these demographics for gender prediction.
  - Accuracy for **Black** individuals is consistently lower compared to other groups across age categories.

Age Accuracy Across Gender and Ethnicity

- **General Observations:**
  - **Male subjects** tend to have slightly better age prediction accuracy than females across most ethnic groups.
  - **East Asian** exhibit the highest accuracy for both genders, indicating these groups may be better represented in the training data in terms of both image quality and quantity.
  - **Black** exhibit the lowest accuracy for both genders.
- **Ethnic-Specific Trends:**
  - **Latino/Hispanic** and **Black** groups show lower age prediction accuracy, suggesting potential biases or limitations in model generalization for these groups.
  - Gender-based disparities are not very pronounced in all the groups, reflecting relatively balanced performance.
- **Room for Improvement:**
  - Accuracy values are below 60% across all gender-ethnicity combinations, highlighting the need for improved age prediction mechanisms.

Ethnicity Accuracy Across Gender and Age

- **Age-Based Trends:**
  - Ethnicity prediction accuracy tends to decline for both genders in extreme age groups (0-2 and 70+), likely due to lesser distinct ethnic features in infants and elderly individuals.
  - The highest accuracy is observed in the 10-19 age group for **females** and the 3-9 age group for **males**, potentially reflecting better feature extraction for younger subjects.
- **Gender-Based Trends:**
  - Female accuracy is generally higher than male accuracy in most age groups, particularly in older (50+) categories, suggesting the model might better capture ethnicity-related features in females.

5. Conclusion and Future Work

While the current study has provided a thorough analysis of classifier performance across different demographic groups, several areas can be explored to further enhance the quality and generalization of the model. Some of the key directions for future work include:

5.1. Improving Data Imbalance Handling

Despite using various metrics such as Coefficient of Variation, Gini Index, and Entropy to analyze data imbalance, further improvements in handling class imbalance could be explored. Specifically, the following methods can be investigated:

- **Advanced Resampling Techniques:** Although SMOTE (Synthetic Minority Over-sampling Technique)[18] was considered for balancing, advanced techniques such as Borderline-SMOTE, ADASYN, or ensemble methods for resampling can be applied to improve the performance of classifiers in imbalanced scenarios [15].
- **Class Weight Adjustment:** Another potential direction is to adjust the weights of classes during training to give higher importance to the minority classes, ensuring the model's bias is reduced towards the majority class.
- **Cost-sensitive Learning:** Exploring cost-sensitive learning frameworks, where different misclassification errors are penalized according to the class distribution, could further enhance model performance in imbalanced datasets.

5.1.1. Model Optimization and Hyperparameter Tuning

Future work could involve an extensive optimization process to improve the model's hyperparameters. Current methods like grid search and random search could be expanded by applying more sophisticated optimization techniques such as:

- **Bayesian Optimization:** A probabilistic model can be used to explore the hyperparameter space efficiently and find optimal configurations for the classifier.
- **Hyperparameter Importance Analysis:** By conducting a more systematic hyperparameter importance analysis, we can identify which factors most influence the performance of the classifier across different groups.

5.1.2. Exploring More Advanced Models

Future work could experiment with more advanced/complex models, particularly deep learning methods:

- **Generative Adversarial Networks (GANs)** GANs can be used for data augmentation, especially in imbalanced datasets. They generate synthetic samples for underrepresented classes to balance the dataset, which can enhance model performance.
- **Transfer Learning:** Leveraging pre-trained models (such as those trained on large datasets like ImageNet or using domain-specific knowledge) for transfer learning can be explored for classification tasks, especially when data is scarce or unbalanced.

5.1.3. Fairness and Bias Analysis

Addressing fairness and bias in machine learning models is essential, particularly when dealing with sensitive demographic factors. Future work should include:

- **Fairness Metrics:** Metrics such as demographic parity, equal opportunity, and disparate impact should be introduced to evaluate and mitigate bias in model predictions across different groups.
- **Bias Mitigation Techniques:** Techniques like adversarial debiasing, bias correction algorithms, and fairness constraints could be applied to ensure that the model performs equitably across different demographic groups.

5.1.4. Model Interpretability

Implementing explainable AI (XAI) methods such as LIME (Local Interpretable Model-agnostic Explanations) or SHAP (Shapley Additive Explanations) to provide transparency regarding how decisions are made by the model.

5.1.5. Conclusion

In conclusion, there are numerous opportunities to improve the performance, fairness, and applicability of the model in addressing demographic disparities. Incorporating more advanced techniques, expanding the dataset, and considering ethical implications will enable the development of more robust and equitable models for real-world applications.

6. Appendix

6.1. Model Implementation Code

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class DepthwiseSeparableConv(nn.Module):
    def __init__(self, in_channels, out_channels, stride=1):
        super(DepthwiseSeparableConv, self).__init__()
        self.depthwise = nn.Conv2d(in_channels, in_channels, kernel_size=3, stride=
            stride, padding=1, groups=in_channels, bias=False)
        self.pointwise = nn.Conv2d(in_channels, out_channels, kernel_size=1, bias=False
            )
        self.bn = nn.BatchNorm2d(out_channels)
        self.relu = nn.ReLU(inplace=True)

    def forward(self, x):
        x = self.depthwise(x)
        x = self.pointwise(x)
        x = self.bn(x)
        return self.relu(x)

class LightweightMTLNet224(nn.Module):
    def __init__(self, num_classes_gender=2, num_classes_age=9, num_classes_ethnicity
        =7):
        super(LightweightMTLNet224, self).__init__()

        # Input layer
        self.conv1 = DepthwiseSeparableConv(3, 64, stride=2)
        self.conv2 = DepthwiseSeparableConv(64, 128, stride=1)
        self.conv3 = DepthwiseSeparableConv(128, 128, stride=2)
        self.conv4 = DepthwiseSeparableConv(128, 256, stride=1)
        self.conv5 = DepthwiseSeparableConv(256, 256, stride=2)
        self.conv6 = DepthwiseSeparableConv(256, 512, stride=1)
        self.conv7 = DepthwiseSeparableConv(512, 512, stride=2)

        # Pooling layer
        self.avgpool = nn.AdaptiveAvgPool2d((1, 1))

        # Task-specific classifiers
        self.fc = nn.Linear(512, 256)
        self.gender_fc = nn.Linear(256, num_classes_gender)
        self.age_fc = nn.Linear(256, num_classes_age)
        self.ethnicity_fc = nn.Linear(256, num_classes_ethnicity)

```

```
    def forward(self, x):
        # Pass through convolutional layers
        x = self.conv1(x)
        x = self.conv2(x)
        x = self.conv3(x)
        x = self.conv4(x)
        x = self.conv5(x)
        x = self.conv6(x)
        x = self.conv7(x)

        # Global average pooling
        x = self.avgpool(x)
        x = torch.flatten(x, 1)

        # Fully connected layers
        x = F.relu(self.fc(x), inplace=False)

        # Multi-task outputs
        gender = self.gender_fc(x)
        age = self.age_fc(x)
        ethnicity = self.ethnicity_fc(x)

        return gender, age, ethnicity
```

Listing 6.1: LightweightMTLNet224 Implementation

6.2. Model Training Code

```
# Loss functions for each output
gender_criterion = nn.CrossEntropyLoss()
age_criterion = nn.CrossEntropyLoss()
ethnicity_criterion = nn.CrossEntropyLoss()

# Optimizer
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

num_epochs = 10
losses = []
test_losses = []
losses_gender = []
losses_age = []
losses_ethnicity = []
test_losses_gender = []
test_losses_age = []
test_losses_ethnicity = []

for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    running_loss_gender = 0.0
    running_loss_age = 0.0
    running_loss_ethnicity = 0.0

    batch_idx = 1
    for images, (gender_labels, age_labels, ethnicity_labels) in train_loader:
        images = images.to(device)
```

```
        gender_labels = gender_labels.to(device)
        age_labels = age_labels.to(device)
        ethnicity_labels = ethnicity_labels.to(device)

        optimizer.zero_grad()

        # Forward pass
        gender_output, age_output, ethnicity_output = model(images)

        # Compute individual losses and combine them
        gender_loss = gender_criterion(gender_output, gender_labels)
        age_loss = age_criterion(age_output, age_labels)
        ethnicity_loss = ethnicity_criterion(ethnicity_output, ethnicity_labels)

        total_loss = gender_loss + age_loss + ethnicity_loss
        total_loss.backward()
        optimizer.step()

        if batch_idx % 200 == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f} \tGender Loss: {:.6f
                } \tAge Loss: {:.6f}\tEthnicity Loss: {:.6f}'.format(
                epoch+1, batch_idx * len(gender_labels), len(train_dataset),
                100. * batch_idx / len(train_loader), total_loss.item(),
                gender_loss.item(), age_loss.item(), ethnicity_loss.item()))
        batch_idx += 1

        running_loss += total_loss.item()
        running_loss_gender += gender_loss.item()
        running_loss_age += age_loss.item()
        running_loss_ethnicity += ethnicity_loss.item()

    losses.append(running_loss/len(train_loader))
    losses_gender.append(running_loss_gender/len(train_loader))
    losses_age.append(running_loss_age/len(train_loader))
    losses_ethnicity.append(running_loss_ethnicity/len(train_loader))

    print(f"Epoch [{epoch+1}/{num_epochs}], Training Loss: {running_loss/len(
        train_loader):.4f}, Gender Loss: {running_loss_gender/len(train_loader):.4f},
        Age Loss: {running_loss_age/len(train_loader):.4f}, Ethnicity Loss: {
        running_loss_ethnicity/len(train_loader):.4f}")

    running_loss = 0.0
    running_loss_gender = 0.0
    running_loss_age = 0.0
    running_loss_ethnicity = 0.0

    for images, (gender_labels, age_labels, ethnicity_labels) in val_loader:

        images = images.to(device)
        gender_labels = gender_labels.to(device)
        age_labels = age_labels.to(device)
        ethnicity_labels = ethnicity_labels.to(device)

        gender_output, age_output, ethnicity_output = model(images)

        # Compute individual losses and combine them
        gender_loss = gender_criterion(gender_output, gender_labels)
        age_loss = age_criterion(age_output, age_labels)
```

```
        ethnicity_loss = ethnicity_criterion(ethnicity_output, ethnicity_labels)

        total_loss = gender_loss + age_loss + ethnicity_loss

        running_loss += total_loss.item()
        running_loss_gender += gender_loss.item()
        running_loss_age += age_loss.item()
        running_loss_ethnicity += ethnicity_loss.item()

    test_losses.append(running_loss/len(val_loader))
    test_losses_gender.append(running_loss_gender/len(val_loader))
    test_losses_age.append(running_loss_age/len(val_loader))
    test_losses_ethnicity.append(running_loss_ethnicity/len(val_loader))
    print(f"Epoch [{epoch+1}/{num_epochs}], Test Loss: {running_loss/len(val_loader)
        :.4f}, Gender Loss: {running_loss_gender/len(val_loader):.4f}, Age Loss: {
        running_loss_age/len(val_loader):.4f}, Ethnicity Loss: {running_loss_ethnicity
        /len(val_loader):.4f}")

    evaluate(model, val_loader)

    # Stopping criteria
    if len(losses) > 1 and (losses[-1] > losses[-2] or losses[-2] - losses[-1] < 0.1):
        print("Early stopping triggered")
        break
```

Listing 6.2: LightweightMTLNet224 Training

Bibliography

[1] Kimmo Karkkainen and Jungseock Joo, "FairFace: Face Attribute Dataset for Balanced Race, Gender, and Age for Bias Measurement and Mitigation", *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[2] UTKFace Dataset: Z. Zhang et al. *Age, Gender and Ethnicity Recognition with Deep Learning*.

[3] LFW+ Dataset: G. Huang et al. *Labeled Faces in the Wild*.

[4] IMDB-WIKI Dataset: R. Rothe et al. *Deep Expectation of Real and Apparent Age from a Single Image*.

[5] MORPH Dataset: K. Ricanek et al. *Morph: Longitudinal Data*.

[6] FotW Dataset: H. Kim et al. *Faces of the World: Diversity in Face Recognition*.

[7] CACD Dataset: B. Chen et al. *Cross-Age Reference Coding for Age-Invariant Face Recognition*.

[8] CelebA Dataset: Z. Liu et al. *Deep Learning Face Attributes in the Wild*.

[9] Paul Viola and Michael Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[10] Timo Ojala, Matti Pietikainen, and Topi Maenpaa, "Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[12] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,2017*

[13] Rich Caruana, "Multitask Learning", *Machine Learning*, 1997.

[14] Rudra Ranjan, Swami Sankaranarayanan, Carlos Castillo, and Rama Chellappa, "An All-In-One Convolutional Neural Network for Face Analysis", *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, 2017.

[15] S. S. Ghosh, "Coefficient of Variation: An Overview", *Journal of Statistical Analysis*, vol. 12, no. 3, pp. 198-212, 2004.

[16] C. E. Shannon, "A Mathematical Theory of Communication", *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[17] C. Gini, "Variability and Mutability, Contribution to the Study of Statistical Distributions", *Memorie della R. Academia delle Scienze di Torino*, vol. 3, pp. 3-37, 1912.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique", *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357, 2002.