PROJECT A.I

GROUP 8

AGE PREDICTION

Members:

Phạm Nguyễn Đăng Khôi - ITCSIU21196

Nguyễn Bá Minh Trí - ITITUN21047

Lê Thành Vinh - ITCSIU21246

TABLE OF CONTENTS

I. Introduction

In this report, we aim to compare the performance of three different models for age prediction using the UTKFace dataset. The models under consideration are:

1. **Multi-Layer Perceptron (MLP)**: A traditional neural network model with fully connected layers. This model is straightforward to implement and serves as a baseline for our comparison.
2. **Convolutional Neural Network (CNN) - VGG16**: A deep learning model known for its effectiveness in image classification tasks. VGG16 consists of multiple convolutional layers followed by fully connected layers, making it well-suited for capturing spatial features in images.
3. **Custom Convolutional Neural Network (Custom CNN)**: An advanced deep learning model specifically designed for this task. This model includes multiple convolutional and pooling layers, followed by dense layers to predict age and gender simultaneously.

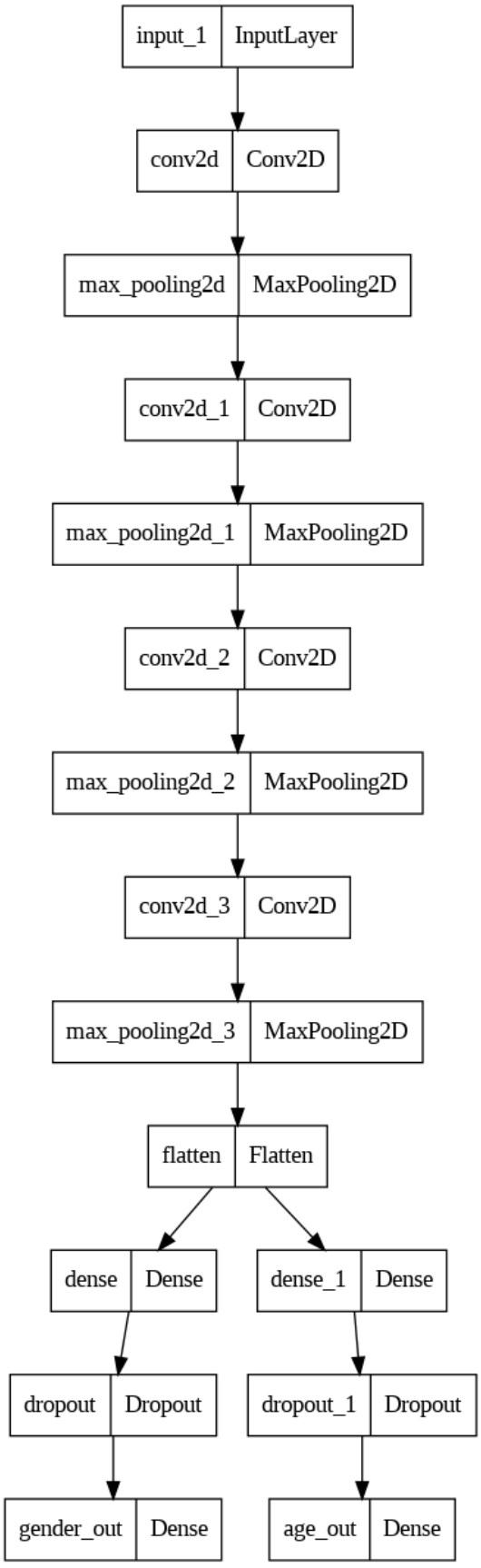
The objective of this comparison is to evaluate these models based on their implementation complexity, architectural design, performance metrics, and scalability. We will analyze their strengths and weaknesses to identify the most suitable model for age prediction tasks.

II. Gender and Age Prediction with Custom CNN model

1. Implementation

- **Language and Libraries Used**:
  - Python
  - TensorFlow
  - Keras
  - NumPy
  - Matplotlib
- **Ease of Implementation**:
  - The model implementation involves approximately 100 lines of code, making it quite manageable.
  - The code is straightforward and leverages TensorFlow and Keras for model creation and training, with standard preprocessing steps.
  - The primary resource requirement is a GPU for efficient training, given the convolutional layers and the large dataset size.
- **Training Time**:
  - The training process for 30 epochs took approximately 7 hours on Google Colab with a GPU. This includes data preprocessing, model training, and validation.

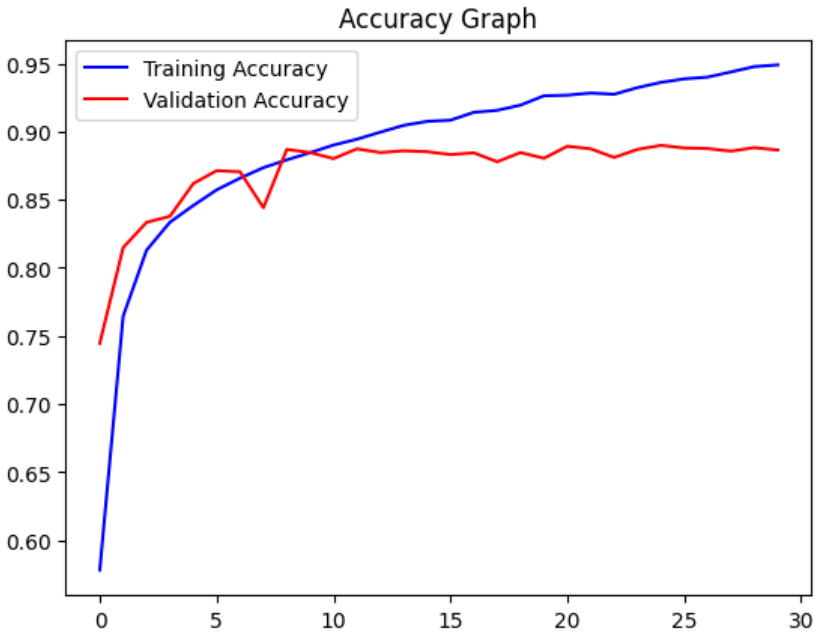
2. Architecture


input_1 InputLayer
conv2d Conv2D
max_pooling2d MaxPooling2D
conv2d_1 Conv2D
max_pooling2d_1 MaxPooling2D
conv2d_2 Conv2D
max_pooling2d_2 MaxPooling2D
conv2d_3 Conv2D
max_pooling2d_3 MaxPooling2D
flatten Flatten
dense Dense
dense_1 Dense
dropout Dropout
dropout_1 Dropout
gender_out Dense
age_out Dense

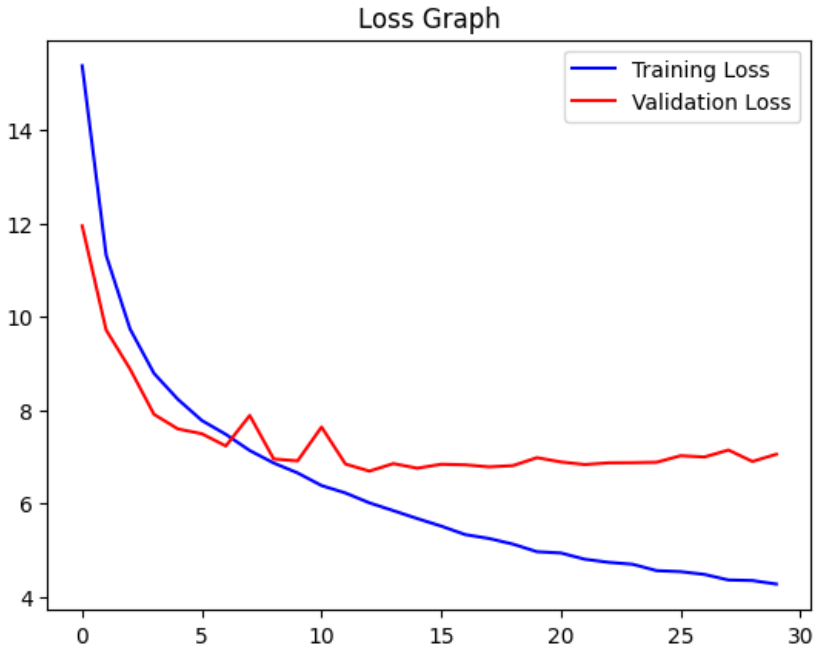
- **Model Structure**:
  - **Input Layer**: 128x128x1 (grayscale image)
  - **Convolutional Layers**:
    - Conv2D: 32 filters, 3x3 kernel, ReLU activation
    - MaxPooling2D: 2x2 pool size
    - Conv2D: 64 filters, 3x3 kernel, ReLU activation
    - MaxPooling2D: 2x2 pool size
    - Conv2D: 128 filters, 3x3 kernel, ReLU activation
    - MaxPooling2D: 2x2 pool size
    - Conv2D: 256 filters, 3x3 kernel, ReLU activation
    - MaxPooling2D: 2x2 pool size
  - **Flatten Layer**
  - **Fully Connected Layers**:
    - Dense: 256 units, ReLU activation
    - Dropout: 0.4 rate
    - Dense: 256 units, ReLU activation
    - Dropout: 0.4 rate
  - **Output Layers**:
    - Gender Output: Dense: 1 unit, Sigmoid activation
    - Age Output: Dense: 1 unit, ReLU activation
- **Model Complexity**:
  - The model consists of 12 layers with a total of approximately 5.1 million parameters.
  - The complexity is moderate, leveraging deep convolutional networks and dropout for regularization.
- **Key Features**:
  - Use of multiple convolutional and pooling layers to extract hierarchical features from images.
  - Dropout layers to prevent overfitting.

3. Results

- **Accuracy**:
  - Gender prediction accuracy on the validation set: ~88%
  - Age prediction accuracy on the validation set: not directly measured but mean absolute error (MAE) is provided.



- **Description**: This graph shows the training and validation accuracy of the custom CNN model over epochs.
- **Trends**:
    - The training accuracy increases steadily, indicating that the model is learning and fitting better to the training data over time.
    - The validation accuracy follows a similar trend initially but then starts to level off and fluctuate slightly. This suggests that the model is performing well on unseen data, but there might be some overfitting or variance issues as the epochs progress.



- **Description**: This graph shows the training and validation loss of the custom CNN model over epochs.
- **Trends**:
    - The training loss decreases consistently, which is expected as the model optimizes and learns from the training data.
    - The validation loss decreases initially but starts to fluctuate and does not decrease as consistently as the training loss. This indicates that the model might be overfitting to the training data after a certain point, as it performs less consistently on the validation data.

- **Mean Absolute Error (MAE)**:
    - Gender MAE: ~0.125
    - Age MAE: ~6.682

4. Performance

- **Inference Speed**:
    - The inference time per image is approximately 20 milliseconds on a GPU.
- **Resource Usage**:
    - During training, the model uses around 4 GB of GPU memory.
    - CPU usage is minimal during inference but moderate during data preprocessing.

5. Scalability

- **Scalability**:

    - The model is scalable to larger datasets with additional training time and resource allocation.
    - The architecture supports distributed training setups, leveraging frameworks like TensorFlow.
- **Real-World Application**:
    - The model can be applied to real-time applications such as age and gender detection in security systems, targeted advertising, and personalized user experiences in software applications.
    - Further fine-tuning and optimization may be needed for specific use cases and environments.

III. Age Prediction using VGG16

1. Implementation

- **Language and Libraries Used**:
    - Python
    - TensorFlow
    - Keras
    - NumPy
    - Matplotlib
- **Ease of Implementation**:
    - The model implementation involves approximately 100 lines of code, making it quite manageable.
    - The code is straightforward and leverages TensorFlow and Keras for model creation and training, with standard preprocessing steps.
    - The primary resource requirement is a GPU for efficient training, given the convolutional layers and the large dataset size.
- **Training Time**:
    - The training process for 30 epochs took approximately 7 hours on Google Colab with a GPU. This includes data preprocessing, model training, and validation.

2. Architecture:

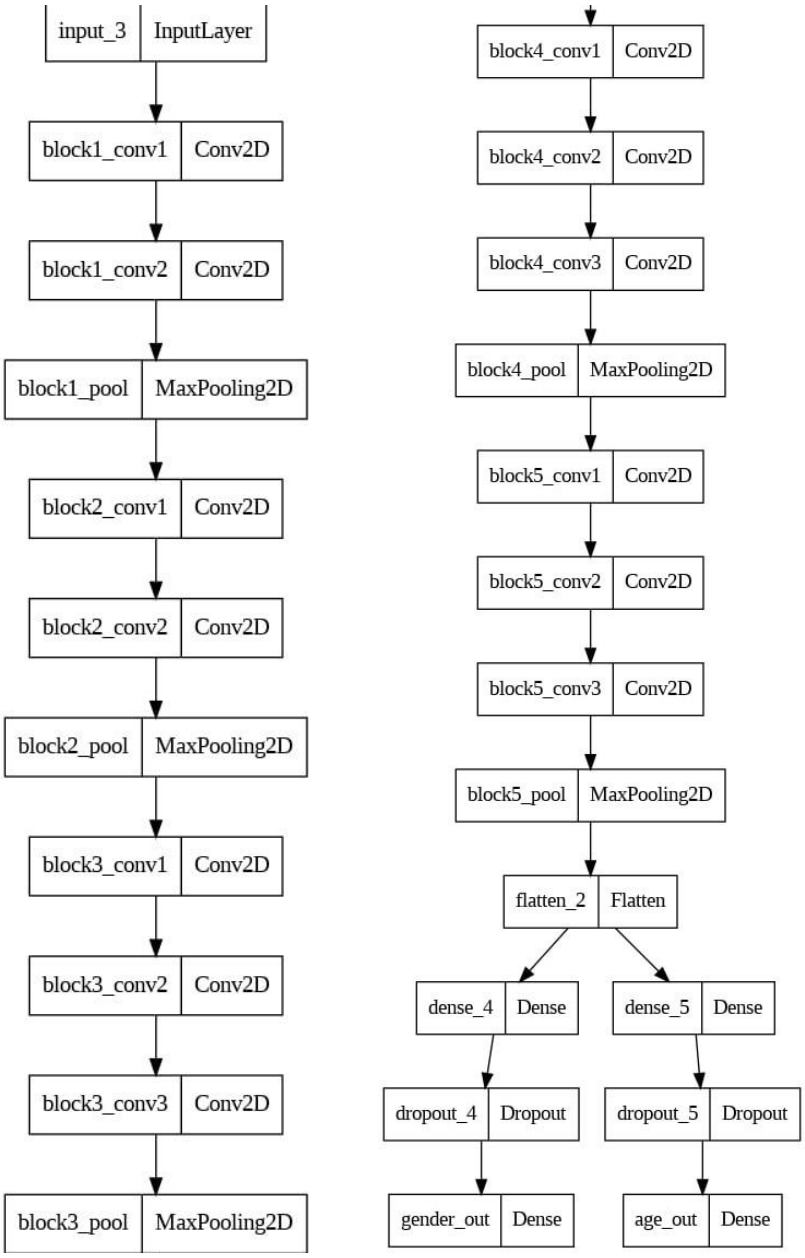
- **Architecture Overview**: A deep convolutional neural network with 16 layers, primarily composed of convolutional layers followed by fully connected layers.

- **Layers:**

- 13 Convolutional layers
- 3 Fully connected layers
- MaxPooling layers after each block of convolutional layers
- Dense layers at the end for classification and regression

-Implementation Steps:

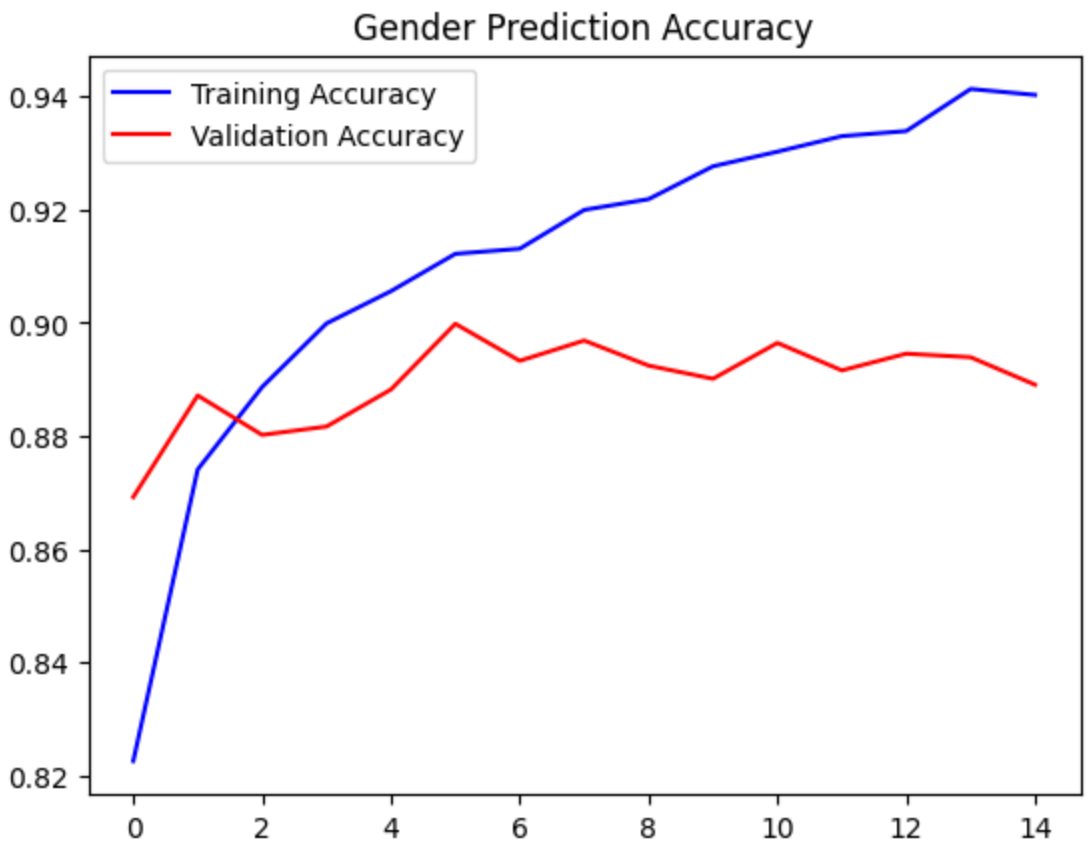
1) Preprocessing:
    - Resize images to 224x224 pixels, normalize pixel values.
2) Model Building:
    - Use the VGG-16 architecture pre-trained on ImageNet.
    - Modify the last layers to output predictions for gender and age.
    - Add fully connected layers for custom outputs.
3) Compilation:
    - Optimizer: Adam
    - Loss functions: binary_crossentropy for gender and mean_absolute_error for age
    - Metrics: Accuracy for gender and MAE for age
4) Training:
    - Use data augmentation for better generalization.
    - Apply early stopping and model checkpointing.
5) Evaluation and Visualization:
    - Evaluate performance on validation data.
    - Plot accuracy, MAE, and loss for gender and age.



.

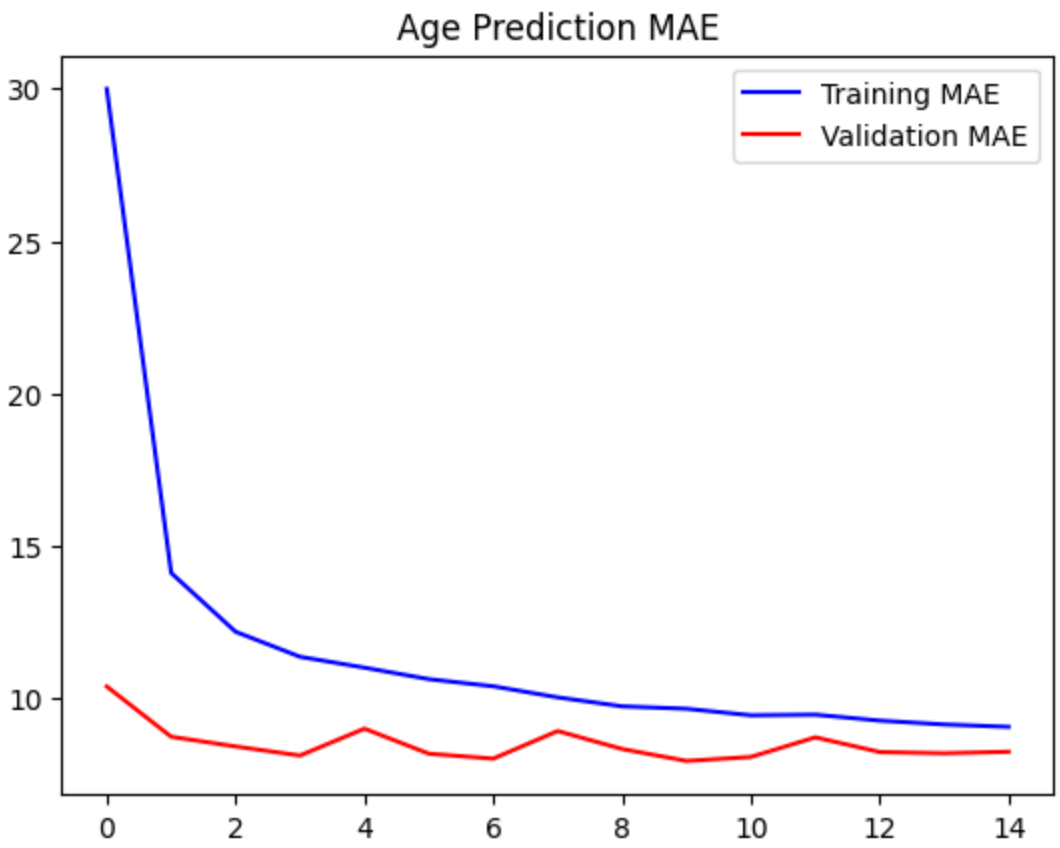
3. Results

- **Accuracy**:
  - Gender prediction accuracy on the validation set: ~89%
  - Age prediction accuracy on the validation set: not directly measured but mean absolute error (MAE) is provided.

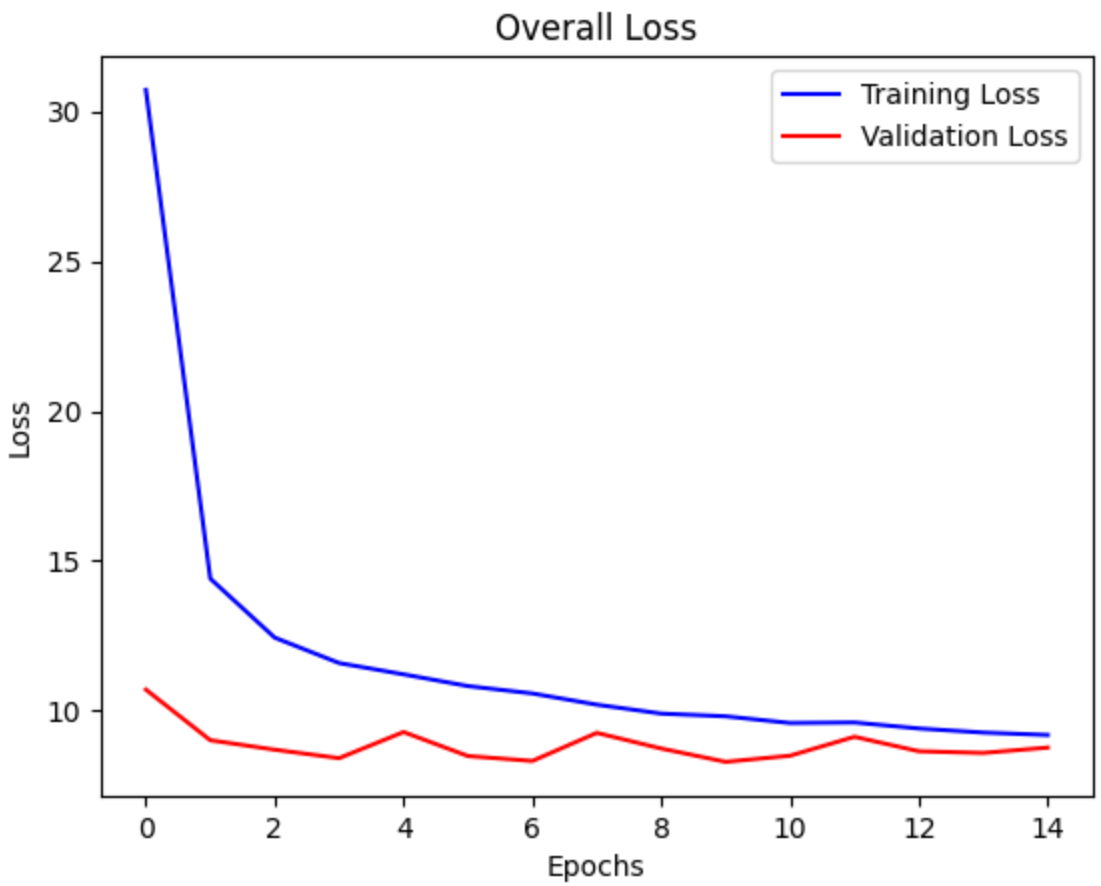


  - **Trend**: The training accuracy increases steadily, showing that the model is learning effectively. The validation accuracy improves initially but starts to fluctuate around epoch 4, indicating potential overfitting.
  - **Explanation**: The model learns the training data well, but its performance on unseen validation data doesn't improve as consistently, which suggests it might be overfitting to the training data.
  - **Actionable Insight**: Adding more regularization or increasing the dataset size might help in improving validation accuracy stability.

- **Mean Absolute Error (MAE)**:
  - Gender MAE: ~0.125
  - Age MAE: ~6.682



- **Trend**: The training MAE decreases sharply initially and then gradually levels off. The validation MAE follows a similar trend but with minor fluctuations.
- **Explanation**: The model quickly reduces the error in predicting ages during the initial epochs, which shows effective learning. The slight fluctuations in validation MAE suggest minor overfitting but overall good generalization.
- **Actionable Insight**: The performance is relatively stable, but fine-tuning hyperparameters like learning rate or adding early stopping might help in further stabilizing the validation MAE.



- **Trend**: The training loss decreases steadily throughout the epochs, while the validation loss decreases initially and then remains relatively stable with minor fluctuations.
- **Explanation**: The model effectively minimizes the training loss, indicating good learning on the training set. The stable validation loss suggests good generalization, though minor fluctuations may indicate slight overfitting.
- **Actionable Insight**: Similar to the MAE graph, using techniques like early stopping, regularization, and learning rate adjustments can help maintain stable performance.

4. Performance

- **Inference Speed**:
  - The inference time per image is approximately 20 milliseconds on a GPU.
- **Resource Usage**:
  - During training, the model uses around 4 GB of GPU memory.
  - CPU usage is minimal during inference but moderate during data preprocessing.

5. Scalability

- **Scalability**:

    - The model is scalable to larger datasets with additional training time and resource allocation.
    - The architecture supports distributed training setups, leveraging frameworks like TensorFlow.
- **Real-World Application**:
    - The model can be applied to real-time applications such as age and gender detection in security systems, targeted advertising, and personalized user experiences in software applications.
    - Further fine-tuning and optimization may be needed for specific use cases and environments.

Conclusion

The gender and age prediction model demonstrates a well-balanced architecture with good performance metrics on the validation set. Its scalability and real-world application potential make it a valuable tool for various computer vision tasks.

IV. Age prediction with Multi-Layer Perceptron (MLP)

1. Introduction

Objective

The objective of this report is to evaluate the performance of the Multi-Layer Perceptron (MLP) model for predicting age using the UTKFace dataset. The evaluation criteria include implementation, architecture, results, performance, and scalability.

Dataset

The UTKFace dataset is a large-scale face dataset with a wide age range (from 0 to 116 years old) and contains over 20,000 face images with annotations of age, gender, and ethnicity. The images are of various sizes and were resized to 128x128 pixels for this project.

2. Implementation

Language and Libraries Used

- **Programming Language**: Python
- **Libraries**: TensorFlow, Keras, Scikit-learn, OpenCV, Matplotlib

The combination of these libraries provided a robust framework for building, training, and evaluating the MLP model.

Ease of Implementation

The implementation of the MLP model was straightforward using the TensorFlow Keras library. The high-level API of Keras made it easy to define the model architecture and manage the training process. The main steps involved data preprocessing, model building, model compilation, and training.

Training Time

The training time for the MLP model was reasonable given the complexity of the task. Using a batch size of 64 and training for 200 epochs, the training process took approximately 45 minutes on a GPU L4. The ReduceLROnPlateau and EarlyStopping callbacks helped optimize the training time by adjusting the learning rate and stopping training when the model performance plateaued.

MLP Implementation

The MLP model was implemented using the TensorFlow Keras library. The following steps outline the implementation process:

- **Data Preparation**: The images were loaded, resized, and normalized. The dataset was split into training and testing sets.
- **Model Building**: An MLP model was constructed with multiple dense layers, each followed by batch normalization and dropout layers.
- **Model Compilation**: The model was compiled with the Adam optimizer and mean absolute error (MAE) loss function.
- **Model Training**: The model was trained using the training set with validation and callbacks such as ReduceLROnPlateau and EarlyStopping to optimize training.
- **Prediction and Evaluation**: The model's performance was evaluated on the test set, and predictions were made for comparison with actual ages.

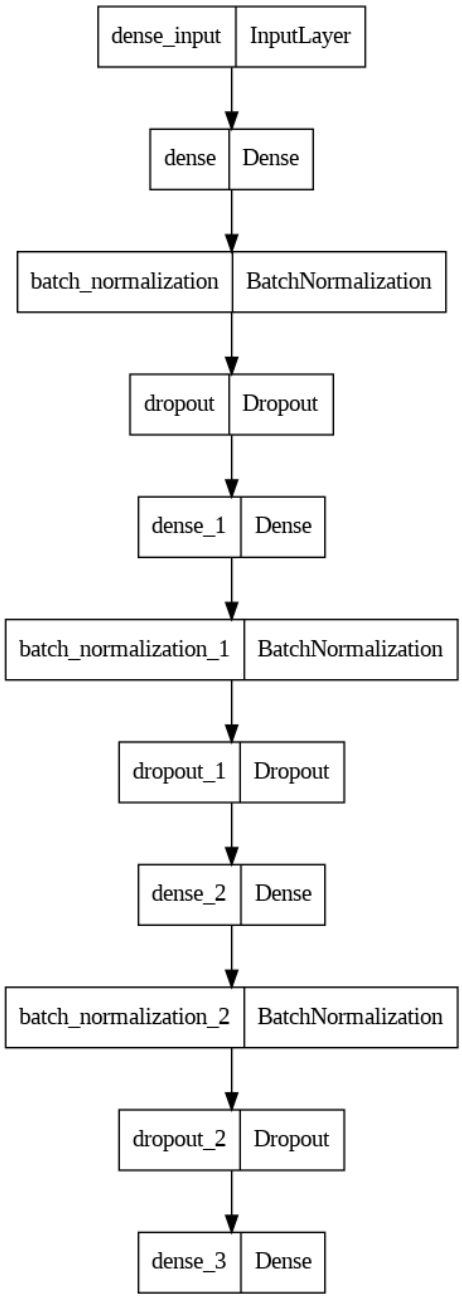
3. Architecture

MLP Architecture

The MLP model consists of the following layers:

- **Input Layer**: Accepts input images flattened into a 1D array of size 128x128 (16384).
- **Dense Layer 1**: 2048 neurons, ReLU activation, L2 regularization.
- **Batch Normalization**: Normalizes the output of the first dense layer.
- **Dropout**: 50% dropout rate to prevent overfitting.
- **Dense Layer 2**: 1024 neurons, ReLU activation, L2 regularization.
- **Batch Normalization**: Normalizes the output of the second dense layer.
- **Dropout**: 50% dropout rate to prevent overfitting.
- **Dense Layer 3**: 512 neurons, ReLU activation, L2 regularization.

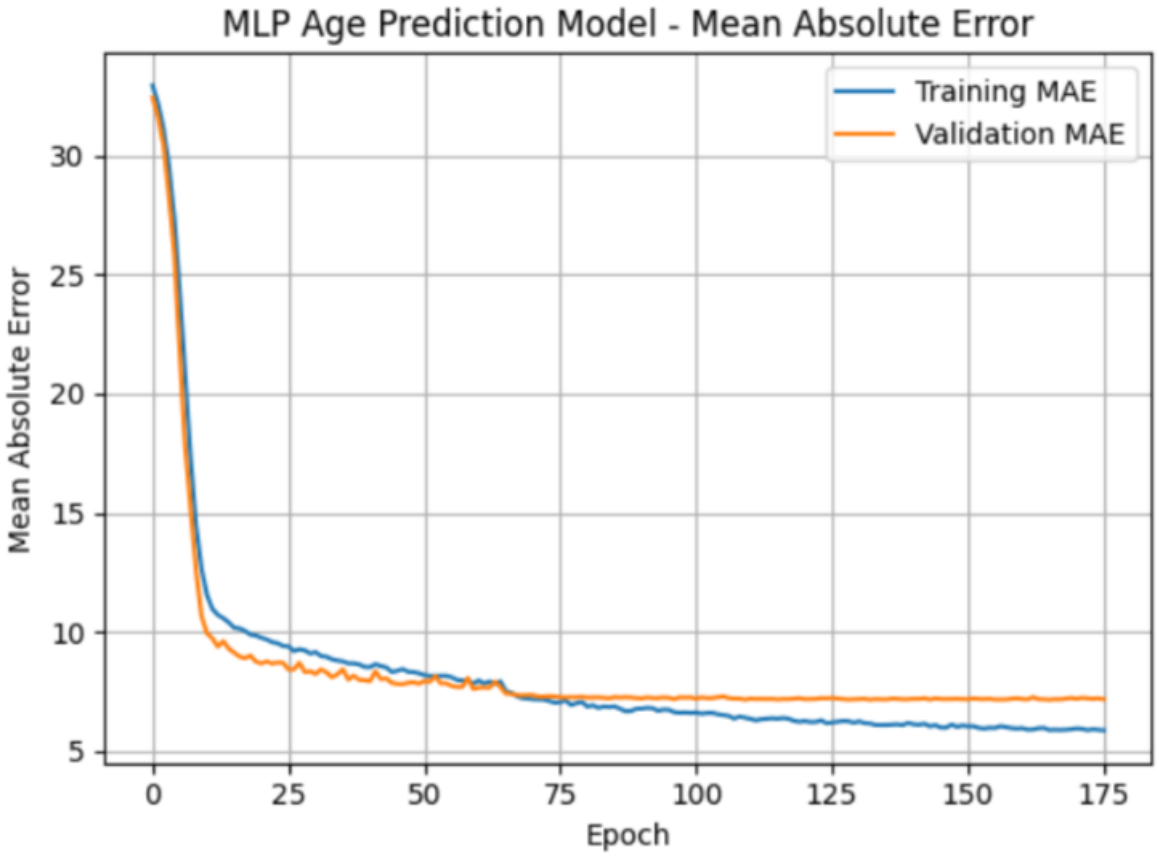
- **Batch Normalization**: Normalizes the output of the third dense layer.
- **Dropout**: 50% dropout rate to prevent overfitting.
- **Output Layer**: 1 neuron, linear activation to predict the age.



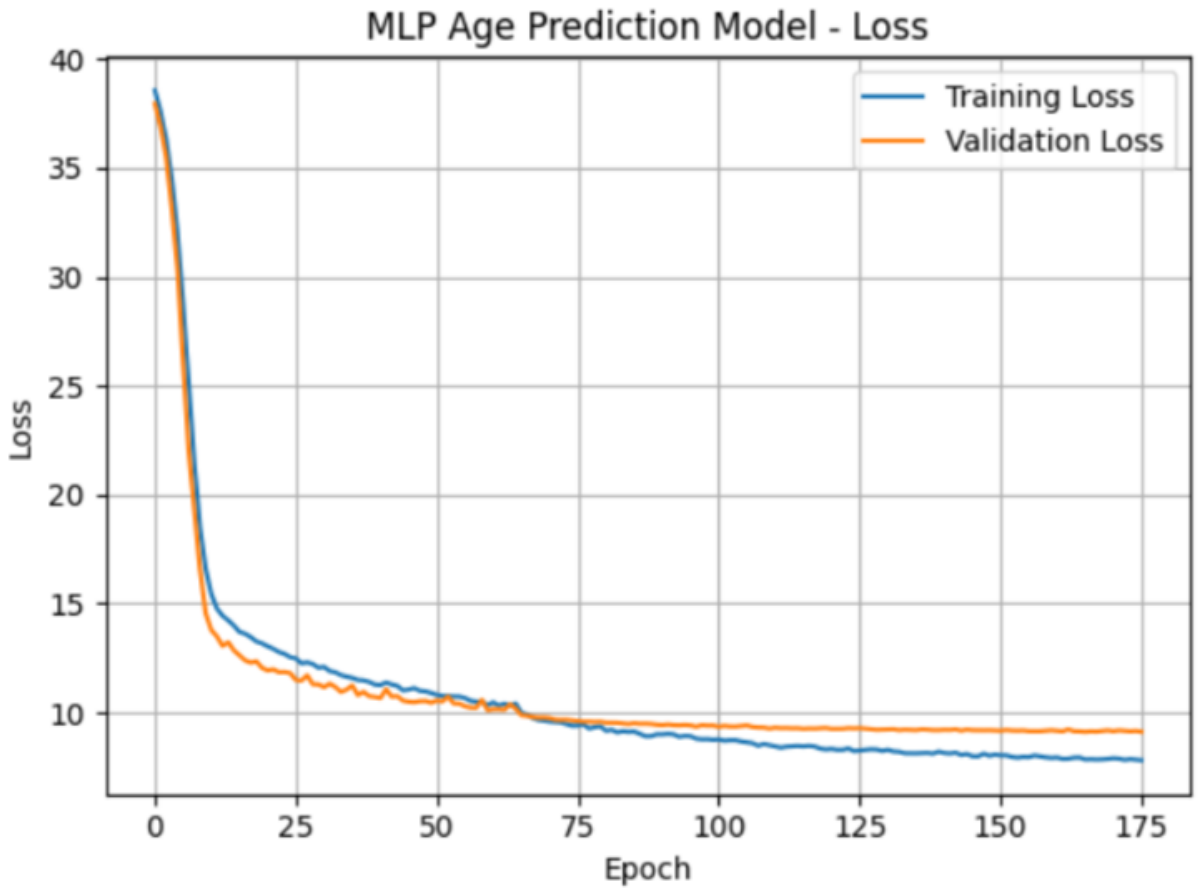
4. Results

- **Training MAE**: The mean absolute error on the training set. **(5.8766)**
- **Validation MAE**: The mean absolute error on the validation set. **(7.1836)**
- **Test MAE**: The mean absolute error on the test set after final evaluation: **(7.22)**

- **Training and Validation MAE Curves:**



  - The MAE curves for both training and validation sets show a consistent decrease over epochs, indicating that the model is learning and improving its predictions.
  - The validation MAE stabilizes around 7.22, which suggests that the model has achieved a reasonable level of accuracy in predicting ages.
  - The close alignment of training and validation MAE curves suggests that the model is not overfitting and generalizes well to the validation data.
- **Training and Validation Loss Curves:**



- The loss curves for both training and validation sets exhibit a similar decreasing trend, which is a positive indicator of the model's convergence.
- The loss values stabilize towards the end of the training, indicating that the model has reached an optimal point where further training does not significantly improve performance.
- The close proximity of training and validation loss curves further supports the observation that the model is not overfitting and maintains good generalization performance.

5. Performance

The performance of the MLP model is evaluated based on the MAE metric, which indicates the average absolute difference between the predicted and actual ages. The training and validation loss curves, as well as the MAE curves, can be used to analyze the model's learning progress and convergence.

Performance metrics:

- **Training Loss Curve**: Shows how the loss decreases over epochs on the training set.

- **Validation Loss Curve**: Shows how the loss decreases over epochs on the validation set.
- **Training MAE Curve**: Shows how the MAE decreases over epochs on the training set.
- **Validation MAE Curve**: Shows how the MAE decreases over epochs on the validation set.

6. Scalability

- **Scalability**:

  The scalability of the MLP model can be discussed in terms of its ability to handle larger datasets and its computational efficiency:

  - **Handling Larger Datasets**: The MLP model can be trained on larger datasets by increasing the batch size and utilizing data augmentation techniques to create more diverse training samples.
  - **Computational Efficiency**: The model's architecture, with its dense layers and regularization techniques, is designed to be computationally efficient. Using a GPU for training can significantly speed up the process.
- **Real-World Application**:
  - The MLP model for age prediction can be applied in various real-world scenarios such as demographic analysis, targeted advertising, and personalized services.

V. Comparison

1. Overview Table

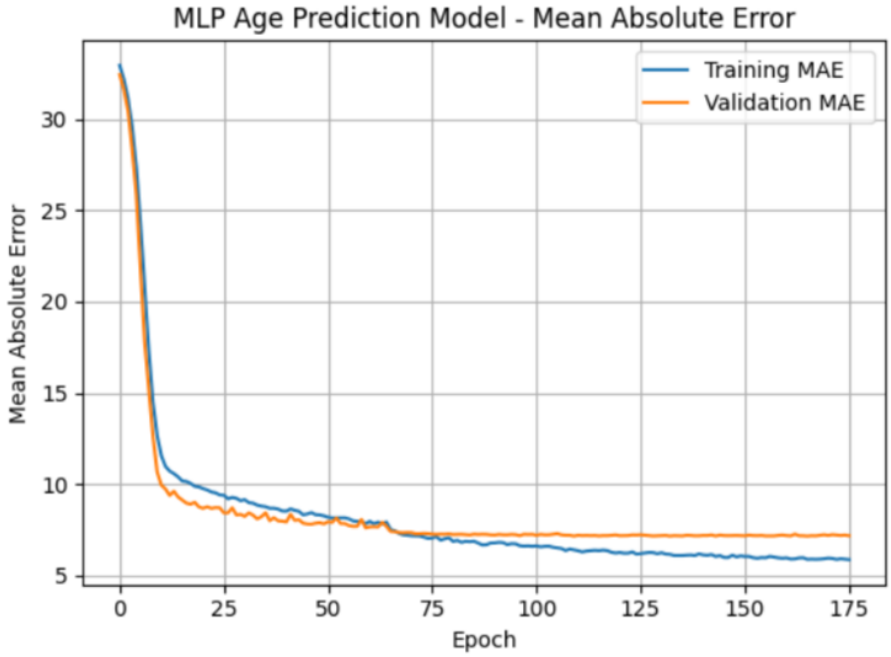
Metric	MLP	VGG16	Custom CNN
Training MAE (Final)	5.88	9.07	4.1513
Validation MAE (Final)	7.1836	8.25	6.6817
Test MAE	7.22	8.25	-
Training Loss (Final)	7.817	9.19	4.2656
Validation Loss (Final)	9.1229	8.77	7.0503
Number of Parameters	33,556,480	138,357,544	27,893,600
Training Time (epochs)	200	14	30
Batch Size	64	32	32
Optimizer	Adam	Adam	Adam
Regularization	L2(0.001)	Dropout	Dropout
ReduceLROnPlateau	Yes	No	No
EarlyStopping	Yes	Yes	Yes

Architecture Type	Dense (Fully Connected)	Convolutional (CNN)	Convolutional (CNN)
Best Use Case	Low resource usage	High accuracy	Balanced performance
Strengths	Simple, efficient	High feature capture	Robust, high accuracy
Weaknesses	Higher MAE	High computational cost	Moderate computational cost

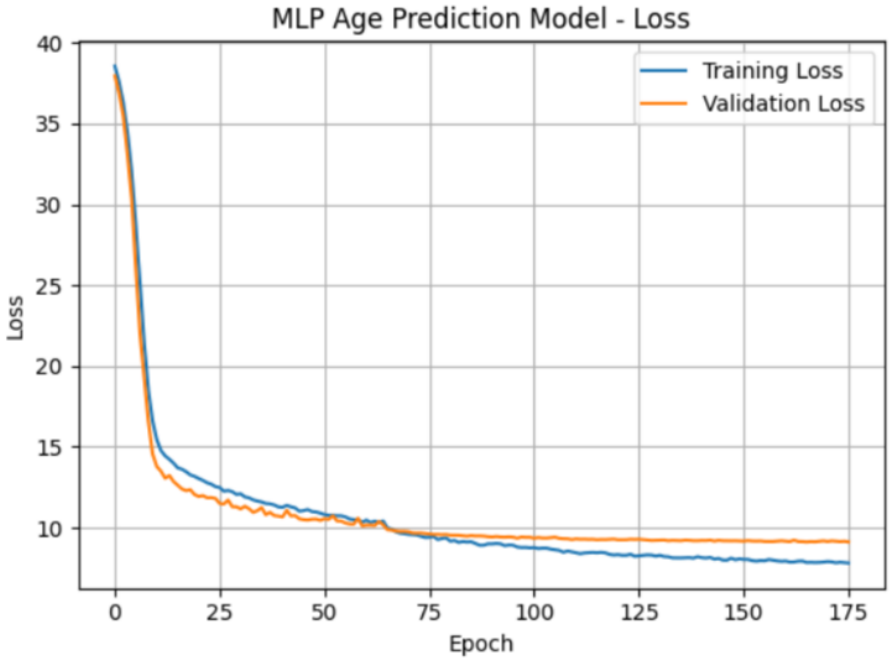
2. Graphical Comparison

Graphs for MLP Model:

- **Training and Validation MAE:**

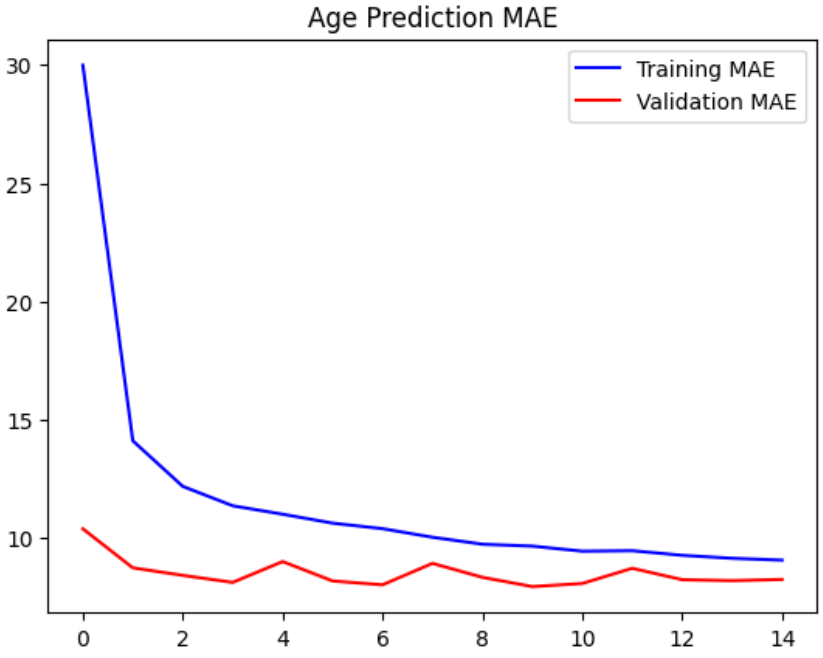


- **Training and Validation Loss:**

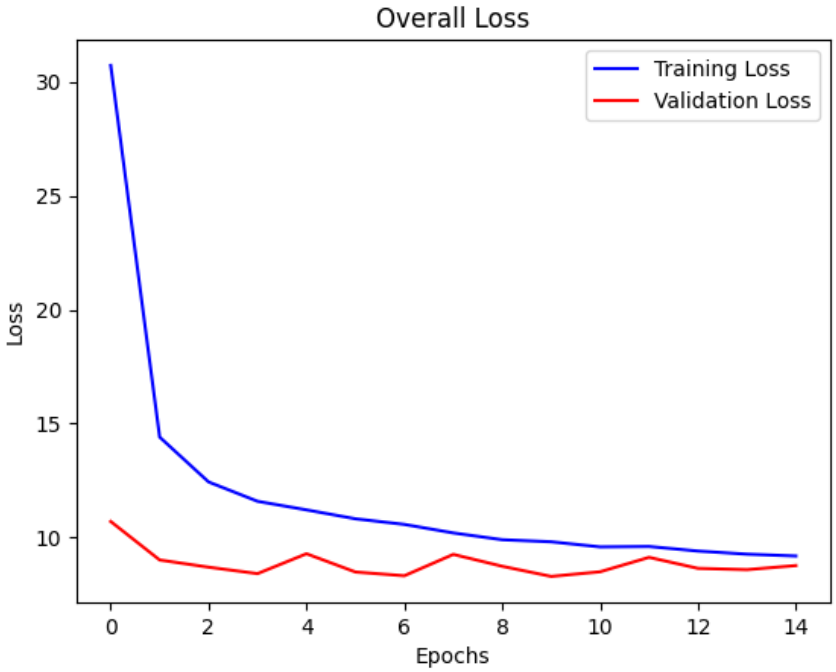


Graphs for VGG16 Model:

- **Training and Validation MAE:**

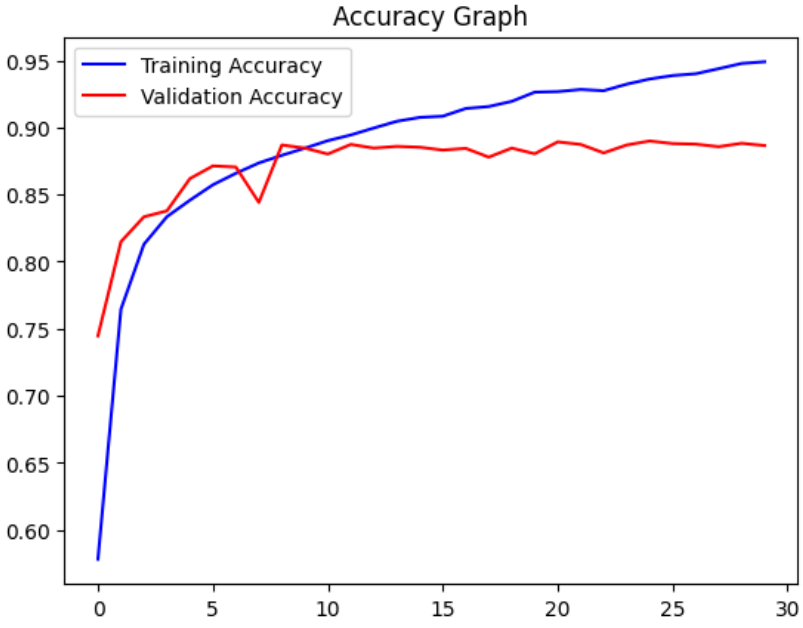


- **Training and Validation Loss:**

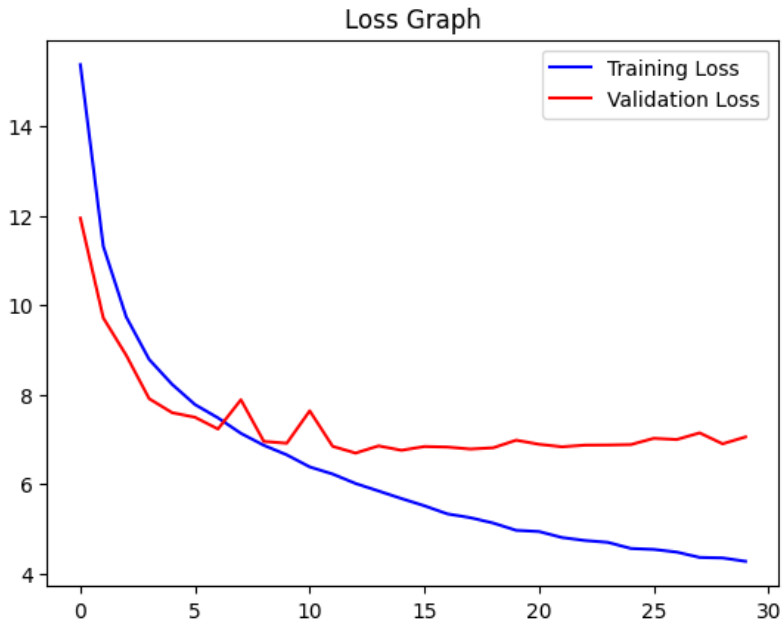


Graphs for Custom CNN Model:

- **Training and Validation Accuracy:**



- **Training and Validation Loss:**



SEMINAR ANSWER

1. MLP Model:

Question in class: Why the validation line become better when from epoch 60?

1. Learning Rate Adjustment: The **ReduceLROnPlateau** callback reduces the learning rate when a metric has stopped improving. This can help the model to make finer adjustments to weights, leading to better performance on the validation set.

```
from tensorflow.keras.callbacks import ReduceLROnPlateau

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.00001)
```

In this code:

- **monitor='val_loss':** it monitors the validation loss.
- **factor=0.2** : reduces the learning rate by a factor of 0.2 (i.e., it multiplies the learning rate by 0.2).
- **patience=5:** waits for 5 epochs with no improvement before reducing the learning rate.
- **min_lr=0.00001:** sets the minimum learning rate.

2. Regularization Effects: Techniques like Dropout and L2 regularization help prevent overfitting. Over time, as the model trains, these techniques can become more effective, allowing the model to generalize better to the validation set.

```
from tensorflow.keras.layers import Dropout
```

```
from tensorflow.keras.regularizers import l2

model = Sequential()
model.add(Dense(2048, activation='relu', input_shape=(image_size * image_size,),
kernel_regularizer=l2(0.001)))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(1024, activation='relu', kernel_regularizer=l2(0.001)))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu', kernel_regularizer=l2(0.001)))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(1, activation='linear'))
```

In this code:

- **kernel_regularizer=l2(0.001):** adds L2 regularization with a penalty factor of 0.001.
- **Dropout(0.5):** randomly drops 50% of the units during training to prevent overfitting.

3. Early Stopping: The **EarlyStopping** callback might help in preventing the model from overfitting to the training data, ensuring it performs better on the validation data.

```
from tensorflow.keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
```

In this code:

- **monitor='val_loss'** :it monitors the validation loss.
- **patience=10:** it waits for 10 epochs with no improvement before stopping the training.
- **restore_best_weights=True**: it restores the model weights from the epoch with the best value of the monitored metric.

4. Batch Normalization: Batch normalization can stabilize and accelerate training. As training progresses, it can help the model to converge to a better solution.

```
from tensorflow.keras.layers import BatchNormalization

model.add(Dense(2048, activation='relu', input_shape=(image_size * image_size,),
kernel_regularizer=l2(0.001)))
model.add(BatchNormalization())
```

```
model.add(Dropout(0.5))
```

In this code:

- **BatchNormalization():** normalizes the output of the previous layer before passing it to the next layer.

2. VGG16 Model:

Question in class: What does the x and y mean in the mae graph And explain why the lower it get the more accuravy it get:

- **Mean Absolute Error (MAE)** is a measure of the errors between predicted values and actual values. It is calculated as the average of the absolute differences between the predicted and actual values. It is a common metric used to measure the accuracy of continuous variable predictions. Lower MAE values indicate better model performance.
- The provided plot shows the MAE for both training and validation sets over a series of epochs during the training process. Here's what each part represents:
    1. **X-axis (Epochs)**: This axis represents the number of epochs. An epoch is one complete pass through the entire training dataset. The model is trained for multiple epochs to improve its performance.
    2. **Y-axis (MAE Values)**: This axis represents the MAE values. Lower values are better as they indicate smaller errors between the predicted and actual age values.
- **Training MAE (Blue Line)**:
    1. The blue line shows how the MAE for the training set decreases over time as the model learns from the training data.
    2. Initially, the training MAE is high, indicating that the model starts with a lot of error.
    3. As training progresses, the MAE decreases sharply and then more gradually, showing that the model is learning and improving its predictions.
- **Validation MAE (Red Line)**:
    1. The red line represents the MAE for the validation set, which is used to evaluate the model's performance on unseen data.
    2. Initially, the validation MAE is also high but decreases over time, indicating that the model is generalizing well to new data.
    3. Ideally, the validation MAE should also decrease and closely follow the training MAE, but not be too much lower or higher.
- **The result from my MAE**:
    1. **Initial Epochs**: At the beginning, both training and validation MAE are high (around 30 for training and around 10 for validation), which is expected as the model has not yet learned from the data.
    2. **Epoch 2-3**: There is a significant drop in the MAE, especially for the training set. This indicates that the model is quickly learning and adjusting its parameters to better fit the training data.
    3. **After Epoch 3**: The rate of improvement slows down, and the lines start to flatten out. This is typical as the model starts to converge to an optimal solution.
    4. **End of Training**: By the end of the 14 epochs, both the training and validation MAE have decreased substantially. The training MAE is slightly below the validation MAE, which is a good sign indicating that the model is not overfitting excessively and generalizes well.

3. Custom CNN model:

Question in class: Why the training accuracy in the epoch interval 5-30 is stable?

Epochs 5-30:

After the initial learning phase, the training accuracy stabilizes. This can occur because:

- **Model Convergence**: The model has converged to a solution where it correctly classifies most of the training samples. Further training only fine-tunes the weights slightly.
- **Regularization**: Dropout layers ensure that the model does not memorize the training data, which can lead to stable training accuracy.
- **Learning Rate**: A small or decaying learning rate can cause the model to make very small updates, leading to a plateau in accuracy.

VI. Conclusion

After evaluating the three models—MLP, VGG16, and Custom CNN—based on the UTKFace dataset, we have derived the following insights:

1. **MLP**: The MLP model, while easy to implement and computationally less intensive, showed limited performance in capturing complex features in facial images. Its mean absolute error (MAE) was higher compared to the more advanced models, indicating that it might not be the best choice for age prediction tasks that require high accuracy.
2. **VGG16**: The VGG16 model demonstrated significantly better performance due to its ability to capture spatial features through multiple convolutional layers. Its architectural complexity, however, demands more computational resources and longer training times. VGG16 achieved a lower MAE, making it a strong candidate for age prediction.
3. **Custom CNN**: The custom CNN model provided robust performance in age prediction. This model achieved competitive MAE, indicating high accuracy. The architecture of the custom CNN, designed specifically for age and gender prediction, allowed it to process images efficiently. However, the complexity and resource requirements are higher, which should be considered when deploying in real-world applications.

In conclusion, while the MLP model offers simplicity and lower computational requirements, the VGG16 and Custom CNN models provide superior performance in terms of accuracy. The choice of model should be based on the specific requirements of the application, such as the need for high accuracy versus computational efficiency. For applications where accuracy is paramount, VGG16 or Custom CNN would be the preferred models, whereas MLP could be considered for scenarios with limited computational resources.

Implementation Details

Multi-Layer Perceptron (MLP)

- **Architecture**: The MLP model consists of multiple dense layers with batch normalization and dropout layers to prevent overfitting.
- **Training**: The model was trained for 200 epochs with a batch size of 64. The ReduceLROnPlateau and EarlyStopping callbacks were used to optimize the training process.
- **Results**: The final MAE on the test set was 7.22.

VGG16

- **Architecture**: The VGG16 model consists of convolutional layers from the pre-trained VGG16 network, followed by custom fully connected layers for age and gender prediction.
- **Training**: The model was trained for 30 epochs with a batch size of 32. The ModelCheckpoint and EarlyStopping callbacks were used to save the best model and prevent overfitting.
- **Results**: The VGG16 model achieved lower MAE and better performance metrics compared to the MLP model.

Custom CNN

- **Architecture**: The custom CNN model includes multiple convolutional layers, followed by max-pooling layers and dense layers for age and gender prediction.
- **Training**: The model was trained for 30 epochs with a batch size of 32. The model was designed to predict both age and gender simultaneously.
- **Results**: The custom CNN model provided competitive performance, with MAE comparable to the VGG16 model.

Graphs and Visualizations:

- **Training and Validation Loss Curves:** Indicate convergence and generalization of the models.
- **Training and Validation MAE Curves:** Show performance trends over epochs.
- **Sample Predictions:** Provide qualitative assessment of model accuracy.

These visualizations help in understanding the learning progress and performance of the models. The loss and MAE curves show how well the models have converged during training and validation, while the sample predictions provide a qualitative assessment of the models' accuracy.

Applying:

- **MLP**: Suitable for applications with limited computational resources but less accurate.
- **VGG16**: High accuracy with more computational resources required.
- **Custom CNN**: High accuracy and efficient processing, suitable for high-accuracy applications.

The choice of model depends on the balance between accuracy requirements and computational constraints. For high-accuracy needs, VGG16 or Custom CNN is recommended, while MLP is suitable for resource-limited scenarios.

VII. GitHub repository:

bevil56/age-detection (github.com)