**Predicting Age from Facial Images Using Multi-Task Learning with ResNet50 Backbone**

**Introduction:**

Age estimation from facial images is a complex task with significant real-world applications. In this report, I propose an innovative approach to improve age prediction accuracy by leveraging multi-task learning and the ResNet50 neural network architecture. The task of predicting age from facial features is inherently intricate due to the diverse aging processes influenced by genetics and environmental factors. Our motivation for multi-task learning arises from the recognition that age estimation shares underlying facial features with gender and ethnicity prediction. By jointly training these tasks, I aim to enhance age prediction precision through shared feature learning.

My choice of the ResNet50 architecture as the backbone of our model stems from its effectiveness in computer vision tasks. I employ transfer learning to capitalize on the pre-trained weights from a model trained on a vast dataset, which aids my model in learning robust facial representations. By intertwining age, gender, and ethnicity predictions, I seek to unravel the intricate relationships between these attributes. This approach not only holds the potential to yield more accurate age predictions but also contributes to a deeper understanding of facial data's interconnected nature. In this report, I present the data preprocessing techniques, model architecture, and some related information.

**Data Preprocessing**

In this section, I present a comprehensive account of the data preprocessing steps carried out to prepare the dataset for training a machine learning model that predicts a person's age from facial images. These preprocessing steps are crucial to ensure data quality, consistency, and suitability for subsequent model development and training.

**Step 1: Data Visualization and Cleaning**

This step involves checking for corrupted images and missing values in the age column. Data visualization helps identify anomalies and inconsistencies in the dataset, ensuring the removal of problematic data points for cleaner and more reliable model training. In the project folder, there is a file named “Data\_viz” where I visualize the data to help me understand the data and how to handle the task. Moreover, there is another file ‘data cleaning’ which I used to remove the corrupted images and find out if there are missing ages in the age column. These techniques helped me find the solution to the task.

**Step 2: Load the Dataset**

Reason: The first formal preprocessing step involved loading the dataset from the provided CSV file. This file contains vital information about facial images, including pixel values, age, gender, and ethnicity labels. Loading the data is the initial step to make it accessible for further processing. I used Pandas library to load the data.

Df= pd. read\_csv('/content/drive/MyDrive/TEZDA\_TASK/Dataset/age\_gender.csv')

**Step 3: Define a Function to Convert Pixel Values to Image Arrays**

Reason: To facilitate the conversion of raw pixel values into recognizable image data, a custom Python function was crafted. This function simplifies the process of converting pixel data into usable image arrays, which are essential for model input.

**Step 4: Split Space-Separated Pixel Values and Convert to NumPy Arrays**

Reason: The pixel values in the dataset were initially represented as space-separated strings. To effectively work with this data, these values were parsed and converted into NumPy arrays. This transformation was essential to prepare the data for subsequent analysis and model training.

**Step 5: Reshape the Array to the Appropriate Image Dimensions**

Reason: Ensuring data consistency with the actual image dimensions is crucial. The NumPy arrays were reshaped to match the original image size (48x48 pixels for a 2D image), ensuring that the data aligns with the expected format for image-based models.

**Step 6: Normalize Pixel Values to [0, 1] Range**

I also consider normalizing the pixel values within [0,1] that is because proper data Scaling the pixel values to fall within the range [0, 1] standardizes the data, enhancing model convergence during training and preventing issues related to different data scales. normalization is critical for machine learning models.

**Step 7: Load and Preprocess Images, Extracting Images and Age Labels**

The method was created to extract the images and their labels. This step involved loading and preprocessing the images, extracting both the images themselves and the corresponding age, gender, and ethnicity labels from the CSV file. Extracting relevant information from the dataset lays the foundation for establishing the relationship between image data and their labels.

**Step 8: Convert Pixel Values to Images**

Building upon the pixel-to-image conversion function defined earlier, the processed pixel data was transformed into recognizable images. This transformation was crucial for preparing the data for input into the machine learning model, which expects images as input.

**Step 9: Converting Images to RGB**

Reason: The chosen model architecture, ResNet50, necessitates input images in RGB format. Therefore, the images in the training, validation, and test sets were converted to RGB format to align with the model's input requirements.

**Step 10: Split the Data into Training, Validation**

Effective model training and evaluation require data partitioning. The dataset was divided into three distinct subsets: training, validation, and testing. This division allows for separate phases of model development, fine-tuning, and assessment, ensuring robustness and generalization.

**Step 11: Define Data Augmentation Parameters**

Reason: Augmenting the training dataset is a powerful technique for enhancing model robustness. Carefully defined data augmentation parameters, including rotations, flips, and zooms, diversify the training data, enabling the model to generalize better from the available data.

**Step 12: Create Data Generators for Training Data:** To efficiently manage and feed data into the model during training and evaluation, custom data generators were established for the training. These generators handle essential tasks such as data augmentation, batch processing, and other critical functions, ensuring seamless integration of the dataset with the machine learning model.

**Model Development Documentation**

In this section, I provide detailed documentation of the model development process for age prediction from facial images. Each step is explained, and the reasons for specific choices are highlighted to justify their necessity in achieving the project's objectives. As mentioned in the introduction, the task was given to predict the ages, predicting ages has a strong relationship with the gender and ethnicity of individuals. Since the data given contains labels of all ages, gender, and ethnicity I chose to use a multi-task model that predicts age, gender, and ethnicity to help the model learn the feature relationships between them, and could help in boosting the performance of age detection.

Furthermore, the multi-task model contains both regression and classification because based on the data given ages are continuous data which is a regression problem, while gender and ethnicity are classification. Based on this the model is designed to do regression on ages, binary classification on gender, and multi-class classification in ethnicity.

**1. Model Architecture:**

Input images

(48, 48, 3)

ResNet50

Backbone

Dense Layers

(512)

Age Head

Gender Head

Ethnicity Head

1 Output

1 Output

5 Outputs

Leaky ReLU

Leaky ReLU

Leaky ReLU

softmax

Sigmoid

linear

**Figure 1: Model Architecture**

For this project, I have adopted the Residual Network (ResNet50) architecture as the foundation for my multi-task model. That is because Resnet50 has the following advantages compared to other models like VGG16;

* The selection of ResNet50 is grounded in its proven effectiveness in handling deep neural networks, particularly in computer vision tasks like age prediction. Key reasons for this choice include:
* Residual Connections: ResNet50 addresses the vanishing gradient problem through residual connections, allowing for the training of very deep networks. This architectural innovation contributes to the model's ability to capture intricate features in facial images.
* Skip Connections: ResNet50 incorporates skip connections, which enable smoother gradient flow and enhance training efficiency. These connections facilitate the flow of information across layers, enabling the model to learn hierarchical representations.

**Model Definition:**

The model is constructed by defining its inputs and outputs. The backbone of ResNet50 serves as the input, and the age head layer is the output. This organization allows us to compile and train the model efficiently. The layers of the ResNet50 backbone are explicitly frozen in our implementation. Freezing the layers is a standard practice in transfer learning and is crucial for our project for the following reason: Leveraging Pre-trained Weights: By freezing the layers, we retain the pre-trained weights and feature extraction capabilities of ResNet50, which were originally trained on a large dataset (e.g., ImageNet). This allows us to benefit from the knowledge encoded in these layers without the risk of overfitting to our specific task.

**3. Modify the Final Layer to Match the Number of Age Classes:**

The final layer of the pre-trained ResNet50 model is not suitable for our multi-task, which requires a single numerical output for ages, and genders and five outputs for ethnicity. Therefore, we need to replace the final layer to match the number of age classes, which is one in our case. The three branches for age estimation, ethnicity classification, and gender classification, each consisting of flattened, Dense, Batch Normalization, and Dropout layers.

**Flatten Layer:** The Flatten layer is applied to the output of the shared feature extraction backbone. It transforms the spatial dimensions of the feature maps into a flat vector. This is necessary when transitioning from convolutional layers to fully connected layers.

**Dense Layer with Leaky ReLU Activation:** The Dense layer with 512 units and Leaky ReLU activation is applied to the flattened feature vector. Leaky ReLU is used as the activation function because it allows a small gradient for negative values, which can help mitigate the vanishing gradient problem and speed up training. The dense layer with non-linear activation helps the model learn complex patterns and representations from the data.

**Batch Normalization Layer:** The Batch Normalization layer is used after the Leaky ReLU activation. Batch normalization normalizes the activations of the previous layer, making the training process more stable and accelerating convergence. It reduces internal covariate shifts, which can help the model generalize better and train faster. Batch normalization also acts as a regularizer, reducing the need for excessive dropout.

**Dropout Layer:** The Dropout layer with a dropout rate of 0.3 is applied after batch normalization. Dropout randomly sets a fraction of the input units to zero during each training batch. This technique helps prevent overfitting by introducing randomness and reducing the interdependence of neurons. It encourages the network to learn more robust features.

**Output Layer:** Finally, a Dense layer with linear activation is used as the output layer for the age branch. It outputs a single numerical value for age estimation, making it suitable for regression tasks.

Therefore, the same architectural pattern is repeated for the ethnicity and gender branches with slight variations in the number of units in the dense layers and the activation functions, because of the nature of the problem.

**Loss Weight:**

Setting the loss high weight of 0.6, 0.5, and 0.5 for age, ethnicity, and gender respectively indicates equal importance to the age prediction task during training. The chosen metric, Mean Absolute Error, is a suitable choice for regression tasks like age prediction. This loss function measures the absolute difference between predicted and actual ages, aligning with our objective to predict age as accurately as possible.

**Optimizer Selection (Adam):**

The Adam optimizer with a learning rate of 1e-4 is selected. Adam is known for its efficiency in training deep neural networks. The choice of a learning rate helps stabilize training, preventing the model from diverging during fine-tuning.

**Model Training Process Documentation**

In this section, I provide comprehensive documentation of the model training process for age prediction from facial images. I outline the key steps, including the training and validation plots, sample predictions on test images, and hyperparameter tuning.

**1. Model Training and Validation:**

The model was trained using the training dataset, which was pre-processed and prepared as previously described. During training, both the training and validation datasets were utilized to monitor the model's performance.

The training was performed over a sixty (60) number of epochs, where each epoch represents a complete pass through the training dataset with batch size 32.

The model was compiled with the chosen loss function 'Mean Absolute Error ' for age because is a regression problem, ‘Binary cross-entropy for gender because is a binary classification, and ‘categorical cross-entropy for ethnicities because is a multi-class classification. optimizer (Adam with a learning rate of 1e-4) after tuning the learning rate until get performance required.

Training and validation loss curves were generated to visualize the model's performance over epochs. These plots show how the loss decreases during training while ensuring that the model doesn't overfit the data.

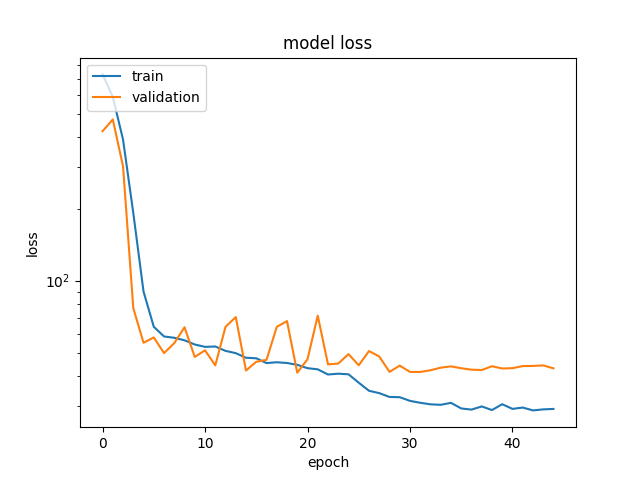
**2. Training and Validation Loss:**

Training and validation loss curves are plotted to provide insights into the model's performance. The x-axis represents the number of training epochs, while the y-axis displays the loss values.

The training loss curve shows how the model's loss decreases over time as it learns from the training data. A smooth and consistent decrease indicates effective learning.

The validation loss curve indicates how the model generalizes to unseen data. It should ideally follow a similar decreasing trend to the training loss curve without diverging, suggesting that the model isn't overfitting.

**Figure 2: Training and Validation Loss**

****

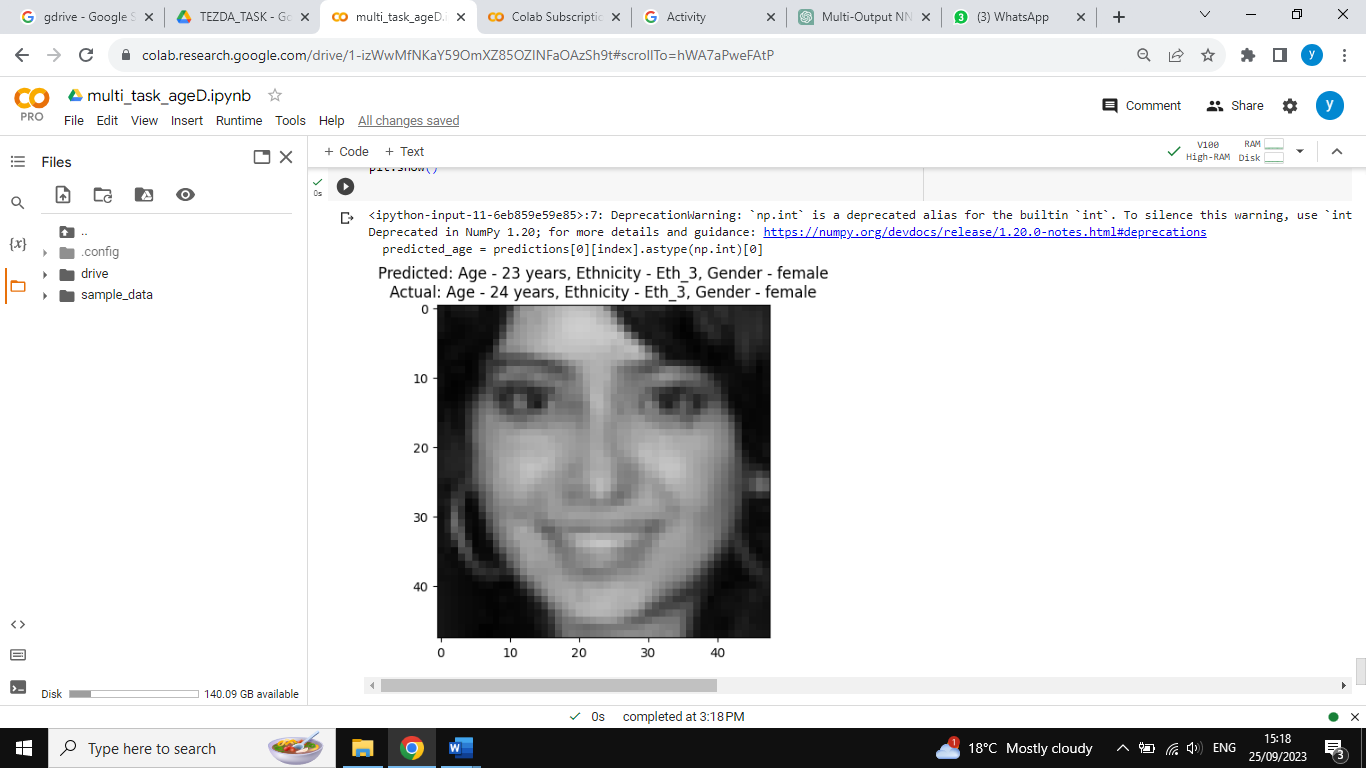
**3. Sample Predictions on Validation Images:**

To assess the model's ability to predict ages accurately, a set of validation images was used.

The trained model was employed to predict the ages of individuals in the validation images.

Sample predictions, along with the corresponding actual age, gender, and ethnicity were generated to evaluate the model's performance qualitatively.

**Figure II: Sample Predictions on Test Images:**



**Hyperparameter Tuning:**

Hyperparameter tuning is a critical step in optimizing the model's performance.

During this process, different hyperparameters such as learning rate, batch size, and the number of training epochs are systematically adjusted and evaluated.

Various combinations of hyperparameters were explored to find the configuration that resulted in the best validation performance.

The hyperparameter tuning process aimed to strike a balance between training efficiency and achieving the desired accuracy on the validation dataset.

**Handling Accuracy Metric in Regression:**

The task was given to evaluate the model with evaluation data and get 70%. However, In the context of my age prediction task, traditional accuracy metrics, designed for classification problems, are not directly applicable due to the regression nature of the problem. Our objective is to predict a person's age as a continuous numerical value, which doesn't align with the standard definition of accuracy.

However, to sort this out, I reviewed some age detection papers to see how the challenge was handled by researchers. I came across two different methods:

1. I devised a unique approach for evaluating the model's performance. The core idea was to categorize both the model predictions and the ground truth ages into predefined age groups or bins. These age groups are thoughtfully defined based on practical age ranges: "child," "Adult," "Senior adult," and "Old man.". Then compare the actual age groups and the predicted age groups to calculate the accuracy based on the groups [1]

After successfully applying this method, I get 93% Accuracy.

1. I use a certain number as a criterion to measure the accuracy between the actual ages and the ground truth. The method was designed to check if the predicted ages and actual ages are less than the criterion set, it would take it as correct predictions[2]

By implementing this method and setting a criterion of 10, I ended up having 96% accuracy with less Mean absolute error of 4.

Moreover, after successfully evaluating the model with their training metrics, I got the results as follows:

|  |  |  |
| --- | --- | --- |
| PROBLEM | METRICS | PERFORMANCE |
| Age Regression | MAE | **5.8** |
| Gender Classification | Accuracy | **85%** |
| Ethnicity Classification | Accuracy | **68%** |

**Future work/Recommendation**

I would like to recommend in the future to get and curate a more diverse and representative dataset for age, ethnicity, and gender prediction. Ensure that the dataset spans different ethnic groups, age ranges, and gender identities to improve model generalization. Investigate the use of transfer learning techniques to leverage pre-trained models on larger and more diverse datasets, which may improve model accuracy.

**References:**

1. Raman, V., ELKarazle, K. and Then, P., 2022. Gender-specific facial age group classification using deep learning. *Intell. Autom. Soft Comput*, *34*, pp.105-118.
2. Kim, J.W., Jung, J.Y., Lee, S.W., Baek, W.Y., Kim, H.A. and Suh, C.H., 2022. S100A8 in serum, urine, and saliva as a potential biomarker for systemic lupus erythematosus. *Frontiers in Immunology*,