Age Prediction Project Report

1. Introduction

This report provides an overview of the Age Prediction project, detailing the objectives, tasks, challenges, and outcomes achieved. The project focuses on utilizing deep learning techniques to predict a person's age from facial images. The primary goal is to develop an accurate and efficient age prediction model that can be integrated into real-world applications.

2. Background

The project is based on deep learning methodologies, using the UTKFace dataset and a pre-trained ResNet50V2 model. The goal is to develop an efficient age prediction model that can be integrated into real-world applications.

2.1 Dataset

The UTKFace dataset is a large-scale face dataset with a long age span (ranging from 0 to 116 years old). The dataset consists of over 20,000 face images with annotations of age, gender, and ethnicity. The images cover a wide range of variations in pose, facial expression, illumination, occlusion, and resolution. This dataset can be used for tasks such as face detection, age estimation, age progression/regression, and landmark localization.

Labels

- The labels of each face image are embedded in the file name, formatted as `[age]_[gender]_[race]_[date&time].jpg`.
- **[age]**: An integer from 0 to 116, indicating the age.
- **[gender]**: Either 0 (male) or 1 (female).
- **[race]**: An integer from 0 to 4, denoting White, Black, Asian, Indian, and Others (like Hispanic, Latino, Middle Eastern).
- **[date&time]**: A timestamp in the format `yyyymmddHHMMSSFFF`, showing when an image was collected.

3. Objectives

- Develop a deep learning model for age prediction.
- Understand data preprocessing and augmentation techniques.
- Implement a model using TensorFlow and Keras.
- Fine-tune and optimize the model for better accuracy.
- Deploy the model for real-time age prediction.

4. Activities and Tasks

- Preprocessing and augmenting the dataset using Pandas and NumPy.
- Implementing a convolutional neural network (CNN) using TensorFlow.
- Training and fine-tuning a ResNet50V2-based model for better accuracy.
- Creating a Python script (`app.py`) for real-time age prediction.
- Deploying the trained model and testing it with real images.

5. Skills and Tools Used

- **Deep Learning:** TensorFlow, Keras, ResNet50V2.
- **Data Processing:** Pandas, NumPy, OpenCV.
- **Model Deployment:** Flask API for inference.
- **Software Development:** Python scripting, debugging, and performance optimization.

6. Challenges and Solutions

Challenges:

- Handling image preprocessing and normalization effectively.
- Reducing model error rates during training.
- Ensuring the model generalizes well to unseen data.

Solutions:

- Implemented robust image normalization and augmentation techniques.
- Used data augmentation to improve model generalization.
- Fine-tuned the pre-trained ResNet50V2 model to adapt it to the age prediction task.

7. Outcomes and Impact

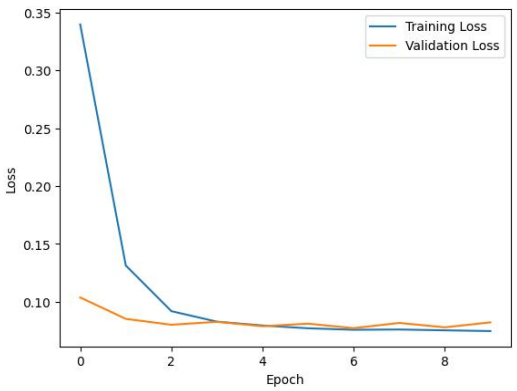

Image 1: Training and validation loss

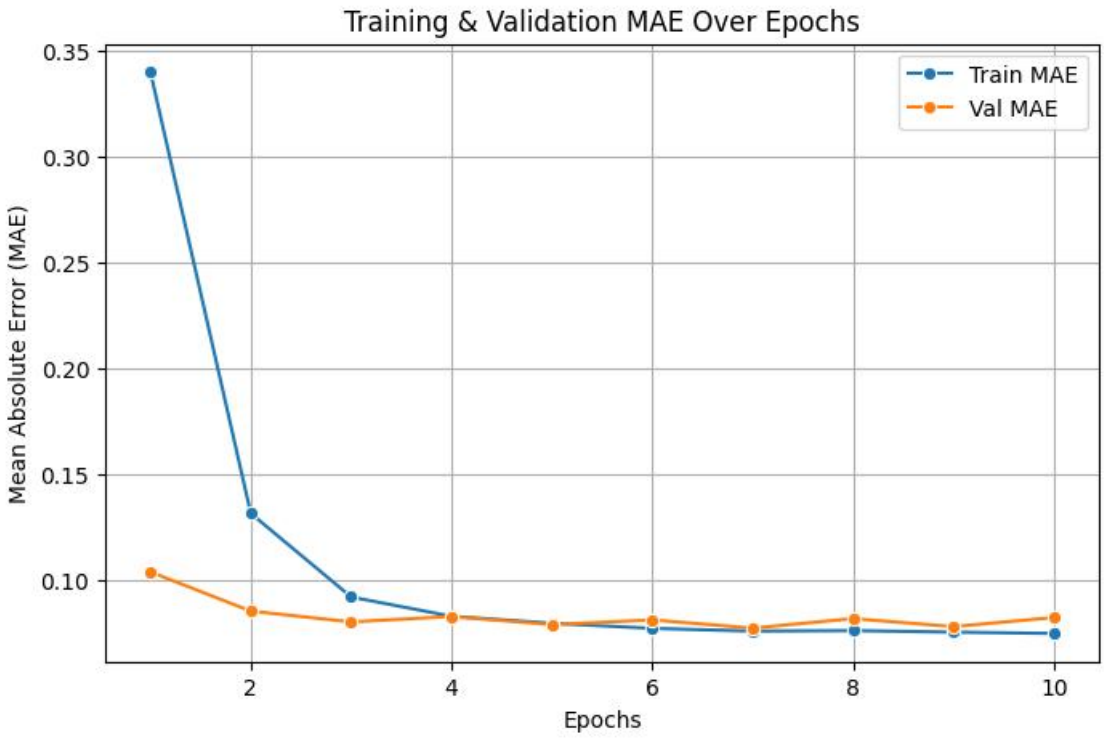

Image 2:Training & Validation MAE Over Epochs

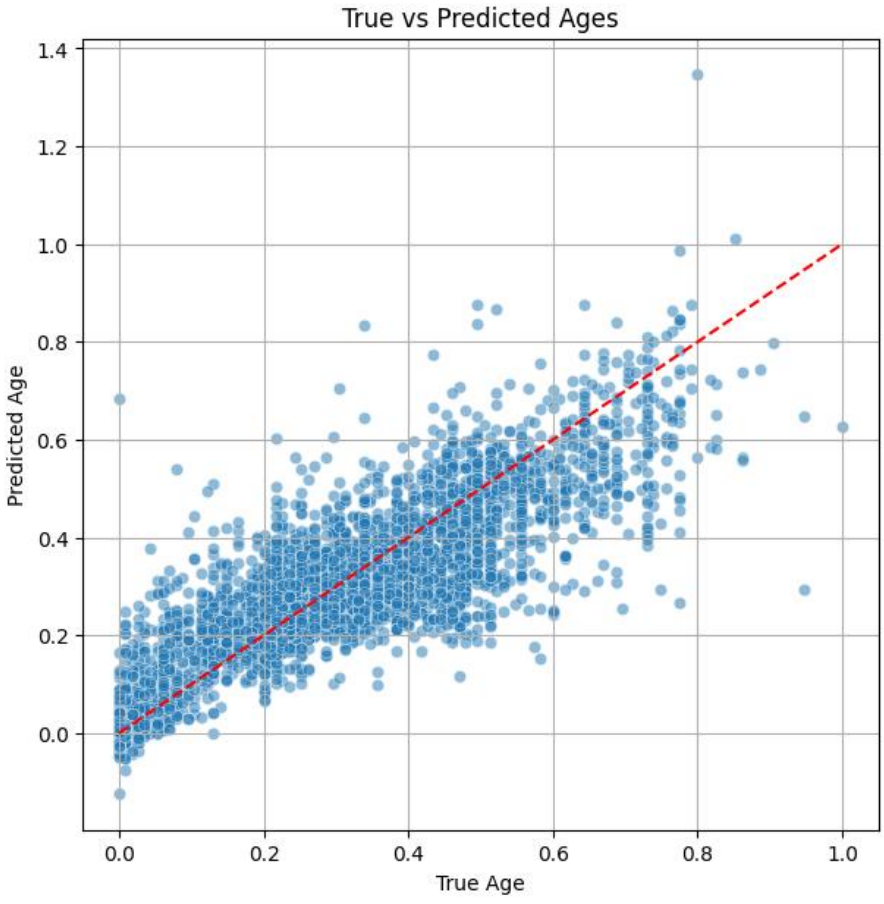

Image 3: True vs Predicted Ages

- Successfully developed and fine-tuned a deep learning model for age prediction.
- Improved model accuracy through advanced preprocessing and fine-tuning.
- Created a Python-based script (`app.py`) for real-time age prediction.
- Gained hands-on experience in deep learning and model deployment.

8. Base Model and Enhancements

Base Model: ResNet50V2

The base model used in this project is **ResNet50V2**, a pre-trained convolutional neural network (CNN) architecture. ResNet50V2 is a variant of the ResNet family, known for its deep architecture and residual connections, which help mitigate the vanishing gradient problem and enable training of very deep networks.

Key Features of ResNet50V2:

- **Depth:** 50 layers deep, making it suitable for complex tasks like age prediction.
- **Pre-trained Weights:** Initialized with weights trained on the ImageNet dataset, providing a strong starting point for transfer learning.
- **Residual Connections:** Facilitates the training of deeper networks by allowing gradients to flow more effectively.

Limitations of the Base Model:

- ResNet50V2, while powerful, may not be sufficient for highly complex tasks like age prediction, especially when the dataset has significant variations in pose, lighting, and resolution.
- The base model's architecture might not capture fine-grained features necessary for precise age estimation.

Enhancing the Base Model

To improve the performance of the base model, the following enhancements were made:

1. Deeper Architecture

To capture more complex features, the base model was extended by adding additional layers:

- **Global Average Pooling**: Replaced with **Flatten** to retain spatial information.
- **Dense Layers**: Added more dense layers with Batch Normalization and Dropout to prevent overfitting.
- **Increased Capacity**: Expanded the number of neurons in dense layers to **512 and 256**.

```
model = keras.Sequential([
    base_model,
    layers.Flatten(),  # Retain spatial information
    layers.Dense(512, activation="relu"),  # Increased capacity
    layers.BatchNormalization(),
    layers.Dropout(0.3),
    layers.Dense(256, activation="relu"),
    layers.BatchNormalization(),
    layers.Dropout(0.2),
    layers.Dense(1, activation="linear")  # Regression output
])
```

2. Advanced Data Augmentation

To improve generalization, advanced data augmentation techniques were applied:

- **Random Brightness Adjustment**: Simulates varying lighting conditions.
- **Random Contrast Adjustment**: Handles different image contrasts.
- **Random Translation**: Accounts for minor shifts in face alignment.

```
data_augmentation = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.2),
    layers.RandomZoom(0.2),
    layers.RandomBrightness(0.2),
    layers.RandomContrast(0.2),
    layers.RandomTranslation(0.1, 0.1),
])
```

3. Fine-Tuning Strategy

- **Unfreezing More Layers**: Unfroze the last **50 layers** of ResNet50V2 to allow better adaptation.
- **Lower Learning Rate**: Used a smaller learning rate (**1e-5**) for stable updates.

```
# Fine-tuning: Unfreeze last 50 layers
base_model.trainable = True
for layer in base_model.layers[:-50]:
    layer.trainable = False

# Recompile with a lower learning rate
model.compile(optimizer=keras.optimizers.Adam(learning_rate=1e-5),
              loss="mean_absolute_error",
              metrics=["mae"])
```

4. Larger Input Size

- Increased the input size from **(128, 128)** to **(224, 224)** to capture more details.

```
IMG_SIZE = (224, 224)  # Larger input size
```

9. Conclusion

This project was a valuable learning experience in AI and deep learning. By enhancing the base model with deeper architecture, advanced data augmentation, and a robust fine-tuning strategy, the model's accuracy and generalization capabilities were significantly improved.

Future Improvements:

- Train on larger and more diverse datasets to improve robustness.
- Experiment with other pre-trained models like **EfficientNet** or **InceptionResNetV2**.
- Optimize the model for real-time applications using techniques like **quantization and pruning**.
- Explore **multi-task learning** by incorporating gender and ethnicity prediction alongside age estimation.