CM50265 Machine Learning 2

Coursework 1: Age Estimation and Gender Classification

Links to the Two Models

age gender A.h5:
https://drive.google.com/file/d/1qRhFNyyDr_A1BwRY1aH3C_6FMrpa5LQn/view?usp=sharing

age gender B.h5:
https://drive.google.com/file/d/147ZxBu87Djw9Z1moFbKUL1Y8vF3ArEyJ/view?usp=sharing

Table of Contents

1. Introduction

In this coursework, two multi-task convolutional neural network (CNN) models which performs age and gender estimation is to be trained with 5,000 labelled face images from the UTKFace dataset. The first model is built from scratch while the second model is built by leveraging the technique of transfer learning. This report will describe the training process and performance of the two CNN models and conclude with a comparison of the two model building techniques explored.

2. Model A: My own CNN

2.1. Data pre-processing

A mapping table is created for each image in the dataset, as shown in Figure 2.1-1. It facilitates the creation of Keras ImageDataGenenerator instances for training and validation respectively. The split ratio of images allocated to training and validation generators is 85:15. Data augmentation is applied to the training generator, whereas rescaling of pixel values to [0, 1] is applied to both generators. The choice of augmentation is implemented with a forward-stepwise approach where augmentations were introduced sequentially and only those that improve validation performance are kept. The final choice of augmentations is listed in Table 2.1-1. Sample augmented images are visualised in Figure 2.1-2.

	Name	Age	Gender
0	40_0_3_20170119183403621.jpg.chip.jpg	40.0	0
1	75_0_0_20170117174511134.jpg.chip.jpg	75.0	0
2	18_1_0_20170109213011914.jpg.chip.jpg	18.0	1
3	70_1_3_20170116224931319.jpg.chip.jpg	70.0	1
4	26_1_1_20170116232602440.jpg.chip.jpg	26.0	1
...	...	...	...
4995	73_0_0_20170120230235659.jpg.chip.jpg	73.0	0
4996	21_0_1_20170114032109958.jpg.chip.jpg	21.0	0
4997	45_0_1_20170117181136802.jpg.chip.jpg	45.0	0
4998	63_1_0_20170110160643845.jpg.chip.jpg	63.0	1
4999	24_0_2_20170104234829387.jpg.chip.jpg	24.0	0

[5000 rows x 3 columns]

Figure 2.1-1: Mapping table for the dataset

Table 2.1-1 Model A: List of augmentations applied

Augmentation	Values
Rotation range	20
Brightness range	[0.2, 1.5]
Zoom range	0.1
Horizontal flip	True

Figure 2.1-2: Model A: Samples of augmented training images

2.2. Model architecture

To perform multi-task learning, a single shared pipeline is created after the input layer where convolutional, pooling and batch normalization operations are performed. The pipeline is then split into two branches which are responsible age and gender estimation respectively. Model A's architecture is inspired from VGGNet's idea of stacking multiple small convolutional filters for feature extraction. Dropouts are added after each dense layer to prevent overfitting. A higher dropout rate is assigned to the gender branch dense layers as we found the model tends to overfit on gender easier during training. The choice of model hyperparameters is detailed in section 2.3.

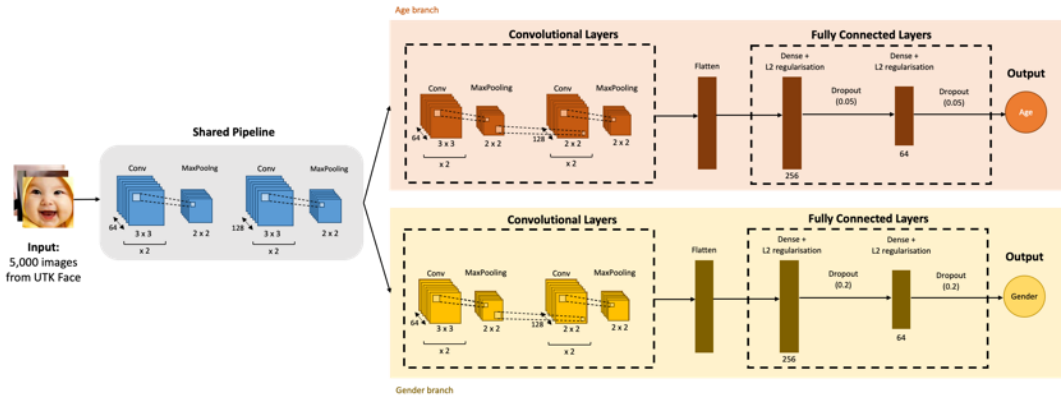

Figure 2.2-1: Model A: Illustration of the model architecture

```
Model: "modelA"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_14 (InputLayer)        [(None, 128, 128, 3       0           []
                             )]

conv2d_99 (Conv2D)           (None, 126, 126, 64       1792        ['input_14[0][0]']
                             )

conv2d_100 (Conv2D)          (None, 124, 124, 64       36928       ['conv2d_99[0][0]']
                             )

max_pooling2d_55 (MaxPooling2D  (None, 62, 62, 64)     0           ['conv2d_100[0][0]']
)

conv2d_101 (Conv2D)          (None, 60, 60, 128)       73856       ['max_pooling2d_55[0][0]']

conv2d_102 (Conv2D)          (None, 58, 58, 128)       147584      ['conv2d_101[0][0]']

max_pooling2d_56 (MaxPooling2D  (None, 29, 29, 128)    0           ['conv2d_102[0][0]']
)

conv2d_103 (Conv2D)          (None, 27, 27, 64)        73792       ['max_pooling2d_56[0][0]']

conv2d_107 (Conv2D)          (None, 27, 27, 64)        73792       ['max_pooling2d_56[0][0]']

conv2d_104 (Conv2D)          (None, 25, 25, 64)        36928       ['conv2d_103[0][0]']

conv2d_108 (Conv2D)          (None, 25, 25, 64)        36928       ['conv2d_107[0][0]']

max_pooling2d_57 (MaxPooling2D  (None, 12, 12, 64)     0           ['conv2d_104[0][0]']
)

max_pooling2d_59 (MaxPooling2D  (None, 12, 12, 64)     0           ['conv2d_108[0][0]']
)

conv2d_105 (Conv2D)          (None, 11, 11, 128)       32896       ['max_pooling2d_57[0][0]']

conv2d_109 (Conv2D)          (None, 11, 11, 128)       32896       ['max_pooling2d_59[0][0]']

conv2d_106 (Conv2D)          (None, 10, 10, 128)       65664       ['conv2d_105[0][0]']

conv2d_110 (Conv2D)          (None, 10, 10, 128)       65664       ['conv2d_109[0][0]']

max_pooling2d_58 (MaxPooling2D  (None, 5, 5, 128)      0           ['conv2d_106[0][0]']
)

max_pooling2d_60 (MaxPooling2D  (None, 5, 5, 128)      0           ['conv2d_110[0][0]']
)

flatten_19 (Flatten)         (None, 3200)              0           ['max_pooling2d_58[0][0]']
flatten_20 (Flatten)         (None, 3200)              0           ['max_pooling2d_60[0][0]']

dense_36 (Dense)             (None, 256)               819456      ['flatten_19[0][0]']

dense_38 (Dense)             (None, 256)               819456      ['flatten_20[0][0]']

dropout_50 (Dropout)         (None, 256)               0           ['dense_36[0][0]']

dropout_52 (Dropout)         (None, 256)               0           ['dense_38[0][0]']

dense_37 (Dense)             (None, 64)                16448       ['dropout_50[0][0]']

dense_39 (Dense)             (None, 64)                16448       ['dropout_52[0][0]']

dropout_51 (Dropout)         (None, 64)                0           ['dense_37[0][0]']

dropout_53 (Dropout)         (None, 64)                0           ['dense_39[0][0]']

age_output (Dense)           (None, 1)                 65          ['dropout_51[0][0]']

gender_output (Dense)        (None, 1)                 65          ['dropout_53[0][0]']

=================================================================================================
Total params: 2,350,658
Trainable params: 2,350,658
Non-trainable params: 0
_________________________________________________________________________________________________
```

Figure 2.2-2: Model A: Details of the model architecture

2.3. Model training

2.3.1. Grid search on model hyperparameters

Grid search on model hyperparameters are performed to determine the final model architecture. Due to time constraint, only the hyperparameters which we believe have the largest impact on the model performance were tested. Table 2.3-1 lists the hyperparameters and corresponding values tested during our model training process.

Table 2.3-1: Model A: Model hyperparameters values tested during training

Hyperparameters	Values tested
Number of convolutional blocks in the shared pipeline	0, 1, 2
Number of convolutional blocks in the age and gender branches	0, 1, 2
Kernel size of convolutional layer	2 x 2, 3 x 3
Number of filters in convolutional layer	64, 128, 256
Number of neurons in dense layers	64, 128, 256
Training batch size	32, 64, 128

2.3.2. Loss weights optimisation

To allow the model to optimise both the age and gender loss functions simultaneously, a linear combination of the loss functions for age and gender predictions is defined as the model's objective function. The sigmoid loss output for gender is always less than 1, while the age MSE output is usually much higher. Hence, to maintain a more balanced evaluation of the total loss without prioritising a particular task, a significantly higher weight is assigned to gender. The final objective function for model A assigns a weight of 1 for age and 100 for gender.

2.4. Model performance

Figure 2.4-1 illustrates the training and validation performance in 100 epochs with the final model architecture. The final model chosen is trained at epoch 85, which achieves the lowest validation age MAE and highest gender accuracy among all epochs. It achieves a validation gender accuracy of 89.40% and age MAE of 6.12. Figure 2.4-2 captures the output of the final 30 training epochs.

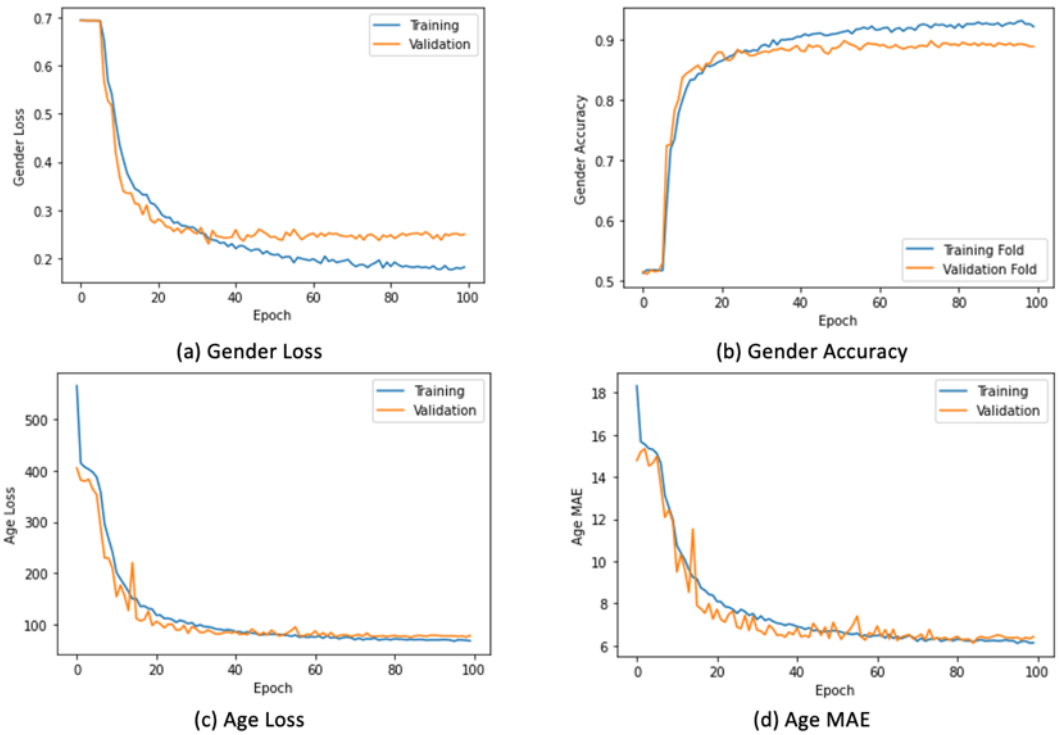

Figure 2.4-1: Model A: Final model performance

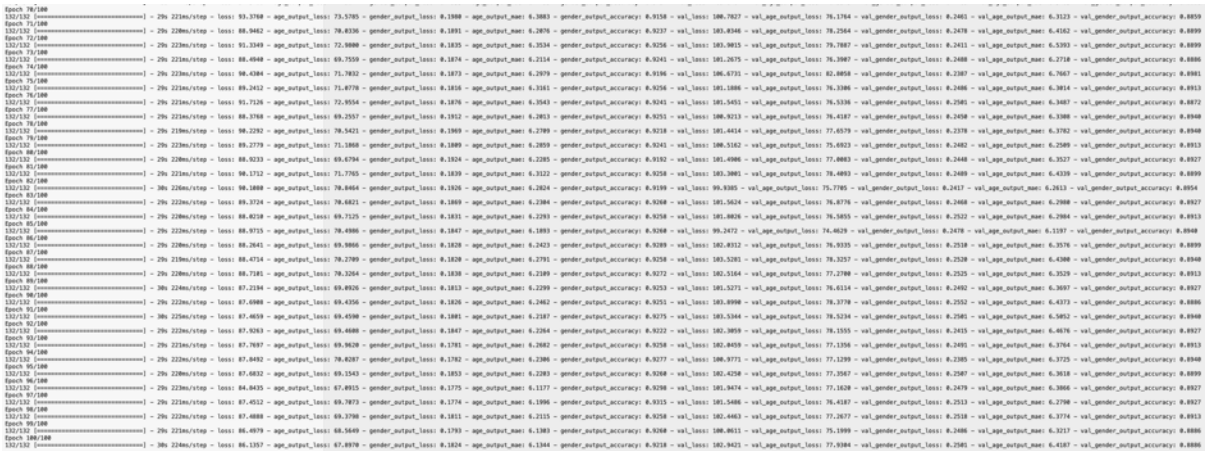

Figure 2.4-2: Model A: Screenshot of training output

3. Model B: Pre-trained CNN

3.1. Data pre-processing

The data pre-processing processes are identical to those performed for Model A, with minor differences in the augmentations applied. Table 3.1-1 lists the augmentations applied to the training images in Model B. Sample augmented images are visualised in Figure 3.1-1.Figure 2.1-2

Table 3.1-1 Model B: List of augmentations applied

Augmentation	Values
Rotation range	45
Horizontal flip	True
Colour contrast	[0.5, 2.5]

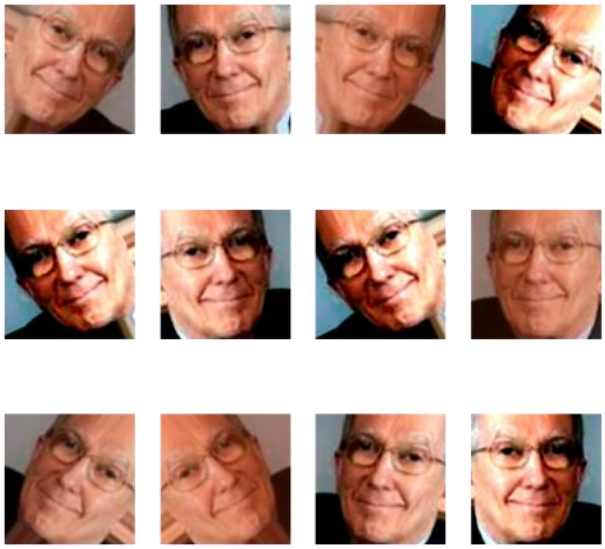

Figure 3.1-1: Model B: Samples of augmented training images

3.2. Model architecture

Our model leverages the DenseNet169 architecture pretrained on ImageNet. The layers within DenseNet are shared for both the age and gender estimations. The number of dense layers and dropout rates are selected through trial-and-error detailed in 3.3.1.

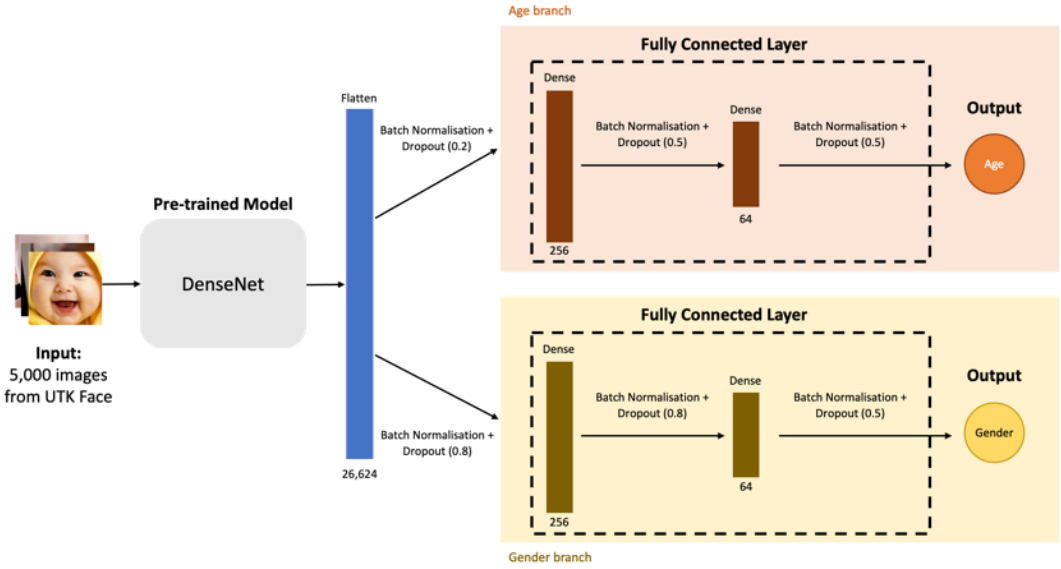

Figure 3.2-1: Model B: Illustration of the model architecture

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	[(None, 128, 128, 3)]	0	[]
densenet169 (Functional)	(None, 4, 4, 1664)	12642880	['input_3[0][0]']
flatten_1 (Flatten)	(None, 26624)	0	['densenet169[0][0]']
batch_normalization_5 (BatchNormalization)	(None, 26624)	106496	['flatten_1[0][0]']
dropout_6 (Dropout)	(None, 26624)	0	['batch_normalization_5[0][0]']
dropout_9 (Dropout)	(None, 26624)	0	['batch_normalization_5[0][0]']
age_fc1 (Dense)	(None, 256)	6816000	['dropout_6[0][0]']
gender_fc1 (Dense)	(None, 256)	6816000	['dropout_9[0][0]']
batch_normalization_6 (BatchNormalization)	(None, 256)	1024	['age_fc1[0][0]']
batch_normalization_8 (BatchNormalization)	(None, 256)	1024	['gender_fc1[0][0]']
dropout_7 (Dropout)	(None, 256)	0	['batch_normalization_6[0][0]']
dropout_10 (Dropout)	(None, 256)	0	['batch_normalization_8[0][0]']
age_fc3 (Dense)	(None, 64)	16448	['dropout_7[0][0]']
gender_fc2 (Dense)	(None, 64)	16448	['dropout_10[0][0]']
batch_normalization_7 (BatchNormalization)	(None, 64)	256	['age_fc3[0][0]']
batch_normalization_9 (BatchNormalization)	(None, 64)	256	['gender_fc2[0][0]']
dropout_8 (Dropout)	(None, 64)	0	['batch_normalization_7[0][0]']
dropout_11 (Dropout)	(None, 64)	0	['batch_normalization_9[0][0]']
age_output (Dense)	(None, 1)	65	['dropout_8[0][0]']
gender_output (Dense)	(None, 1)	65	['dropout_11[0][0]']

Total params: 26,416,962
Trainable params: 13,719,554
Non-trainable params: 12,697,408

Figure 3.2-2: Model B: Details of the model architecture

3.3. Model training

3.3.1. Feature extraction

The first stage of training is feature extraction, which is performed by freezing the pre-trained model and training only some dense layers appended to the model. To optimise the number of dense layers, we compared the model performances with two to four dense layers between the pre-trained model and the output layer. Other hyperparameters such as optimiser and learning rate were kept constant. It is found that the model with two dense layers added achieves the best validation performance with the least number of epochs. Hence, the structure is selected for our final model.

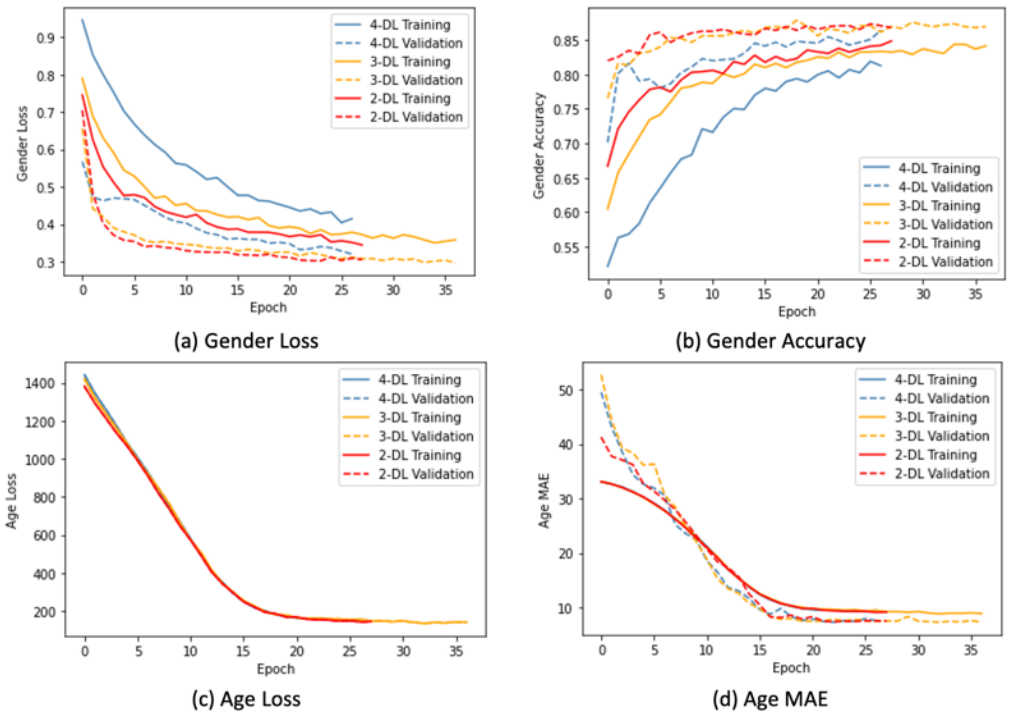

Figure 3.3-1: Model performance with different number of dense layers applied

To prevent overfitting, dropout layers are added after each dense layer. Through trial-and-error, it is found that gender tends to overfit easier than age. Thus, a higher dropout rate is applied after the first layer of the gender branch. Besides, batch normalisation layers are also added to normalise the inputs to each layer and provide more regularisation effect.

3.3.2. Fine-tuning

After feature extraction, fine-tuning is performed by unfreezing dense blocks within the pre-trained DenseNet to allow the model to acquire task-specific knowledge.

We compared models with different number of dense blocks unfrozen to optimise the number of blocks to fine-tune. The performances are illustrated in Figure 3.3-2. The models are trained with Adam optimiser with a slow learning rate of $1e^{-5}$ to avoid distorting the pre-trained weights. It is observed that the model with all layers fine-tuned has a highly fluctuating validation performance, while the model with only one layer fine-tuned learns significantly slower. Hence, we decided to opt for the structure with three dense blocks fine-tuned, which demonstrates a more stable validation performance.

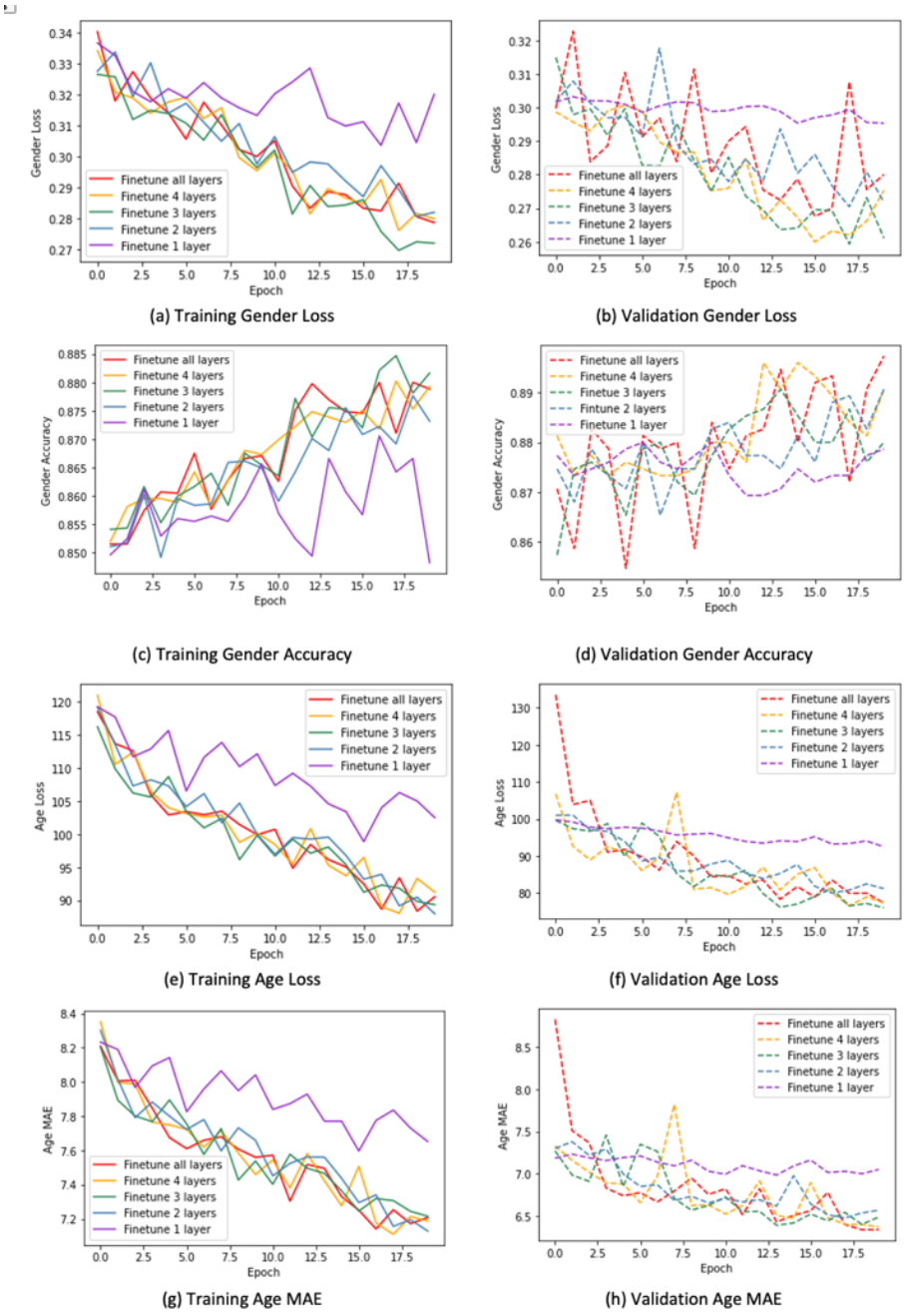

Figure 3.3-2: Model performance of the pre-trained CNN with different number of layers finetuned

3.3.3. Loss weights optimisation

Similar to model A, the optimal loss weight ratios for age and gender in model B's objective function is determined through random search. During feature extraction, the age loss is assigned a weight of 0.5, while the gender loss is assigned a weight of 300. In fine-tuning, the weight for age is 4, while the weight for gender is 250.

3.4. Model performance

Figure 3.4-1 shows the performance of running the model with three dense blocks unfrozen for 150 epochs. The improvement on both age and gender estimation started slows down significantly after epoch 80. The final model chosen is the model trained at epoch 143, which achieves a validation MAE of 5.75 and gender accuracy of 90.53%. The training output of the final 20 epochs are shown in Figure 3.4-2.

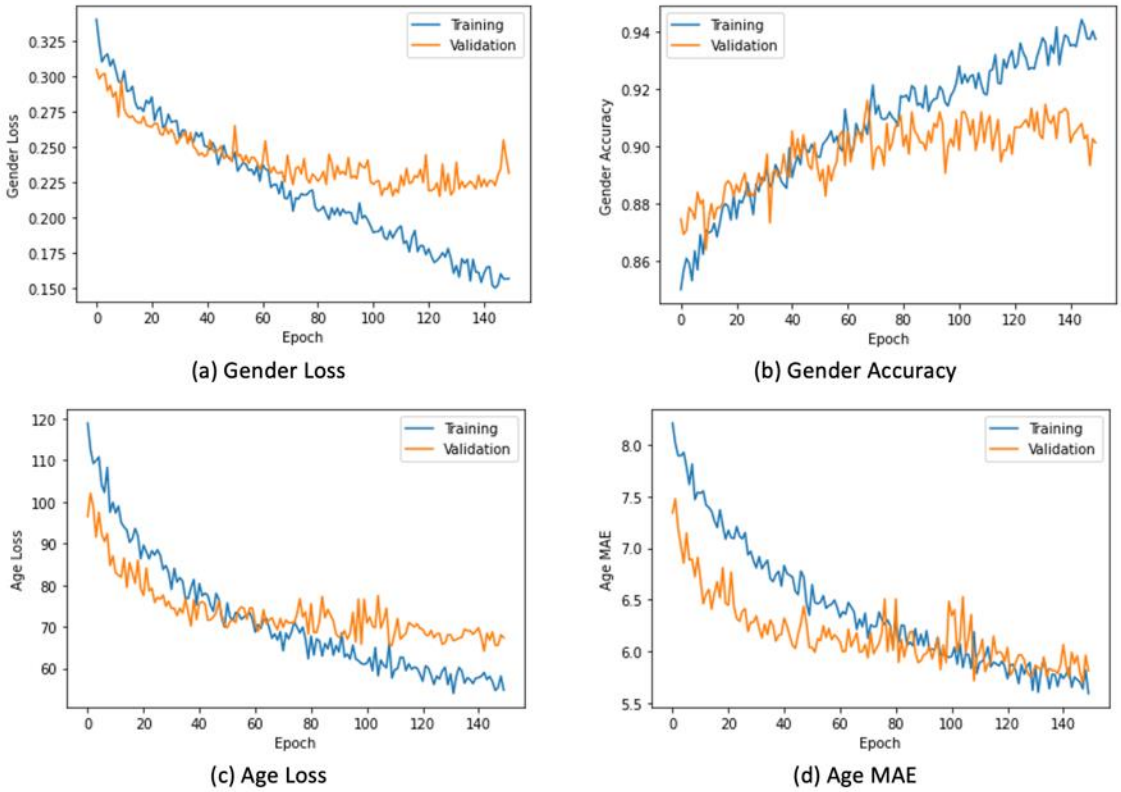

Figure 3.4-1: Model B: Model performance

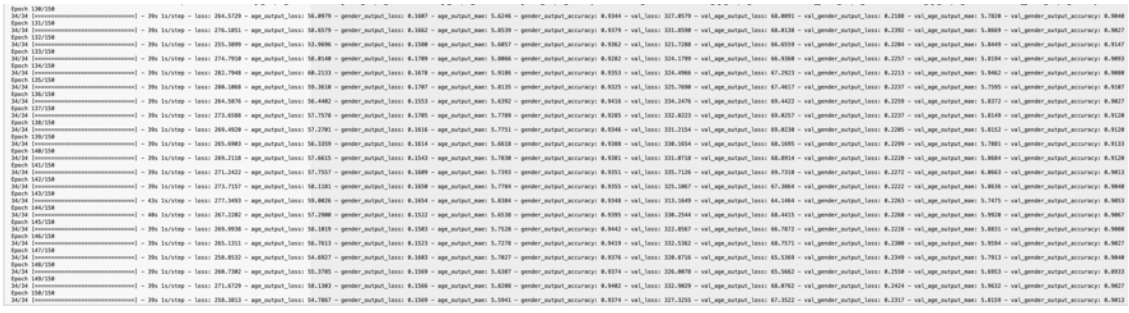

Figure 3.4-2: Model B: Screenshot of training output

4. Summary and Discussion

Two approaches of CNN model building were explored. Model A is built from scratch and required lots of trial-and-error on the architecture before a decent performance is achieved. On the other hand, Model B is built with transfer learning, which involves a pre-trained model in the architecture. By simply attaching and training fully connected layer on top of the existing model, its validation performance reaches the similar level as Model A's. After fine-tuning, the validation performance of the transfer learning model even surpasses that of Model A's. Hence, if there exists pre-trained models in a domain similar to the current problem, transfer learning might allow us to achieve a satisfactory result in a more efficient manner.