OPT 2 – COMPUTER VISION (2022 / UB)
REPORT: **Task 1**

Group members:

Johnny Núñez Cano (1), <jnunezca11@alumnes.ub.edu>, johnnynunez
Qijun Jin (2), <qjinjinj9@alumnes.ub.edu>, QJ

1. SUMMARY OF CONTRIBUTIONS

In this practice, we are trying to predict the ages of the people, given a set of metadata such as gender, ethnicity, and facial expression. We will apply the ResNet as our backbone model of the neural network. In addition to this model, it will be nested with a set of fully-connected layers in order to improve the accuracy of the prediction. More concretely, we will set up some enhancements over the following points:

1- Model and regularization

By the configuration of the project, the strategy to follow is to optimize the baseline model with a set of nested fully connected trainable layers until finding one that gives the best result and then to optimize the whole trainable layers of the neural network, including the ResNet model. By performing these two steps, we want to get a model that is decent for us to predict the ages. Moreover, if the model stacks at some point do not minimize the loss value, then we will apply some regularization methods to fix the problem. One of them is to add dropout layers in between the fully connected layers which will help our neural network to avoid memorization.

2- Hyperparameters

We have also applied some modifications to the hyperparameters of the model such as the learning rate, the optimizer, and other losses.

With respect to the learning rate, our first attempt is to increase the learning rate to 1e-3, but the loss of the validation curve jumps a lot along the training curve, which results to be too much for the model to learn, meaning that the model stacks at some local minimal. After several attempts, we find out that the best learning rates are 1e-4 and 1e-5. So to find the best learning rate for the model, we have used a scheduler that will decrease the value of the learning rate dynamically during the learning process. It will decrease the value of the learning rate after performing a period of epochs if the loss of the validation is not decreased at all.

With respect to the loss, we have compiled the model with MeanAbsoluteError to be the stopping metric, and the result we have obtained is slightly better than the original compile method, MeanSquaredError. Therefore, we have tried to improve it with the Adam optimizer of Adam, modifying some parameters like beta_1, beta_2, epsilon. But the result was not improved at all.

In the experiment section, we will show our chosen model with a different configuration of the fully connected layers and hyperparameters.

3- Data augmentation

Once we have the model improved with the dropout layers, we are still facing the problem of underfitting, meaning that the loss of the validation cannot be properly adapted to the loss of the training. Therefore, another method is needed in order to give a higher accuracy of the prediction. In this case, we are needed to perform a data augmentation of both the training set and validation set. As a result, we can see in the experiment section that the accuracy of the model is higher.

2. EXPERIMENTAL SETUP & DISCUSSION OF THE RESULTS

M1 - Baseline model - last layers trainable - without data augmentation:

By the initial configuration of the project, we have trained the baseline model with only the last layers trainable. In this case, we want to get a result that helps us to compare our future prediction's accuracy.

M2 - Baseline model - all layers trainable - without data augmentation:

Then, we have performed the same model with all the layers trainables, so that we can get a prediction that fits the whole neural network.

M3 - Model improved - all layers trainable - without data augmentation:

After this step, we have improved the last fully connected layers to be nested with neurons of size [512, 128, 64, 32] and added a dropout layer of rate 0,1.

M4 - Model improved - all layers trainable - without data augmentation - MAE:

In the fourth model, we have chosen the MeanAbsoluteError to be the metric to optimize the loss function.

M5 - Model improved - all layers trainable - with data augmentation - MAE:

In this last model, we have performed the data augmentation in order to improve the prediction of the ages. And also, we have improved the last fully connected layers to be nested with neurons of size [512, 256, 128, 64, 32] and added a dropout layer of rate 0,2.

Comparison of error for all the experiments:

Model	***Learning rate***	***Training strategy***	***Gender bias***	***Expression bias***	***Ethnicity bias***	***Age bias***	***MAE***
M3	*1e-5*	*1*	*0.044*	***1.4***	*2.8*	*8.45*	*10.45*
M4	*1e.5*	*2*	*0.07*	***0.578***	*0.706*	*7.07*	*8.420*

M5	1e-5	2	**0.044955**	**0.162736**	**0.539348**	**3.056905**	**6.340045**

Table 1: Comparison table of error and bias, results obtained for all models

As we can observe on the table of different kinds of errors, we have assumed that augmenting data of those datasets which have less data can provide a better performance in the prediction. In our case, we have a good prediction with low bias on gender, facial expression, and age while having an MAE of 6.34 as a global error. The only biased error is produced on the ethnic category.

```
# Using the FC layer before the 'classifier_low_dim' layer as feature vector
fc_512 = model.get_layer('dim_proj').output

# adding a dropout layer to minimize overfiting problems
dp_layer = Dropout(0.2)(fc_512)

# adding a few hidden FC layers to learn hidden representations
fc_256 = Dense(256, activation='relu', name='f_256')(dp_layer)

dp_layer_2 = Dropout(0.2)(fc_256)

# adding a few hidden FC layers to learn hidden representations
fc_128 = Dense(128, activation='relu', name='f_128')(dp_layer_2)

dp_layer_3 = Dropout(0.2)(fc_128)

fc_64 = Dense(64, activation='relu', name='f_64')(dp_layer_3)

dp_layer_4 = Dropout(0.2)(fc_64)

fc_32 = Dense(32, activation='relu', name='f_32')(dp_layer_4)

# Includint an additional FC layer with sigmoid activation, used to regress
# the apparent age
output = Dense(1, activation='sigmoid', name='predict')(fc_32)

# building and pringing the final model
model = Model(inputs=model.get_layer('base_input').output,outputs=output)
#print(model.summary())
```

Exercise 1

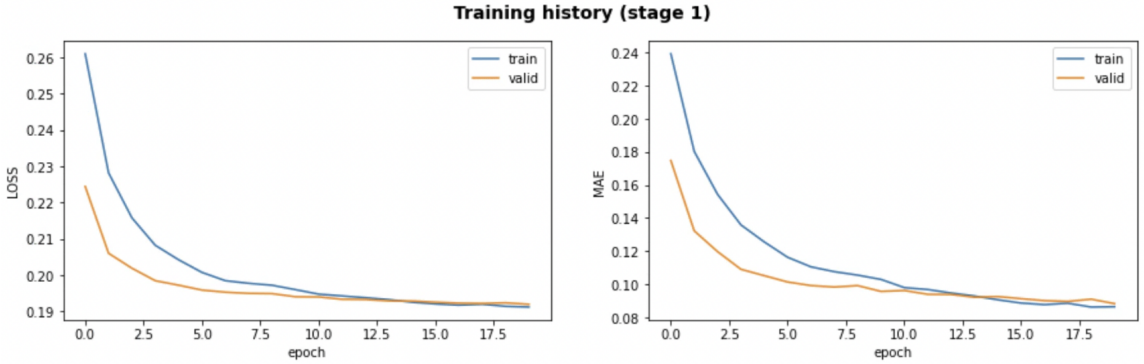

Figure 1: illustrative example of training curve on Exercise 1.

As we can see in table 1, we have a lot of improvement with respect to the default model. We get the network to really learn only by training the last layers.

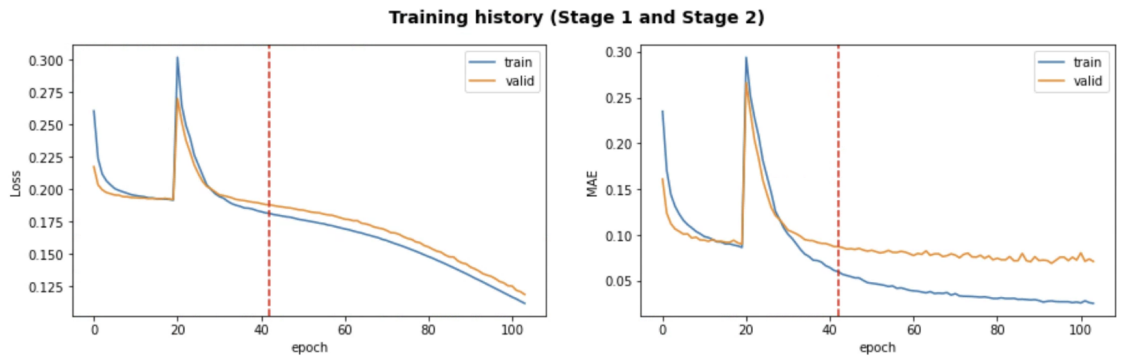

Figure 2: Same Model, All layers trainable

As we can see in figure 2, dropouts produce some noise and we could stop training before because val_mae is not decreasing. If we use high values on dropouts we could stay in a case about “underfitting”. In the beginning, val_loss and val_mae are less because the NN uses all neurons to check.

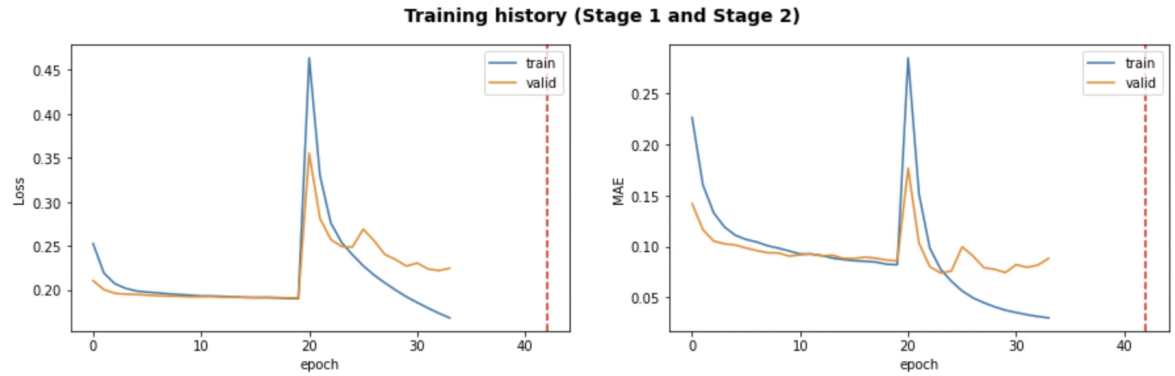

Figure 3: Improvement Model, All layers trainable + Data Augmentation

Here, we changed early stopping to monitor val_mae and new architecture to increase one more layer of 256 and dropouts after each layer with 0.2. We can see that after 21 epochs we obtain the best result, 6.21 in MAE exactly. We think that there exists more improvement in our model seeing that val_mae loss.

3. FINAL REMARKS

To conclude, we have seen that the distribution of the dataset might produce a biased prediction over some subcategories. So a method to unbias this problem is to balance the dataset of the subcategories which have fewer data. Moreover, a data augmentation helps the neural network to learn more precisely the features of the dataset and then give a more accurate prediction. This accuracy is observable by looking at the losses on each epoch. We believe that applying multiple fully connected layers with dropout layers can help the neural network predict with less error. We also believe that a variation over our model can also improve the performance, or even introduce a more balanced dataset over certain subcategories.