UNIVERSITY OF
SCIENCE AND TECHNOLOGY
OF HANOI

MARCH 2023

AG PROJECT REPORT

PREPARED BY

NGUYEN PHAN GIA BAO
CAN TRUNG HIEU
CHU BAO MINH
DOAN HOAI NHI

Table of contents

I. INTRODUCTION

- The identification of age and gender are crucial aspects of our social interactions. Various languages across the world have different ways of addressing individuals based on their gender and age, with different vocabularies used for addressing young and old people. These customs are primarily reliant on the ability to estimate a person's age and gender based on their facial appearance. With the surge of social media and networks, numerous application developers are now developing automatic age identification tools. Age and gender are essential facial characteristics for social interactions.

- The face of a human being has a variety of characteristics that can help determine their identity, age, gender, emotions, and ethnicity. Among these features, being able to identify a person's age and gender can be very beneficial in many real-world scenarios, including visual surveillance, medical diagnosis (such as detecting premature facial aging), human-computer interaction systems, access control or soft biometrics, demographic data collection, law enforcement, and marketing intelligence. Consequently, it is crucial to be able to recognize age and gender from images of faces. Nevertheless, researchers still see several difficulties in age and gender identification as unresolved challenges.

- Now, with the advancement in deep learning, it has become possible to classify age and gender accurately using deep neural networks. In this report, we will discuss a deep-learning model for the classification of age and gender from facial images.

II. DATASETS USED IN THE PROJECT

- In this case, we use a set of images from the UTKFace Kaggle. 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. Then we transform the data of those images. The most common data transformations are raw data conversion into a clean and usable form, data type conversion, and duplicate data removal to benefit the organization.

Figure 1: Sample facial images found in UTKFace Dataset

- In order to be able to easily analyze and run the model, we use some support libraries for the fastest and most accurate image processing such as cv2, TensorFlow, and Keras, etc.

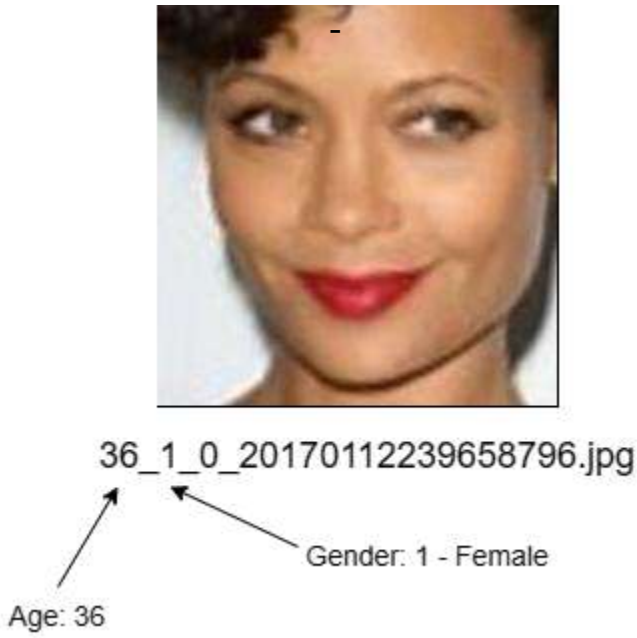


Figure 2: Image label_name

- Here, we can see a character string that says 16_1_20170112239658798. These numbers each have their own meanings and purposes. The number "36" indicates the age of the person in the photo and "1" represents the

female gender. We normalized to scale the values of a variable to a range between 0 and 1. “1” is normalized as female and “0" is normalized as male. This helps us when training the model, which can help the machine distinguish and give a higher accuracy rate.

III. METHODOLOGY

- In this project, we analyze and train the model by 3 methods: Convolutional neural network and K-fold cross-validation and label division

A. Convolutional neural network (CNN)

A1. Introduction of CNN

- For image processing and object recognition applications, convolutional neural networks (CNNs) are a unique kind of neural network design that is frequently utilized. CNNs have also been used in recent years to forecast age and gender based on facial scans. In a variety of industries, such as marketing, security, and healthcare, this is a crucial task. We will look at some of the opportunities and difficulties that come with using CNNs to predict age and gender in this essay.

- Due to the variety in appearance and the possibility that different persons will age and look different over time, determining age and gender from facial photographs is a difficult undertaking. Variations in lighting, facial emotions, and stance further complicate the endeavor. Nonetheless, CNNs have been employed in many applications and have proven to be very good at this task.

- The capacity of CNNs to learn features directly from the raw picture data is one of their important characteristics that makes them suitable for the age and gender prediction. This indicates that CNNs do not require manual feature engineering to automatically extract pertinent elements from the image, such as facial structure, wrinkles, and hair color. Furthermore, CNNs have the capacity to learn features at a variety of sizes and degrees of abstraction, which is crucial for capturing the intricate variations in appearance that exist across different ages and genders.

A2. Apply CNN in our project

A2.1. For age_model

- In this case, our group creates the convolutional layers of the model. We conv2D which defines a convolutional layer with 32,64,128,256 and 512 filters, each of these has a kernel size 3x3 and RELU activation function. In addition, we used “maxpolling2d” to reduce the dimensions of the features by a factor of 2.

- Next, we used the function “Dropout” to randomly drop 25% of the input to prevent overfitting. During training, a certain proportion of neurons are lost at random, forcing the network to pick up more reliable characteristics. The output of the convolutional layers is transformed into a 1D tensor via the Flatten function.

- The neural network has six fully connected layers with, correspondingly, 512, 256, 128, 64, 32, and 20 neurons for age prediction. The final layer outputs a probability distribution over the 20 age groups using the sigmoid activation function.

- The neural network comprises six fully connected layers with 128, 64, 32, 16, 8, and one neuron each for gender prediction. A probability distribution over the two genders is output by the last layer using the sigmoid activation function.

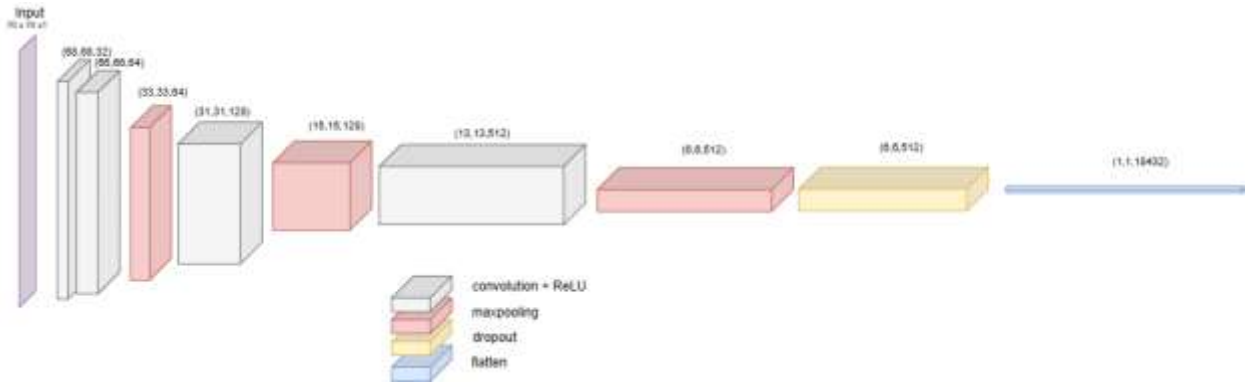


Figure 3: Step by step of CNN

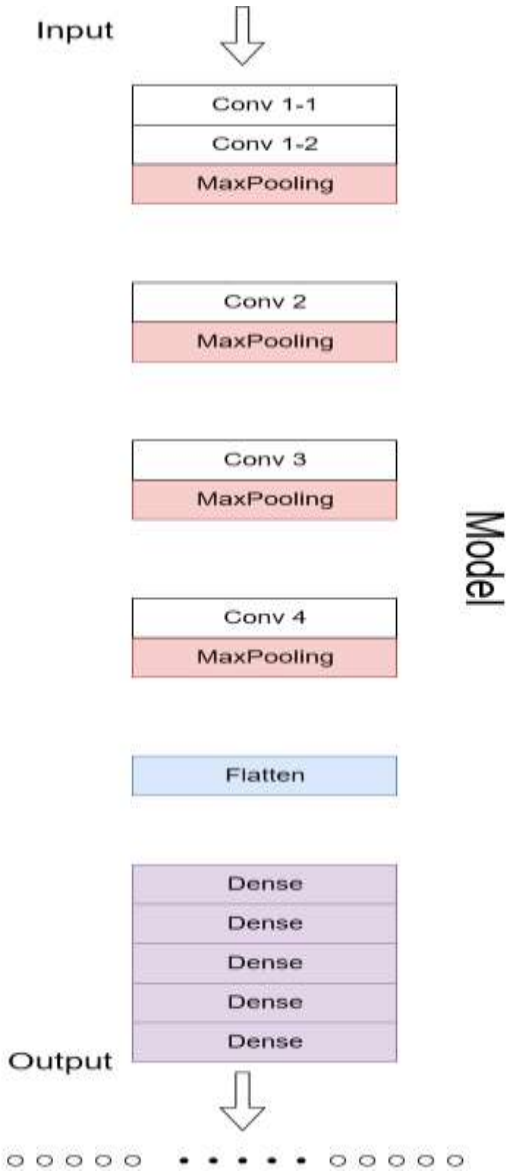


Figure 4: Illustration of age_model using CNN

A2.2. For gender_model

- We create function img_cnn takes an optional parameter drop_rate which specifies the dropout rate to use in the network.

- The network architecture consists of five convolutional layers, each followed by a max pooling layer. The convolutional layers use 3x3 kernels and have 32, 64, 128, 256, and 512 filters, respectively. The max pooling layers use 2x2 pooling windows. The output of the final max pooling layer is

passed through a dropout layer with a dropout rate of 0.25 and then flattened.

- The flattened output is then passed through a series of fully connected (Dense) layers with ReLU activation functions. Each of these layers except for the final one also includes a dropout layer with the specified dropout rate. The final Dense layer has a single output with a sigmoid activation function, which produces a probability score indicating the predicted gender of the input image (0 for male and 1 for female).

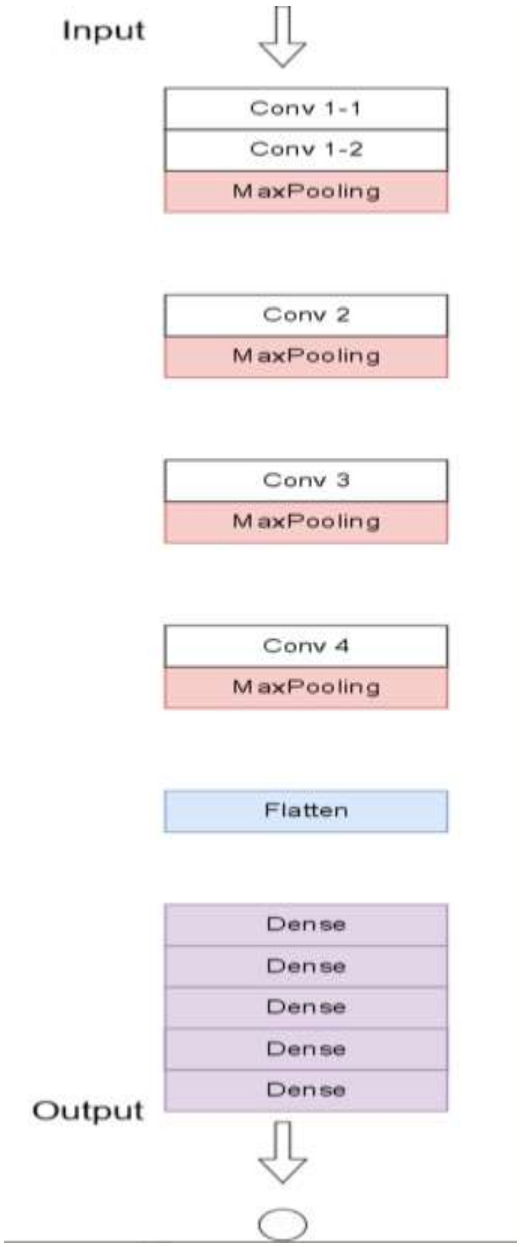


Figure 5: Illustration of gender_model using CNN

B. K-fold

- In data analysis and machine learning, the K-fold cross-validation technique is used to assess a model's performance. An equal-sized subset or fold is created by splitting the dataset into k parts. The model is then developed on k-1 folds before being evaluated on the final fold. Each fold serves as the test set just once as this process is performed k times.

- By giving a more reliable estimate of the model's performance, the k-fold cross-validation procedure helps to reduce the danger of overfitting. Also, it helps to guarantee that the model is not biased toward any certain subset of the data.

- Our group used the K-fold cross-validation technique to evaluate the performance of the model by dividing the data into 10 subsets. We also take random_state equal to 42 to sets the random seed to make the data split consistently across multiple runs. To prevent any systematic bias resulting from the order of the samples, the shuffle parameter is set to true, causing the data to be shuffled before being split into folds.

- The KFold object's 10 folds are iterated over in the for loop. The training and validation sets' indices are acquired using the train index and test index, respectively, in each iteration. Following that, the matching subsets from the original dataset are extracted using these indices.

⇒ By evaluating the model on several subsets of the data, k-fold cross-validation in this code helps to enhance the model's generalization performance while also ensuring that the model is not overfitting or underfitting to any particular subset of the data.

C. Label division

- Label division is the process of categorizing or classifying a dataset according to the labels or tags that have been given to each individual data item. This method is frequently used in data analysis and machine learning to categorize data and create predictions based on those categorizations.

- In this dataset, we have ages ranging from 0 to 116. Therefore, to facilitate the accurate prediction of age for each image, we divided the ages into 20 groups, with each group consisting of a range of 5 years. Age grouping helps to increase the accuracy of predictions and facilitates the handling of age-related tasks.

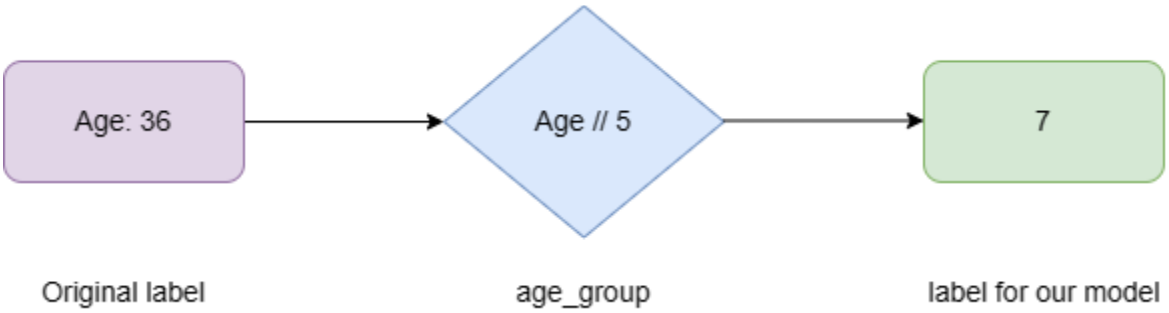


Figure 6: Decomposition of original age labels into labels for different models.

IV. ANALYSIS OF RESULTS

A. Data pre-processing

- Preprocessing the data is the first stage in creating any deep-learning model. In this instance, facial traits must be extracted from the photos. For this aim, a variety of face landmark detection algorithms are available. Once the facial landmarks have been recovered, we can utilize them to crop and align the photos. This process is crucial because it lessens the diversity in the photos brought on by various positions and facial expressions.

- Before going into data processing, our team has statistics on the number of men and women in the age range that we have divided and here is the graph showing that:

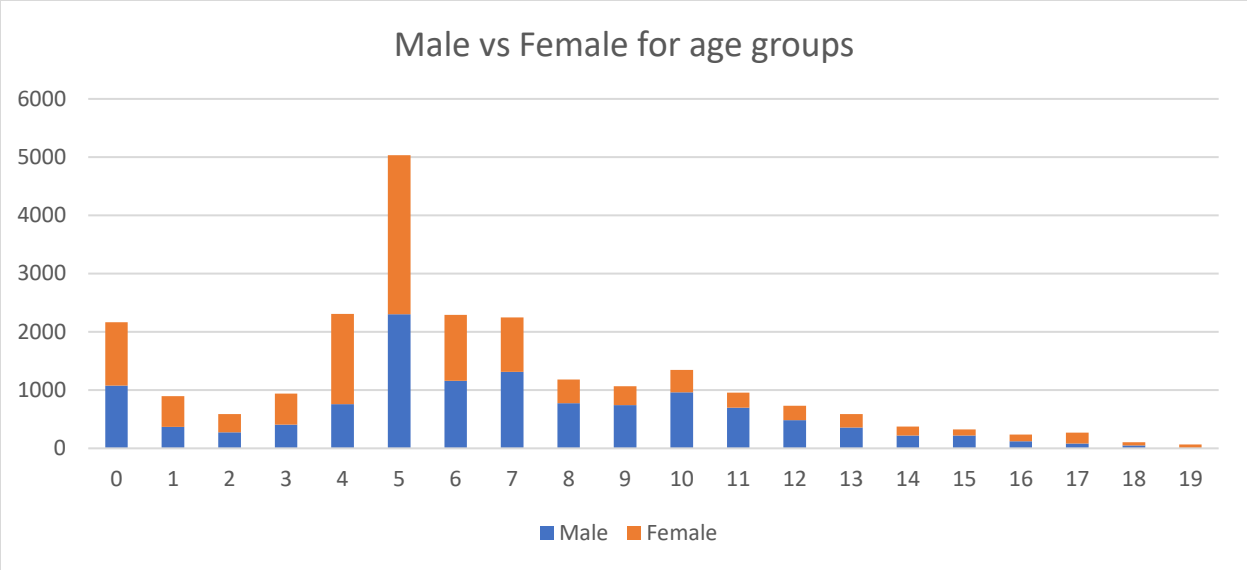


Figure 7: Chart shows the number of age male and female

⇒ Based on this chart, we see that the total number of males and females in each age group is different. The number of males is more than the number of females in total. In the youngest age group (0-4), there are slightly more females than males. Conversely, in the two oldest age groups (18-19), there are slightly more males than females. The number of people in the age group 5 to 9 is quite large compared to other age groups, with the highest number in the age group 5. The number of people in the older age groups (over 19) is very low. This could potentially indicate a population with a low birth rate or a higher mortality rate among older adults.

⇒ The number of men and women in each age group increased gradually from the young age group to the middle age group and then gradually decreased from the middle age group to the elderly age group. This reflects a change in population structure

A1. Rescale the image to speed up computation

- Because in this project we use google collab to train the model, so all of the images have to switch to black and white to do it. In addition, to be suitable for use, our team has reshaped the entire image to 70x70 pixels. This helps to prevent the image size from being too large or the image too large.

Figure 8: Image after changing to black and white with size 70x70 pixel

- Next, we performed feature scaling on the dataset to determine the number of samples for training and testing. The main purpose of feature scaling is to normalize the features so that they can be compared on a common scale.

- We initialize a target array of the 2-dimensional array to store the age and gender values of the people in the dataset and a feature array is a 4-dimensional array, where the first-dimension stores information about the image, the 2nd and 3rd dimensions store information about the length and width of the image, and the last dimension stores information about the color channel. We loop through the images and add age and gender values to the target array. Also, we add the images to the feature array.

- We normalize the values in the feature array by dividing all values by 255 to get them between 0 and 1. Next, we use the train-test-split function from the sklearn library to split the data into train and test sets with a test-size ratio of 0.2 and shuffle as true to avoid the case that images with the same target value are stacked.

A2. Handle labeling by one-hot-encoding

- We divide ages into 20 groups of 5 years each and create an array named age-one-hot to represent the age value as one-hot encoding, using the one-hot function from the TensorFlow library. This age-one-hot array has a size of 20 and contains the values 0 or 1 corresponding to each age group.

- We also create an array of ages to train and an array of gender to train to contain target values of age and gender to use for the training model. The age-to-train array will be used as one-hot encoding, while the gender-to-train array will keep its original value.

⇒ As a result, we obtained 18,966 images for training, with a size of 70x70 pixels and each element in 70x70 has a length of 1. Similarly, we obtained 4,742 images for testing, with a size of 70x70 pixels, and each element in 70x70 has a length of 1.

B. Experimental results

- In our project, we use 3 models: model using sigmoid and model using RELU and model vgg19

B1. Model using sigmoid

- Our group trained the model using k-fold with batch_size = 25 and epochs = 15. The variable "log" is used to store the index of each training epoch on the CNN model. If the variable "c" is entered as "N" (meaning no retraining from the last epoch), the "log" variable is used to store the index of each epoch in the for a loop. If the variable "c" is entered as "Y" (meaning retraining from the last epoch), the initial value of "log" is assigned to the index of the last epoch saved on the disk and is used to retrieve the training and test indexes from "train_indexes" and "test_indexes". In both cases, the "log" variable is used as an identifier for each epoch to manage to store and access information related to training the CNN model.

⇒After running 20 folds, we achieved an accuracy of 78.19% and a gender accuracy of 99.08%.

- And to compare the loss models after each fold, we plotted the accuracy values after each epoch of age_train, age_val, gender_train, and gender_val.

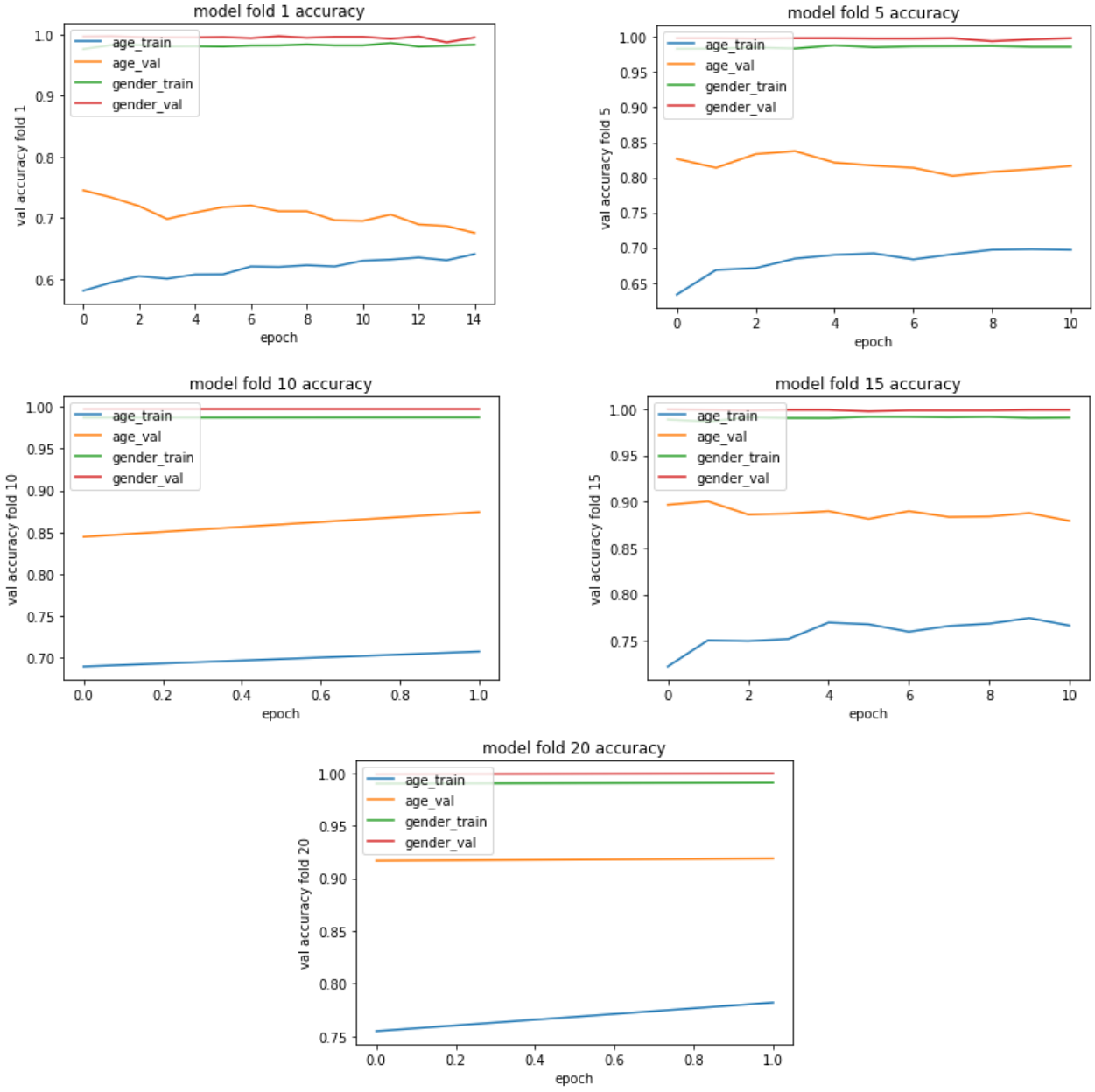


Figure 9: The graph shows the accuracy of the model after fold

⇒ The charts indicate that the accuracy of the models on the training and validation sets grows with the number of epochs, demonstrating that the models are learning and can continue to learn if more epochs are added. There are instances where, nevertheless, as the number of epochs is raised, the accuracy on the validation set declines, suggesting that the models are overfitting. Both

the training and validation sets' accuracy values for both models are quite high (over 90%), demonstrating the models' success with the given dataset.

- After that, we are going to predict the age. In this case, we used a function named decode_one_hot to take a one-hot-encoded array of age values and return the index of the maximum value in the array, which corresponds to the predicted age category.

- Then, we get the age range which is defined as a 5-year interval, with the lower bound given by d*5 and the upper bound given by d*5+4. If the age value is outside the range 0-19, it will return "Unknown".

- Similar to gender, we create a function “get_gender” and have a probability. This function takes a probability value between 0 and 1 and returns the predicted gender as a string. If the probability is less than 0.5, the function returns "Male"; otherwise, it returns "Female".

```
Actual Gender: Female
Age: 38
Age values: [[2.2975435e-05 6.0980557e-05 4.9186576e-05 3.3155363e-04 6.7611336e-04
  1.7806198e-02 6.8246879e-02 2.1230361e-01 2.3592751e-01 2.5398067e-01
  1.4511259e-01 5.5186067e-02 7.1078497e-03 2.6258421e-03 3.6208174e-04
  1.8037419e-04 1.4836843e-05 4.5997517e-06 1.4817964e-07 1.0859999e-13]] Gender values: [[0.9875144]]
Index predicted:  9
Predicted Gender: Female Predicted Age: 45 - 49
```

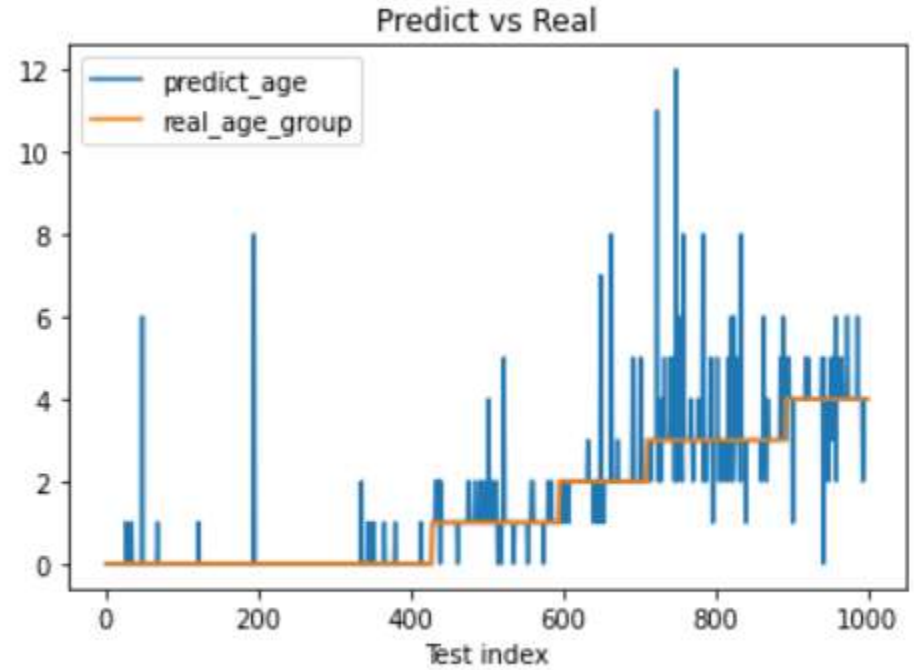
```
Actual Gender: Male
Age: 28.0
Age values: [[9.0138874e-06 1.0350604e-06 4.6539088e-07 1.7956909e-04 3.3258609e-02
  8.1106740e-01 1.4368784e-01 1.1595743e-02 1.9560120e-04 4.6144628e-06
  1.4381462e-07 2.3422007e-08 1.7448064e-09 1.9884427e-10 4.1337000e-10
  2.0231981e-11 8.5926774e-13 5.2649079e-12 8.3797578e-17 6.3612844e-24]] Gender values: [[7.5649886e-19]]
Index predicted:  5
Predicted Gender: Male Predicted Age: 25 - 29
```

```
Actual Gender: Male
Age: 1.0
Age values: [[1.0000000e+00 1.4037824e-19 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00
  0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00 0.0000000e+00]] Gender values: [[0.00442577]]
Index predicted:  0
Predicted Gender: Male Predicted Age: 0 - 4
```

```
Actual Gender: Female
Age: 67.0
Age values: [[4.45886653e-05 6.18505292e-05 1.25895795e-05 1.80514453e-05
  2.49004006e-05 1.66635471e-03 6.58673700e-03 1.20957578e-02
  3.27956565e-02 4.48835045e-02 1.69520184e-01 2.09661275e-01
  2.54371822e-01 1.84968814e-01 6.00774549e-02 1.77995730e-02
  4.29867953e-03 8.62483284e-04 2.47943361e-04 1.79328526e-06]] Gender values: [[0.99999684]]
Index predicted:  12
Predicted Gender: Female Predicted Age: 60 - 64
```

Figure 10: The result after predicted age and gender

- And to give everyone a better view of the predicted value and the actual value, our team plotted the graph and their accuracy.




```
Accuracy:  0.7397722479966259
```

Figure 11: The plot shows the predicted age and real age

⇒ As we can see from the graph, the distribution of real ages is uneven and tends to increase significantly in the age group of 80-100 years old. At the same time, the predicted ages follow a stepped pattern and do not remain fixed in any particular age group. The largest number of people fall into the age group of 0-20 years old. In addition, the accuracy rate is quite high, reaching nearly 74%.

- Finally, we draw the confusion matrix. A confusion matrix is a technique used in data science and machine learning to assess how well a classification model is performing. The number of the model's true positive, false positive, true negative, and false negative predictions are listed in a table.

- A classification model's performance can be evaluated using a confusion matrix. It can be used to compute a number of evaluation measures, including F1-score, recall, accuracy, and precision. These metrics are useful for evaluating the model's performance and making choices on how to make it better.

⇒ Overall, the confusion matrix is a significant tool for assessing and optimizing classification model performance, and its application is essential to the effective implementation of machine learning models in practical applications.

- In this case, we calculate the predicted mean, real mean, predict variance, and real variance. And with this dataset, we calculate predicted mean = 6.30, real mean = 6.45, predict variance = 15.36 and real variance = 16.68

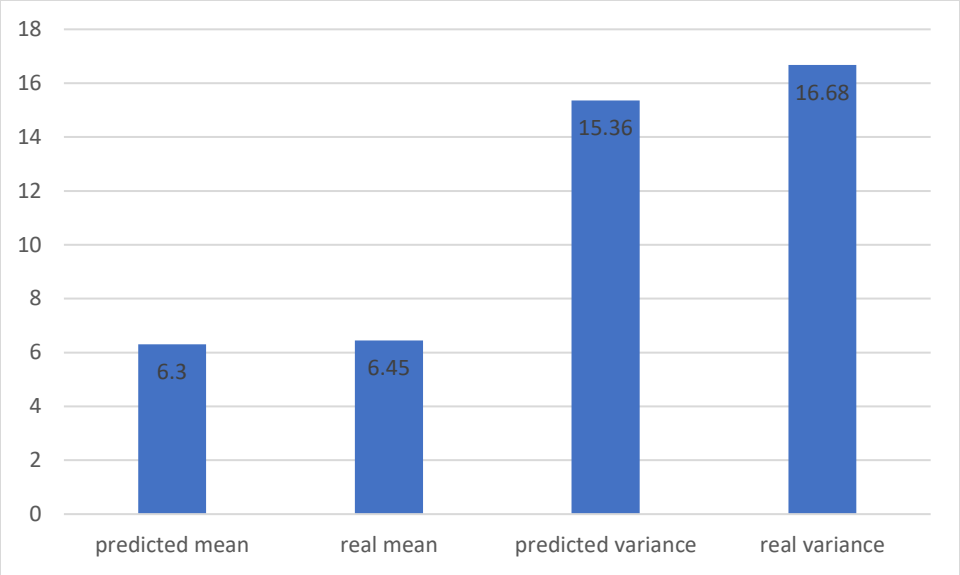


Figure 12: Chart to shows the value and compare

- Compared to the actual mean of 6.445, the predicted mean of 6.303 is lower. This demonstrates that the model, on average, consistently underestimates the target variable.
- In comparison to the true variance of 16.681, the predicted variance of 15.362 is smaller. This suggests that the model is not fully capturing the range of variability in the data because the model's predictions are less evenly distributed than the actual values.
- Since the model is constantly under-predicting and not capturing the full range of variability, these results collectively suggest that the model may not be effectively predicting the target variable. To pinpoint specific areas for improvement, it is crucial to do further analysis of the model's performance, maybe by employing a confusion matrix.

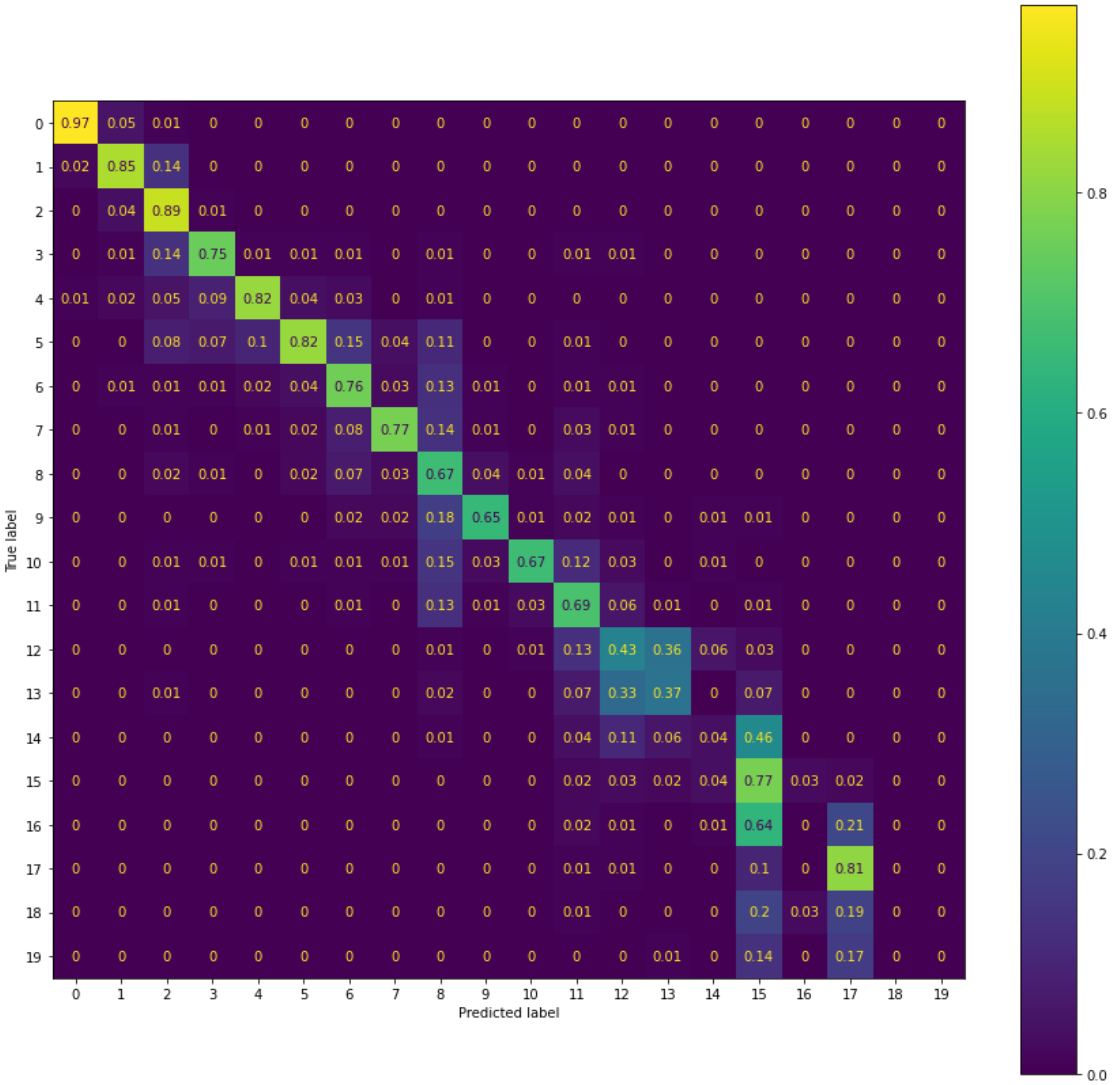


Figure 13: Confusion matrix

B2. Model using RELU

- We model using RELU with the same steps as we do with sigmoid except that the last layer of age_model converts from sigmoid to RELU with 1 value
- After running 20 folds with ReLU method, we achieved an overall accuracy of 75.06% with over 29% for predicting age and 98.02% for predicting gender.
- And to compare the loss models after each fold, we plotted the accuracy values after each epoch of age_train, age_val, gender_train, and gender_val.

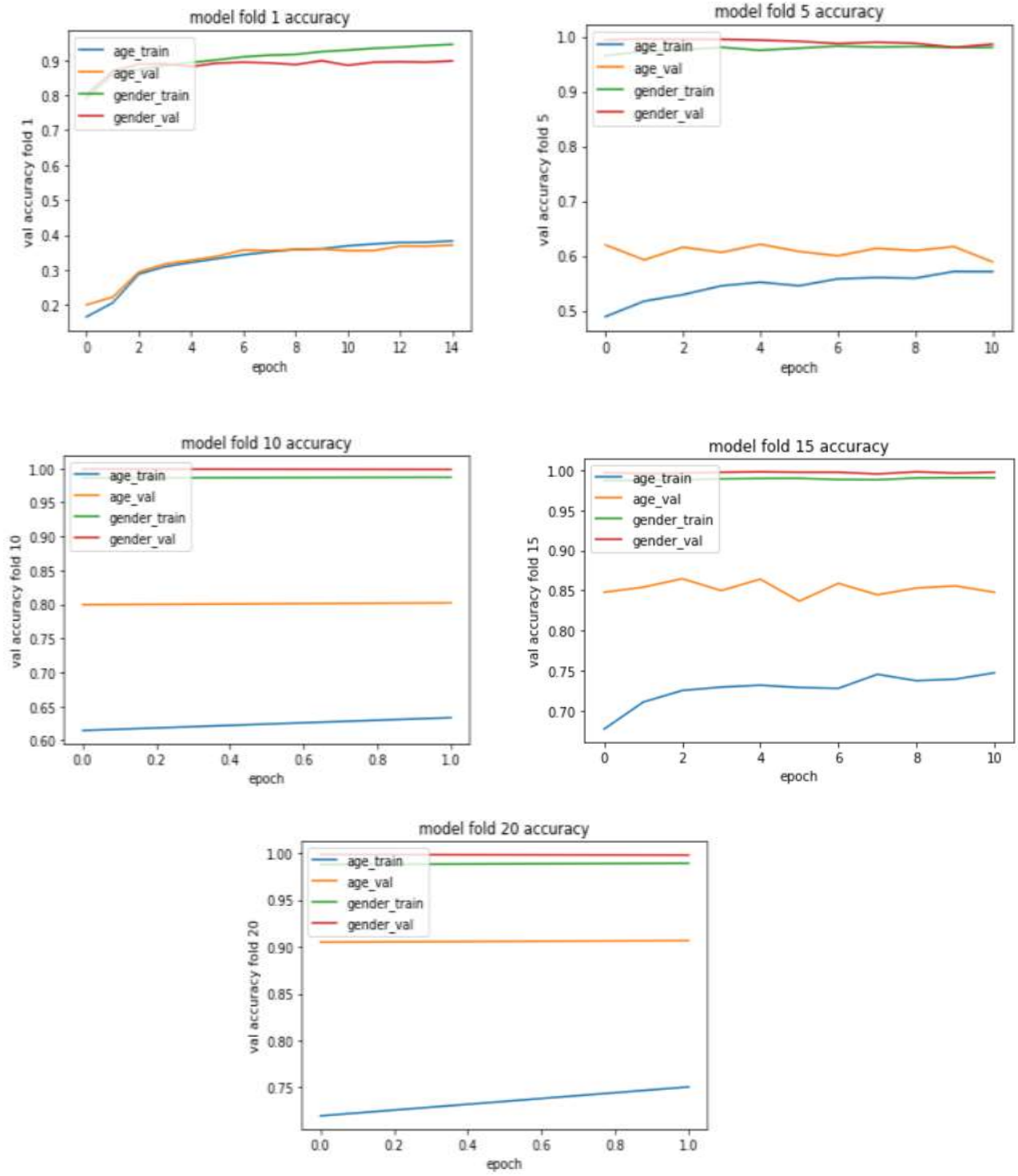


Figure 14: The graph shows the accuracy of the model after each fold

⇒ The values of gender_train and gender_val are essentially identical, with a very high accuracy rate of almost 100%, as can be seen by looking at the plot. They parallel each other and go even closer to 100% as we get to the latter folds. Moreover, the age_val accuracy rate varies and gradually rises after each fold. They began at a very low level, but by the last fold, they had increased to a relatively high accuracy rate of nearly 90%. Similar to age_val, age_train increases steadily and reaches over 75% accuracy in the final fold.

- The steps and for predicting age and gender are similar to the model using sigmoid

```
Actual Gender: Male
Age: 29
1/1 [==============================] - 0s 41ms/step
Age values: [[0.31101552]] Gender values: [[2.5228715e-34]]
Predicted Gender: Male Predicted Age: 30 - 34
```

```
Actual Gender: Male
Age: 43
1/1 [==============================] - 0s 46ms/step
Age values: [[0.42054904]] Gender values: [[0.]]
Predicted Gender: Male Predicted Age: 40 - 44
```

```
Actual Gender: Female
Age: 40
1/1 [==============================] - 0s 41ms/step
Age values: [[0.42683893]] Gender values: [[1.]]
Predicted Gender: Female Predicted Age: 40 - 44
```

Figure 15: The result after predicting age and gender using ReLU

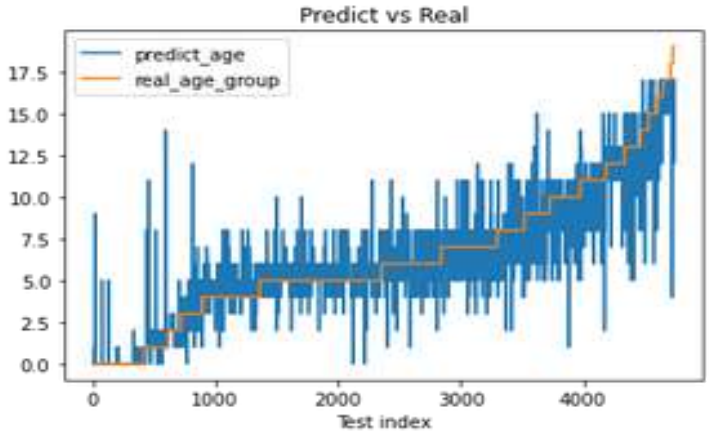


Figure 16: Accuracy of the predicted age and real age on the test set

⇒ With an accuracy of approximately 29% on the test set, the predicted ages are mostly concentrated in the age groups of 6 to 12, which corresponds to actual ages of 30 to 60. This can be explained by the following reasons:

- Insufficient data for each age group: the dataset contains more than 15,000 images within this age range (accounting for about 65% of the dataset).
- Insufficient training time.

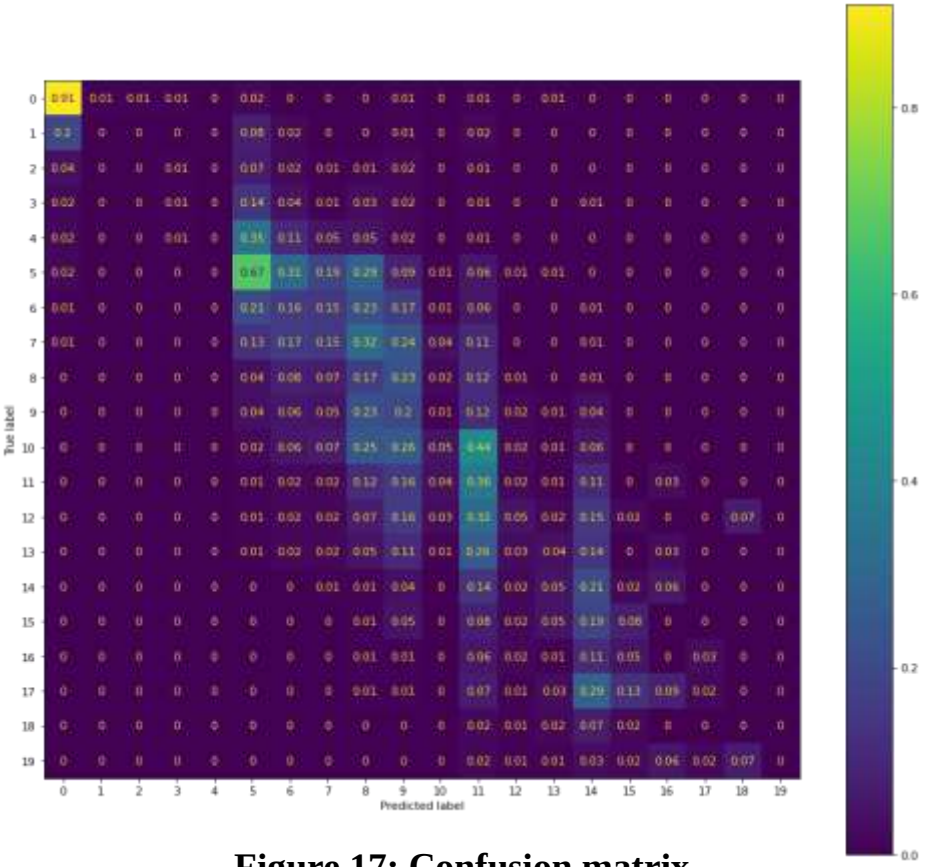


Figure 17: Confusion matrix

As predicted in the chart above, to the confusion matrix, this is clearly shown:

- The rate of prediction into groups from 5 to 12 is quite high, especially the rate of other groups being wrongly predicted into these groups

- For the remaining groups, for example, groups from 14 onwards, the amount of data is relatively small, so the ratio is approximately 0

B3. Model using VGG19

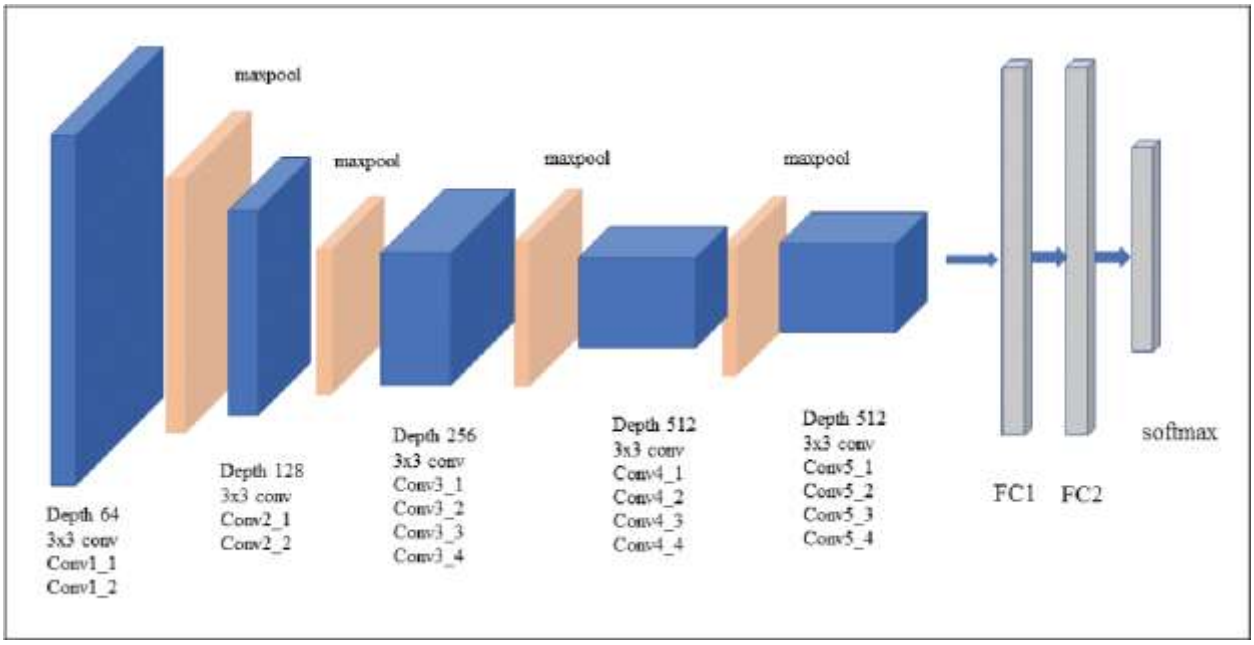


Figure 18: Step by step of VGG19

- The step is similar to the RELU and Sigmoid but it has some differences in the CNN

- Convolutional layers (Conv2d), activation function layers (RELU), max-pooling layers (MaxPool2d), fully connected layers (Linear), dropout layers (Dropout), and a log-softmax layer are among the layers that make up the network (LogSoftmax).

- There are a total of five convolutional layers in the CNN, and each is followed by a max pooling layer and a ReLU activation function. The output is then flattened and transmitted through six fully connected layers, each of which is followed by a dropout layer and a ReLU activation function. To obtain the predictions, the output is finally sent through a linear layer before being transmitted through the log-softmax layer.

- The purpose of the dropout layers is to prevent overfitting by randomly dropping out some of the neurons during training, forcing the network to learn more robust features. The dropout rate is set to 0.5, which means that each neuron has a 50% chance of being dropped out during training.

- The network accepts a tensor of the form [batch size, 1, 28, 28], where the batch size is the number of photos in each batch, 1 is the number of channels (since the images are grayscale), and 28x28 is the size of the input images. The result is a tensor of the shape [batch size, 10], each element of which is a probability tensor for the relevant class.

- We use VGG19 for predicting the age of a person based on his/her image. The default model is VGG19, but it can be switched to CNN with the mtype parameter. The loaded parameter can be used to load a pre-trained model. The optim, learn_rate, and crit parameters are used to set the optimizer, learning rate, and loss function for the training process. The max_ep parameter sets the maximum number of epochs for training, and the max_fd parameter sets the maximum number of folds for k-fold cross-validation.

- The main function starts by initializing the model and the optimizer. If a pre-trained model is provided, the function loads the model and optimizer states using the load_state_dict function. The function then runs a loop over the number of epochs specified. Inside this loop, it performs k-fold cross-validation using the kfold function from the sklearn library. For each fold, it initializes the data for training and validation, and creates data loaders using the DataLoader function from the torch.utils.data library, and trains the model on the training data. It then evaluates the model on the validation data and calculates the loss. After training and evaluating the model on all the folds, the function saves the model and optimizer states to a file.

- Next, the function tests the model on a separate test set and calculates the test loss. It also calculates the accuracy of the model by counting the number of correct predictions and dividing it by the total number of predictions. If the accuracy is better than the previous best accuracy, the function saves the model and optimizer states to a file.

- Finally, we calculate the mean, accuracy, and variance of this dataset. And we have mean = 1.04723, variance =1.689977 and accuracy = 40.36%.

⇒ Mean has a value that is nearly 1.05. This would suggest that the positive class is favored in the majority of the model's predictions. Variance has a quite big value, about 1.69. This can be a symptom of considerable data dispersion, a wide range in the model's anticipated values, and a lack of stability. The accuracy value is only 40.36%, which is really poor. This shows that the model struggles to distinguish between positive and negative situations. In conclusion, the model has to be improved in order to improve accuracy and decrease data dispersion.

- And to show those data more clearly, we plotted out so that we can clearly see predict age and real age.

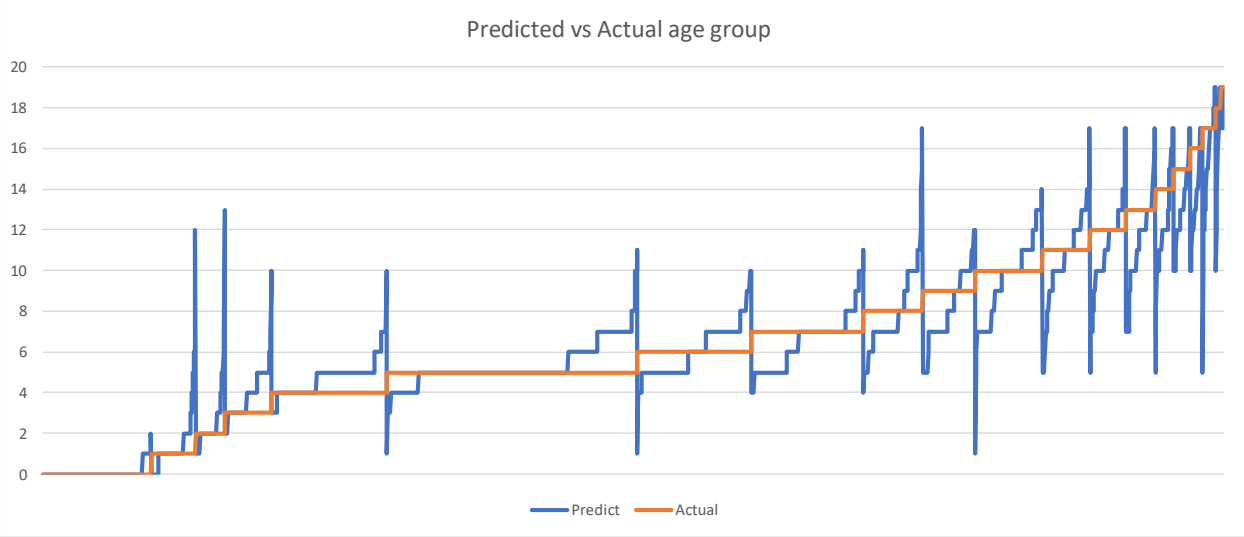


Figure 19: The plot shows the predicted age and real age

⇒ Looking at this graph we can see that the actual ages are unevenly distributed and tend to be laddered. The maximum number of real ages is around 40-60 years old. Also, the true age tends to increase with the number of tests. In addition, the predicted age is also unevenly distributed and tends to increase as the number of tests is increased.

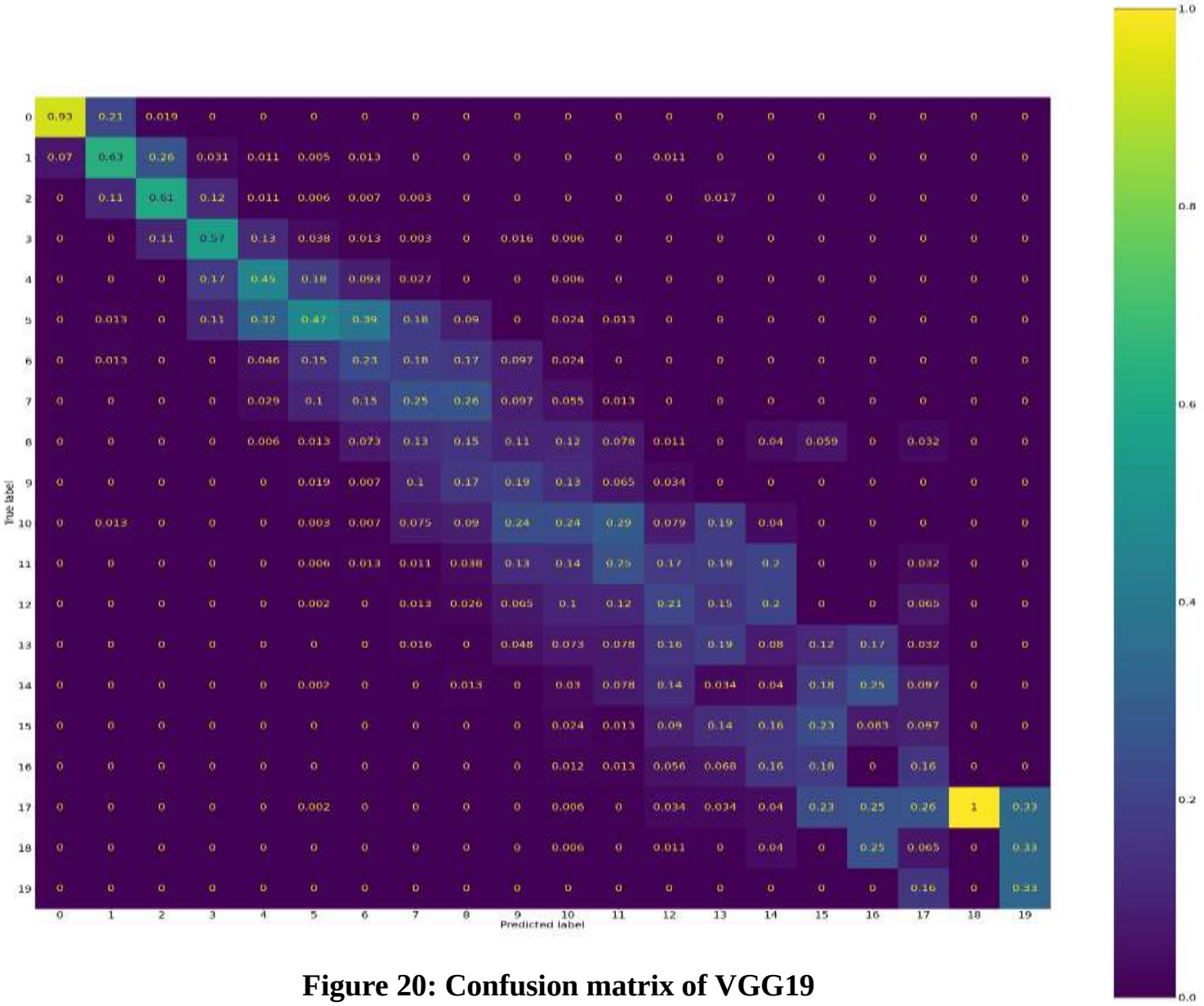


Figure 20: Confusion matrix of VGG19

⇒ The diagonal of the confusion matrix shows the number of correctly classified instances for each class, and by looking at the diagonal, we can see how the accuracy changes with each training iteration. If the accuracy decreases along the diagonal, it indicates that the model's classification ability is poor, and the overall accuracy is not high.

V. COMPLEXITY AND COMPARING MODEL

- The first model, which utilized the sigmoid activation function, had a 74% test-set accuracy. However, this model's training took a lengthy time—2 hours and 30 minutes—to complete.

- The second model, which made use of the RELU activation function, had a training time of just 1 hour and 15 minutes. This model's accuracy, at 29.21%, was much lower.

- The third model, which made use of the VGG 19 architecture, had a 40% test-set accuracy rate. This model's testing duration was barely an hour, despite the training period being 4 hours longer than the second models

⇒ Overall, it seems that the sigmoid model—which has the highest accuracy despite requiring more training time—might be the best choice for this particular task. If time is a concern, however, the RELU model can also be a wise choice because of its quick training period. However, because it requires significantly more training time and has significantly poorer accuracy than the other two models, the RELU model might not be the ideal option for this purpose. VGG19 can be a good choice in terms of timing for training but not a good choice in terms of accuracy.

VI. SOURCE CODE

1) Code plan for the model using sigmoid and RELU:

1.1 Process images

- Read images with greyscale preset
- Resize images to 70x70
- Reshape image to (70, 70, 1)

1.2 Split age group

- Split into 20 group
- Each group span 5 years
- Group 0 is from 0-4 years old
- Group 19 is from 95 years old to higher

1.3 Split the train-test data

- Split with an 80/20 ratio for train-test

1.4 Set up the model

- Using CNN 5 times from 32 channels to 512 channels
- Here we have 2 models: model predicting age and model predicting gender
- Both of them will use RELU activation when performing dense
- Each layer we drop out some nodes to prevent overfitting
- The last layer of model predicting age will use /RELU activation and return 1 value /sigmoid activation and return 20 values

1.5 Training step

- Here I use KFold cross validation with 10 folds to split the train set into small train and validate set but when training, the model will inherit the info of the previous trained model to keep training. Concept here is to train more, just using KFold to auto split train and validate set
- Train 2 times (for more accuracy):
  - First run with 15 epochs
  - Batch size 25 Training
  - Predict the age group and gender for each batch
  - Calculate the lost for each batch
  - Backpropagate
  - Gradient step
  - Decreasing the epoch by 1 after each fold to prevent overfitting
  - The last fold only contains 2 epochs to prevent overfitting
- Validation
  - Predict the age group and gender for each batch
  - Calculate the lost for each batch

1.6 Testing step

- Predict the age group and gender for the test set
- Calculate the accuracy
- Do some statistical analysis
- Plot the predict and real values lines
- Plot the confusion matrix

2) Code plan for model using VGG19:

2.1 Process images

- Resize images to 70x70 with 3-channel RGB
- Each color channel is rescaled to [0, 1]
- Normalize color channels using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]

2.2 Split age group

- Split into 20 group
- Each group spans 5 years
- Group 0 is from 0-4 years old
- Group 19 is from 95 years old to higher

2.3 Split the train-test data

- Split with a 90/10 ratio for train-test

2.4 Setup weight from each class in Cross entropy loss

- Each class has a weight in calculating loss function to compensate for the uneven age group in the dataset

2.5 Set up a model

- Import VGG19 model with pre-trained weights
- Change the last layer from 1000 groups to 20 groups

2.6 Training step

- Run for a maximum of 10 epoch
- Split the training in 9-fold for each
- Batch size 32
- Training
  - Predict the age group for each batch
  - Calculate the lost for each batch
  - Backpropagate
  - Gradient step
- Validation
  - Predict the age group for each batch
  - Calculate the lost for each batch

2.7 Testing step

- Predict the age group for each batch
- Calculate the lost for each batch

2.8 Conclusion

- Calculate the accuracy, the std of the different in predict age group and real age group
- Plot confusion matrix

Here is our source code for this project if you want to understand more clearly how we do it:

https://github.com/Tucker6742/CNN-sex-and-gender?fbclid=IwAR2Sc-DJ4v59F1Bzz4M0BJ_ezsBRGVPB5BS-sM-zQ2UleBWZv5sj7hrtOeU