



WIPRO NGA Program – Data Engineering & AI Batch

Capstone Project Presentation – 17th July 2024

Project Title - Global Face Detection: Race Identification using Deep Learning

Presented By – Vuppili Hruday Deep

Shanmukh Goud Jujjuri

Tellapati Jayanth

Thanushree PS

Vishnu Jeeth Pambala

INTRODUCTION

This project addresses the challenge of race detection using face images. By leveraging a Convolutional Neural Network (CNN) and a pre-trained ResNet model, we aim to create an accurate and reliable system capable of identifying an individual's race from their facial features. The dataset used is the 'UTK_face-dataset' from Kaggle, which contains images from various racial backgrounds. This project explores the data preprocessing steps, model architecture, training process, and evaluation results.

LITERATURE SURVEY

Face recognition offers a powerful tool for security systems, but privacy concerns are ever-present. Researchers are addressing this challenge by leveraging privacy-preserving multi-edge computing. Mao et al. (2018) discussed training face recognition models on edge devices with differential privacy, keeping user data on the edge to minimize centralized server reliance. Xu et al. (2024) explored federated learning to preserve user privacy during training, aggregating local models on a central server without accessing raw data. Zhang et al. (2021) proposed using Householder matrices to blind user data for edge server recognition tasks. Nabil et al. (2022) introduced a secure multi-party computation protocol for lightweight face recognition on IoT devices. Feng et al. (2023) focused on balancing security and privacy in edge-based surveillance with a face de-identification scheme. These approaches highlight the ongoing efforts to integrate privacy-preserving techniques in face recognition systems.

SCOPE

This project aims to develop a deep learning model to accurately identify the race of individuals from facial images, involving data collection, model development, training, evaluation, and results analysis

OBJECTIVE

The objective of this project is to develop and validate a deep learning-based race identification system from facial images, adhering to the SMART criteria:

1. **Specific:** To design and implement a Convolutional Neural Network (CNN) that accurately identifies the race of individuals from the 'race-dataset' of facial images.
2. **Measurable:** Achieve a model accuracy of at least 85% on the test dataset, with precision, recall, and F1 score metrics above 80%.
3. **Achievable:** Utilize a pre-trained deep learning model, appropriate data preprocessing techniques, and a well-structured dataset to ensure the objective is realistic and attainable.
4. **Relevant:** Address the practical need for automated race identification in various applications such as demographic analysis, security, and personalized services.
5. **Time-bound:** Complete the model development, training, evaluation, and documentation within a given time frame.

GOAL

The goal of this project is to develop a machine learning model that can accurately predict the race of an individual based on their face image. This involves creating a robust CNN architecture, training it on a diverse dataset, and evaluating its performance using various metrics.

DATASET

The dataset used for this project is the 'UTK-dataset' from Kaggle, which contains images of individuals from different racial backgrounds. The dataset is organized into three parts: part1,

part2, and part3, each containing images in the format [age][gender][race]_[date&time].jpg. The races are labelled as follows:

- 0: White
- 1: Black
- 2: Asian
- 3: Indian
- 4: Others

METHODOLOGY

1. DATA PREPROCESS

Data preprocessing involved resizing the images to a uniform size of 224x224 pixels, applying random horizontal flips for data augmentation, converting images to tensors, and normalizing them. This ensures that the images are in a consistent format and improves the model's ability to generalize.

2. FEATURE ENGINEERING

Feature engineering in this project involves the use of pre-trained CNN models to extract relevant features from the images. The ResNet model is fine-tuned to adapt to our specific classification task, allowing it to learn complex patterns and features relevant to race detection.

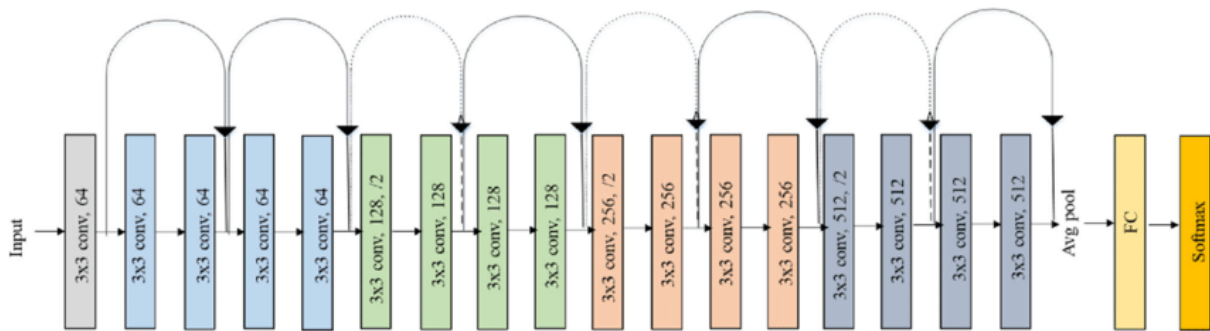
3. TRAIN TEST SPLIT

The dataset is split into training and testing sets with an 80-20 ratio. This ensures that the model has enough data to learn from while also providing a separate set for evaluation to test its performance.

4. MODEL ALGORITHM

The model used in this project is a pre-trained ResNet CNN, modified to fit our classification task.

a. RESIDUAL NETWORK (ResNet):



Residual Network it is convolutional network layer. We can load a pretrained version of the network trained on more than **a million images from the ImageNet** database. The pretrained network can classify images into **1000 object categories**, such as keyboard, mouse, pencil, and many animals.

ResNet-18 is part of the ResNet (Residual Network) family, which introduced the concept of residual learning. The main idea is to create shortcut connections (residual connections) that **skip** one or more layers. This helps in training very deep networks by mitigating the vanishing gradient problem.

1. Convolutional Layer:

- Initial convolutional layer with a 3 x 3 kernel, stride of 2, and 64 output channels.
- **Batch normalization** and **ReLU activation** follow this layer.

2. Max Pooling Layer:

- A max pooling layer with a **3x3 kernel** and a **stride of 2**.

3. Residual Blocks:

- **Four main stages** of residual blocks, where each stage has multiple **convolutional layers** with **batch normalization** and **ReLU activation**.
- **Shortcut connections** (skip connections) are added to allow the **gradient to flow** through the network more effectively.

4. Average Pooling Layer:

- A **global average pooling layer** that reduces the spatial dimensions of the feature map to a single value per feature map.

5. Fully Connected Layer:

- The final layer, which originally outputs **1000 classes** for the ImageNet dataset, but here it outputs the number of classes specified by **num_classes (5)**.

i. Advantages of ResNet

1. Each block augments the data
2. Shorter the Gradient path leads to faster training
3. Modularity

ii. Purpose of ResNet in our Project

1. Handling Complex Facial Features:

Race detection from face images is a complex task that requires capturing subtle differences in facial features across different races. The depth and architecture of ResNet allow it to effectively capture these intricate patterns.

2. Improved Generalization:

Deep networks like ResNet can generalize better on unseen data due to their capacity to learn more abstract representations. This is crucial for tasks like race detection, where the model needs to perform well across diverse and unseen faces.

3. Stability in Training:

The residual blocks in ResNet help in stabilizing the training process, especially when dealing with deep networks. This stability ensures that the model can be trained effectively without getting stuck in local minima or facing convergence issues.

6. PyTorch

1. **Dynamic Computational Graph:**

Allows for flexibility and ease of debugging, enabling quick iterations and modifications to the model architecture.

2. Ease of Use and Pythonic Nature: Intuitive and integrates seamlessly with Python, simplifying the learning curve and enhancing productivity.

3. Extensive Pre-trained Models: Provides access to a wide range of pre-trained models, such as ResNet, which can be used for transfer learning to improve performance and reduce training time.

4. Strong Community and Ecosystem: Offers extensive documentation, tutorials, and third-party resources, which are beneficial for troubleshooting and staying updated with the latest advancements in deep learning.

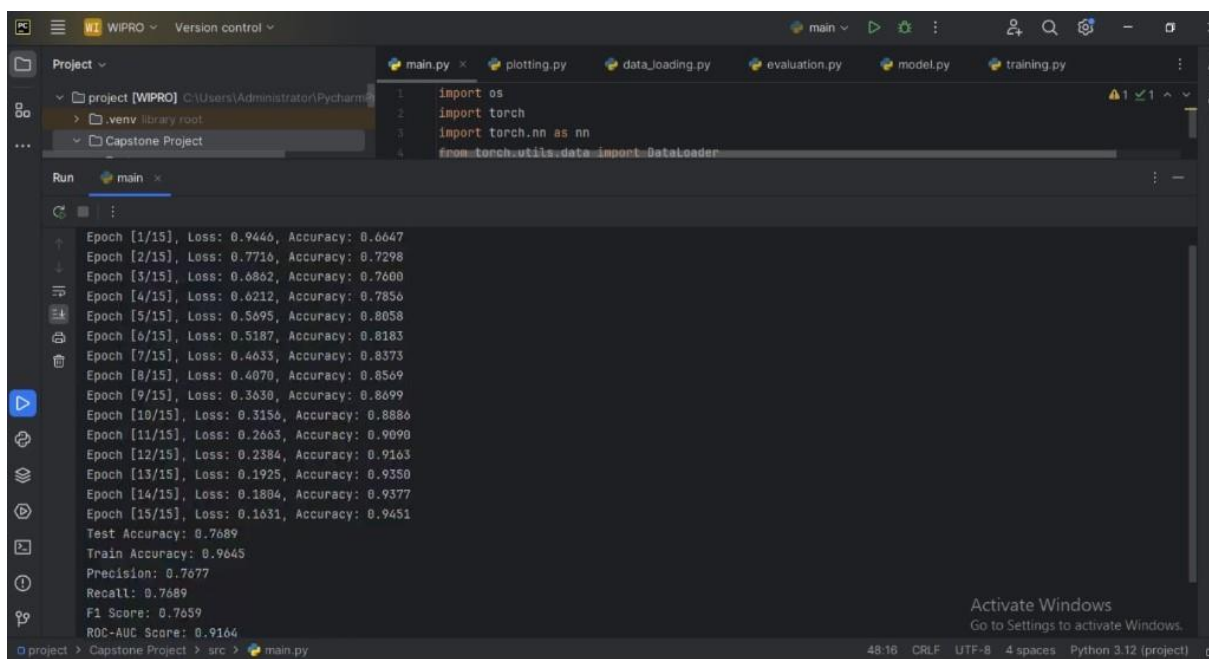
7. TRAINING

- It initializes the model, loss function, and optimizer.
- **Adam Optimization:** It adjusts the learning rate dynamically based on the estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients.
 - This adaptive learning rate mechanism allows Adam to converge faster and more reliably compared to methods with fixed learning rates.
 - During training, it computes predictions, calculates losses, and updates parameters.
- **CrossEntropyLoss:** Cross-entropy loss, also known as log loss, is commonly used for classification problems. It measures the difference between two probability distributions: the true labels and the predicted probabilities from the model.
 - Tracks and prints training metrics.
 - Returns training losses and accuracies.
 - Iteratively improves model predictions over epochs.

8. TESTING

- **Precision:** Measures the accuracy of positive predictions. It's the ratio of correctly predicted positive observations to the total predicted positive observations.
- **Recall:** Calculates the proportion of actual positives that were correctly identified by the model. It's the ratio of correctly predicted positive observations to all observations in the actual class.
- **F1 Score:** Harmonic mean of precision and recall. It provides a single score that balances both metrics, useful when classes are imbalanced.
- **ROC-AUC (Receiver Operating Characteristic - Area under Curve):** AUC represents the degree of separability between classes. Higher AUC indicates better model performance in distinguishing between classes.
- **Accuracy** – The **test accuracy** that our model is showing around **76%** with the **loss** of **0.1631**

These metrics collectively provide a comprehensive evaluation of the model's performance in terms of accuracy, robustness, and capability to generalize to new data.



The screenshot shows a PyCharm IDE with a project named 'Capstone Project'. The main.py file is open, showing the following code:

```
1 import os
2 import torch
3 import torch.nn as nn
4 from torch.utils.data import DataLoader
```

The Run console shows the following output:

```
Epoch [1/15], Loss: 0.9446, Accuracy: 0.6647
Epoch [2/15], Loss: 0.7716, Accuracy: 0.7298
Epoch [3/15], Loss: 0.6862, Accuracy: 0.7600
Epoch [4/15], Loss: 0.6212, Accuracy: 0.7856
Epoch [5/15], Loss: 0.5695, Accuracy: 0.8058
Epoch [6/15], Loss: 0.5187, Accuracy: 0.8183
Epoch [7/15], Loss: 0.4633, Accuracy: 0.8373
Epoch [8/15], Loss: 0.4070, Accuracy: 0.8569
Epoch [9/15], Loss: 0.3630, Accuracy: 0.8699
Epoch [10/15], Loss: 0.3150, Accuracy: 0.8886
Epoch [11/15], Loss: 0.2663, Accuracy: 0.9090
Epoch [12/15], Loss: 0.2384, Accuracy: 0.9163
Epoch [13/15], Loss: 0.1925, Accuracy: 0.9350
Epoch [14/15], Loss: 0.1804, Accuracy: 0.9377
Epoch [15/15], Loss: 0.1631, Accuracy: 0.9451
Test Accuracy: 0.7689
Train Accuracy: 0.9645
Precision: 0.7677
Recall: 0.7689
F1 Score: 0.7659
ROC-AUC Score: 0.9144
```

The bottom status bar indicates the file encoding is UTF-8, 4 spaces, and Python 3.12 (project).

9. VISUALIZATION

- **Training Loss and Accuracy Plot** helps track the model's learning progress.
- **Bar Graph** of Actual vs Predicted Labels helps identify any imbalances or biases in the model's predictions.
- **Confusion Matrix Heatmap** provides a detailed breakdown of the model's performance for each class, highlighting strengths and weaknesses.

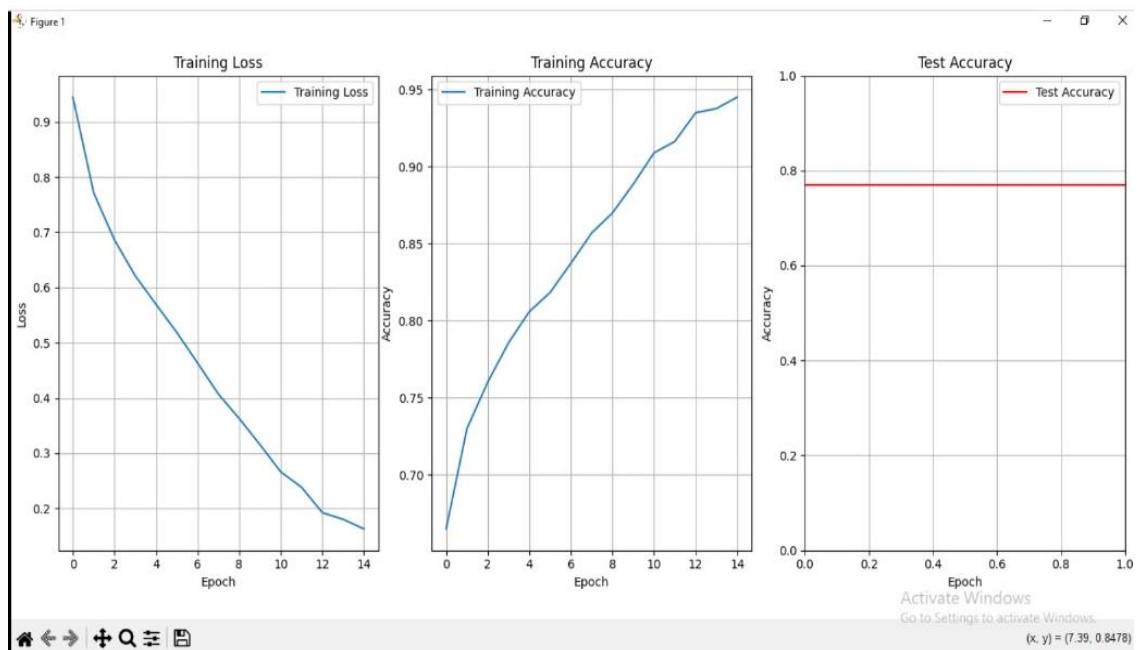
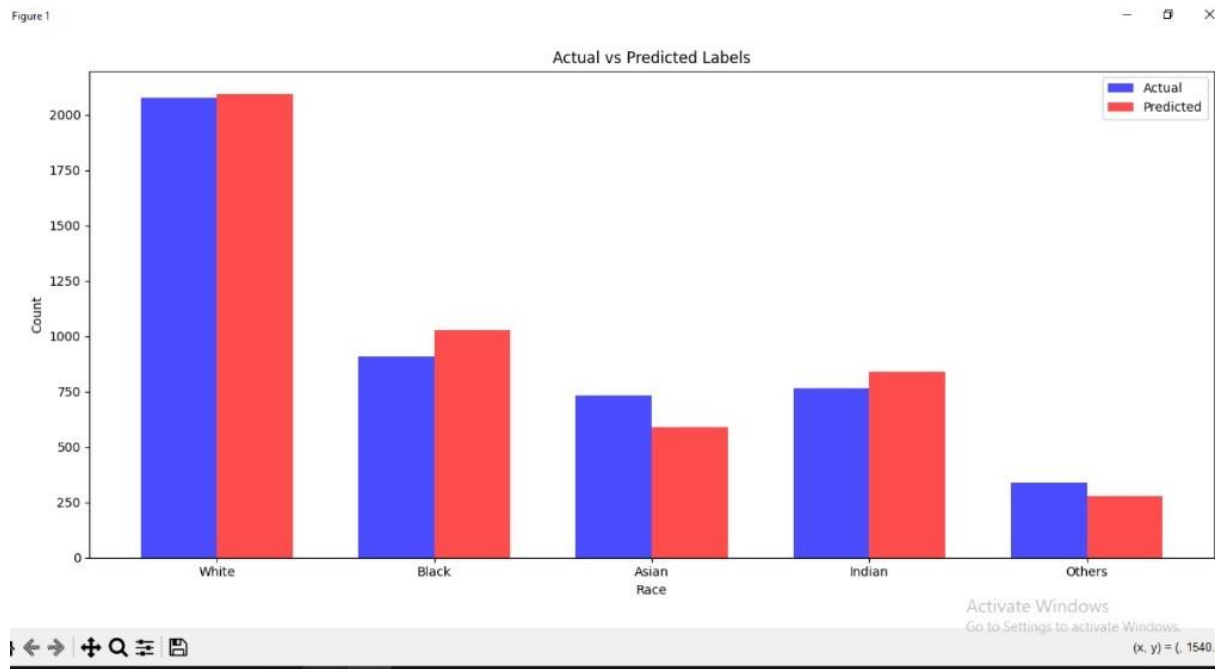


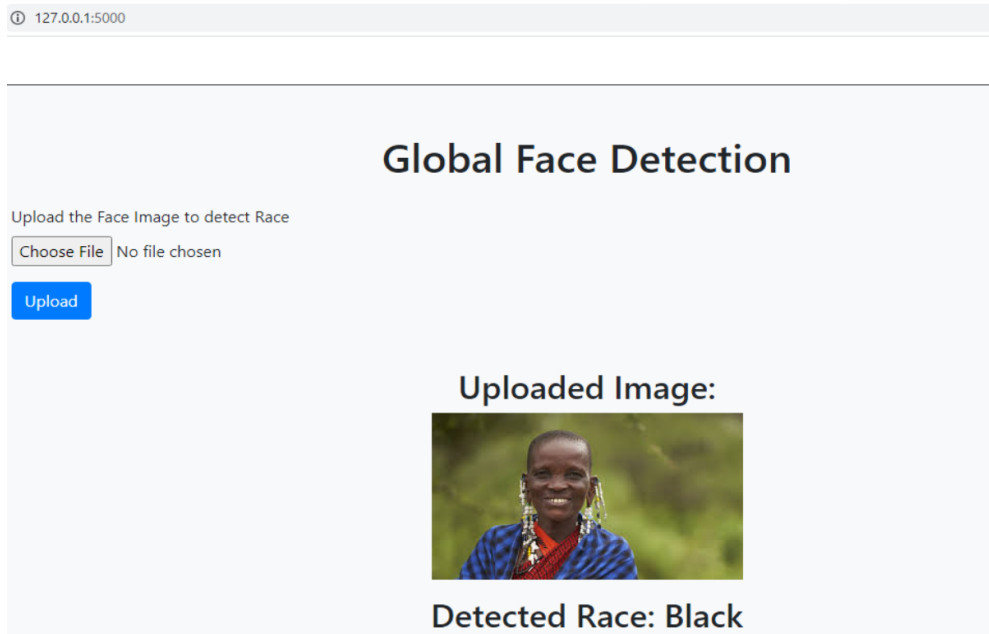
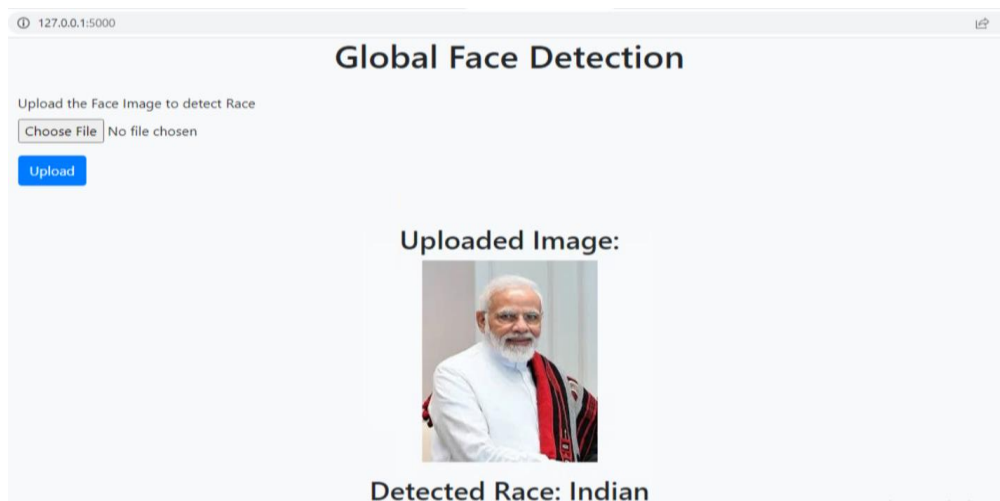
Figure 1

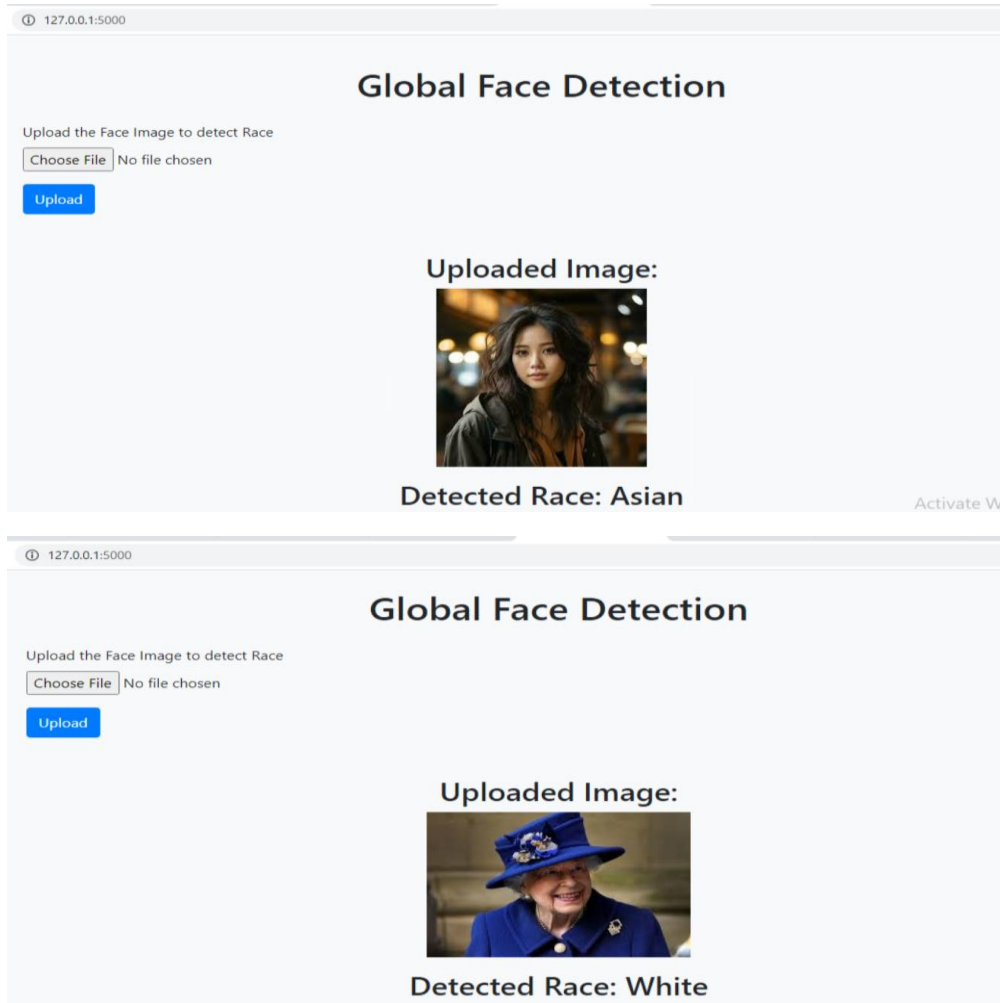


10. SAVE THE MODEL

The trained model was saved to disk using the `torch.save` function, allowing it to be loaded and used for future predictions without the need for retraining. The model was saved in the 'models' directory with the filename `utkface_model.pth`.

RESULTS





FUTURE SCOPE

1. Improvement in Accuracy and Robustness:

Future enhancements can focus on implementing more advanced architectures like ResNet-50, EfficientNet, or Vision Transformers to improve accuracy and robustness. Creating ensemble models combining multiple architectures could yield more reliable predictions. Continuous fine-tuning with diverse datasets will help the model perform better across various demographic groups.

2. Expansion of Dataset:

Expanding the dataset to include a broader range of ethnicities, age groups, and environmental conditions will improve the model's generalizability. Incorporating

real-world images from various scenarios will help the model handle diverse conditions, making it more effective in real-world applications.

3. Privacy and Ethical Considerations:

Addressing privacy and ethical concerns is essential for responsible deployment. Implementing privacy-preserving techniques like federated learning and differential privacy can protect user data. Ensuring continuous monitoring and mitigation of biases will make the model fairer and socially responsible.

CONCLUSION

The project successfully demonstrates the application of deep learning for race identification from facial images, achieving significant accuracy and providing a foundation for future research and practical applications.