Assignment 2 Project D - Gender Classification

Authored by:

Wachakorn Rachanon (CE) U2023114D

Teo Guang Xiang (CE) U2020948F

Luo XiaoYang (CE) U2022803B

Content

1 Introduction

In recent years, deep learning for image recognition systems to recognize and classify human subjects has become increasingly widespread with its breakthrough in achieving high accuracy and precision results. Gender classification within image recognition systems is a subset which aims to study the facial features of a given human to predict one's gender diversity.

Whilst there are numerous experiments based on various state-of-the-art deep learning models used to classify human gender, in this report, we will investigate the model's success and attempt to improve their training time. We will be using the Adience Unfiltered faces for gender and age classification [1].

[1] (https://talhassner.github.io/home/projects/Adience/Adience-data.html)

2 Review of Existing Techniques

2.1 Convolutional Neural Network (CNN)

CNN is a family of neural networks that utilises convolution layers to process data. With the success of AlexNet (A. Krizhevsky, et al, 2012) on ImageNet Large Scale Visual Recognition Challenge 2012, CNN has become the state-of-the-art technique for image related tasks such as image classification, image generation, object detection, and etc. Over the past years there have been overwhelming improvements to the basic CNN which also improved their performance significantly.

2.2 MobileNet

MobileNet (Howard, et al, 2017) is an efficient CNN network architecture for mobile and embedded vision applications. It uses depth wise separable convolutions to significantly reduce the number of parameters, compared to networks with regular convolutions with the same depth. Depthwise separable convolutions will consist of Depthwise convolution followed by pointwise convolution. As a tradeoff for speed, some accuracy is sacrificed.

2.3 Residual Neural Network (ResNet)

ResNet (He, et al, 2015) is a CNN that stacks residual blocks on top of each other to form a deeply layered neural network. The researchers proposed a method to overcome the vanishing gradient problem by using skip connections, an alternative pathway for the gradient to pass, which allows for significantly deeper models and improves performance compared to prior works.

2.4 EfficientNet

EfficientNet (Tan & Le, 2019) is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. EfficientNet has shown that balancing the scale in all the three dimensions (width, depth and image resolution) best improves the overall model performance. With reduced parameter size and FLOPs by an order of magnitude, EfficientNet becomes the state-of-the-art to perform better than ResNet and MobileNet.

3 Experimentations

We will discuss in-depth details of our experiment implementations. We will be using the given reference (Face Image Project - Data, 2014) as a baseline for our experiments. Due to time and resource constraints, we trained the models by combining fold 0 to fold 3 into the training set and fold 4 as validation / testing set. Our experiments are splitted into 2 parts: transfer learning and model modification.

3.1 Image Data Pre-processing

Description

Adience dataset comes in 5 folds, which are specified in a formatted text file. The data for each fold is loaded according to the fold files given in the dataset. The images are loaded with RGB space and the labels are one-hot encoded with male class = 0 and female class = 1. We use data preprocessing steps similar to the baseline reference paper in order to reproduce their results, but with some modifications as the original implementation of the reference was in Caffe Framework.

For training, the following data preprocessing and data augmentation is performed

1. Shuffle and repeat dataset
2. Random crops to 227 x 227 pixels
3. Random horizontal and vertical flips
4. Random brightness with delta of 63/255
5. Random contrast with 0.2x to 1.8x range

For validation, all images are centre cropped to 227 x 227 pixels.

3.2 Baseline

Description

The baseline model is based on the paper "*Age and Gender Classification using Convolutional Neural Networks*" published by Gil Levi and Tal Hassner. The authors provided an official implementation in Caffe Framework. Since we are using Tensorflow Framework for our experiments, we have replicated their model using Tensorflow according to their Caffe implementation. For the baseline model, there is an extra preprocessing step which normalises image pixels globally to zero mean and one variance.

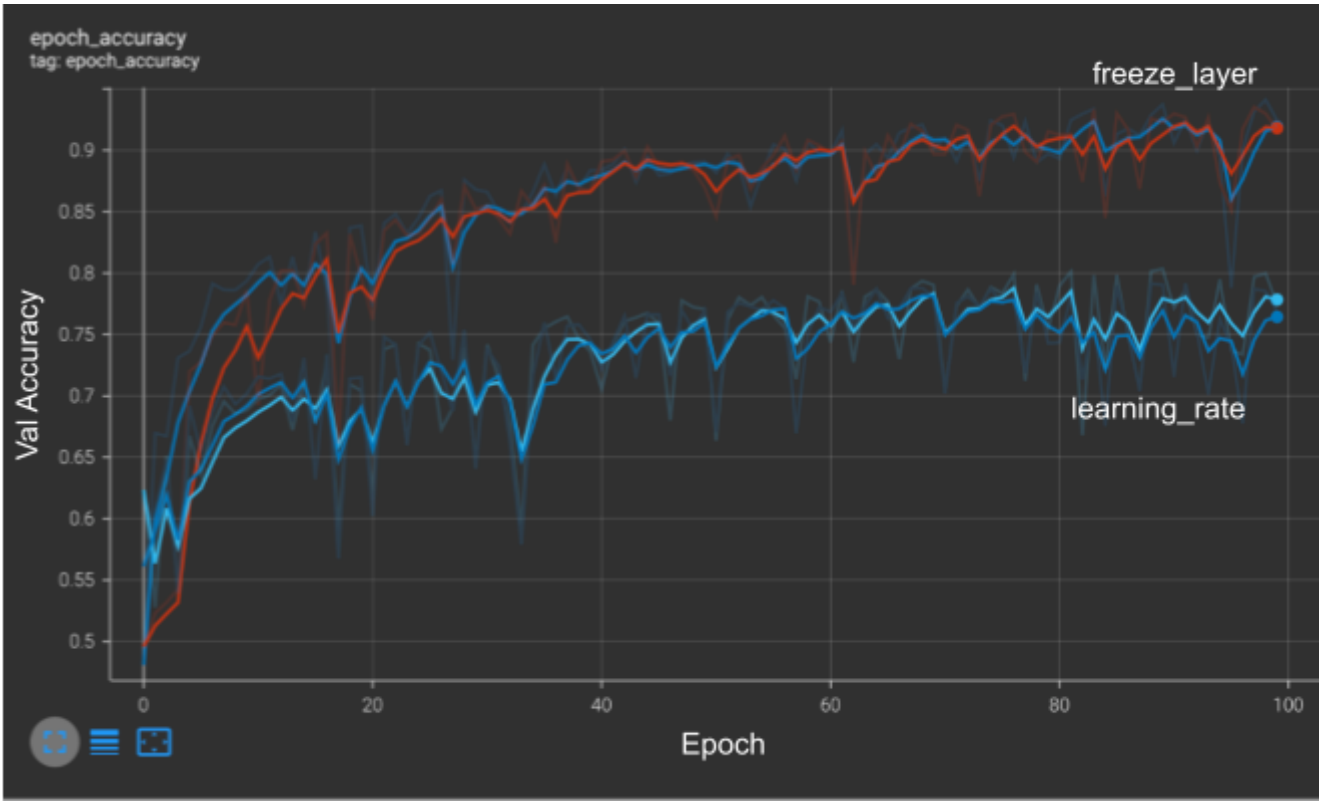
3.3 Transfer Learning

3.3.1 MobileNetV3

Description

In this implementation, we use the MobileNetV3Large model obtained from Tensorflow API. Initially, all layers were frozen and only the learning rate was altered to observe the performance of pre-trained weights. To further improve on the result, different values of frozen layers were experimented.

Results



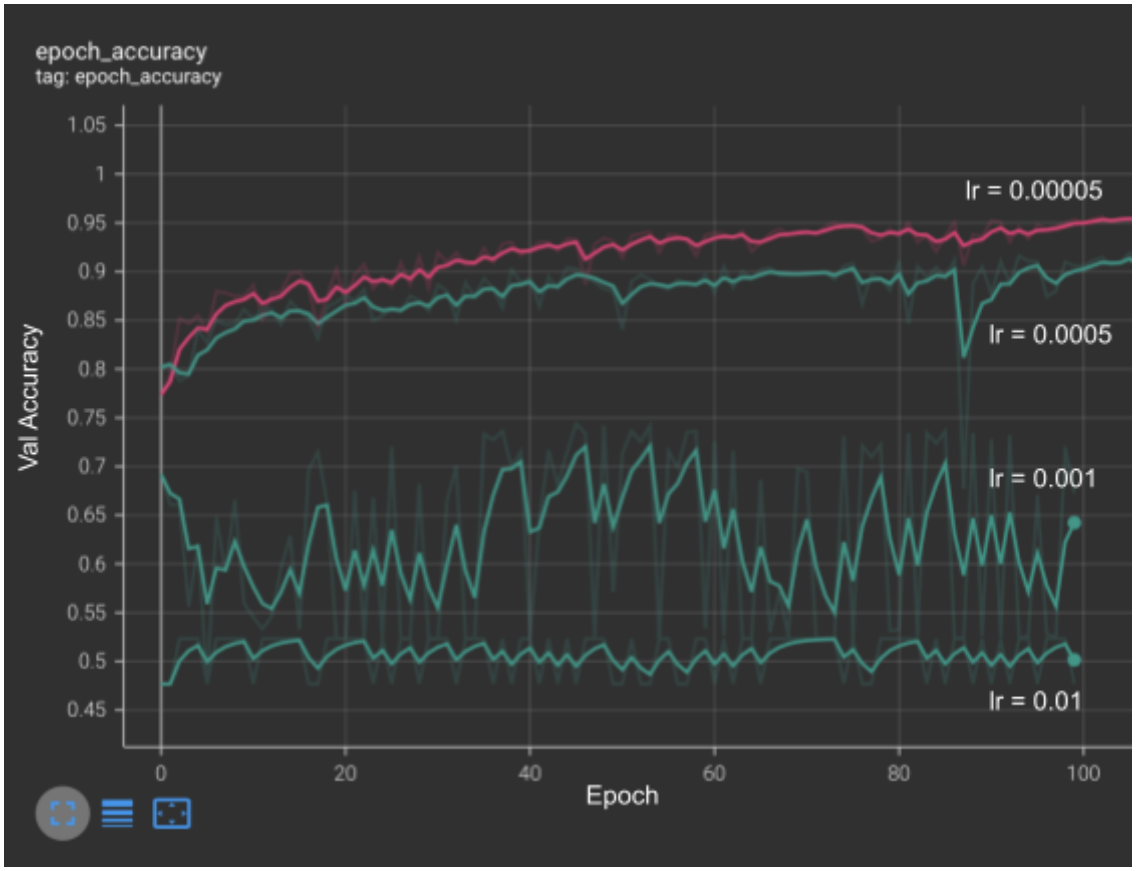
MobileNetV3	Category	# of Frozen Layers	Learning Rate	Epoch	Validation Accuracy
MobileNetV3_1	learning_rate	All	0.00005	100	0.7687
MobileNetV3_2	learning_rate	All	0.00001	100	0.7784
MobileNetV3_3	freeze_layer	10	0.00001	100	0.9314
MobileNetV3_4	freeze_layer	20	0.00001	100	0.9352

Explanation
When the learning rate is lowered, validation accuracy increases slightly, hovering around the value of 0.76-0.77. However, when the number of frozen layers is 20, validation accuracy increases significantly from 0.77 to 0.93. As the number of frozen layers further decreased, validation accuracy decreased instead.

3.3.2 ResNet50V2

Description
In this implementation, we use the ResNetV2 model obtained from Tensorflow API. We altered the learning rate to observe the performance of pre-trained weights.

Results



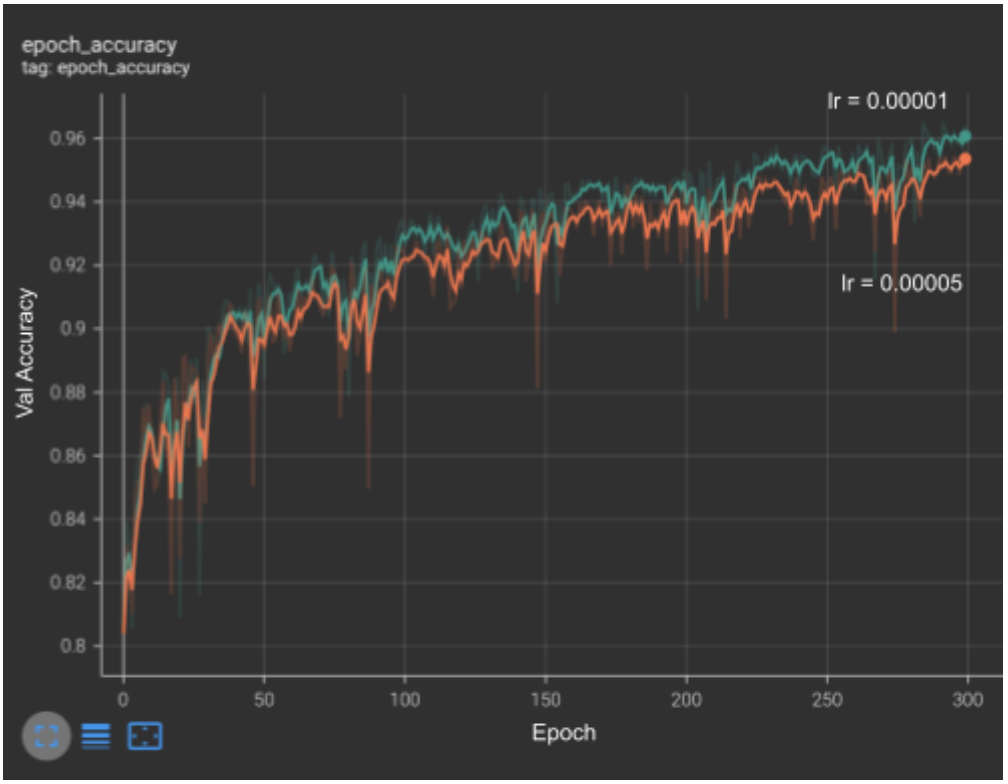
ResNetV2	Learning Rate	Epoch	Validation Accuracy
ResNetV2_1	0.001	100	0.6722
ResNetV2_2	0.01	100	0.4768
ResNetV2_3	0.0005	100	0.8676
ResNetV2_4	0.00005	100	0.9534

Explanation
When the learning rate is lowered from 0.01 to 0.00005, validation accuracy increases significantly from 0.47 to 0.95.

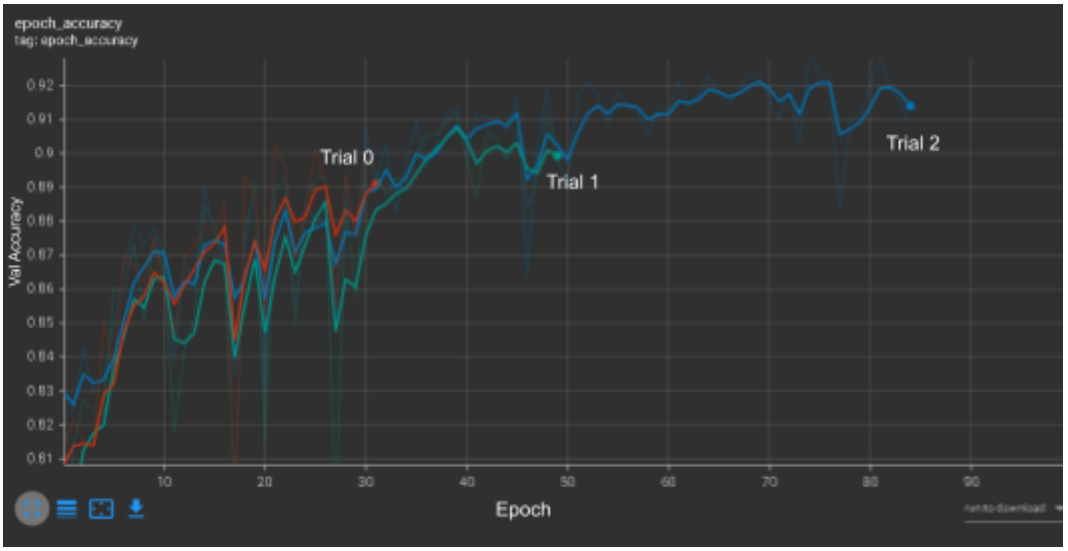
3.3.3 EfficientNetV2

Description
In this implementation, we use the EfficientNetV2M model obtained from Tensorflow API. We altered the learning rate to observe the performance of pre-trained weights. Additionally, a random search experiment was carried out to discover optimal latent size and learning rate with an early stopping of patience 10.

Results



EfficientNetV2	Learning Rate	Latent Size	Epoch	Validation Accuracy
EfficientNetV2_1	0.00005	128	300	0.9561
EfficientNetV2_2	0.00001	128	300	0.9641



EfficientNetV2 Random Search	Learning Rate	Latent Size	Epoch	Highest Validation Accuracy
Trial 0	0.000017546	672	31	0.90284
Trial 1	0.000040957	320	49	0.91205
Trial 2	0.000010032	320	84	0.93033

Explanation

When the learning rate decreased from 0.00005 to 0.00001, validation accuracy increased slightly from 0.95 to 0.96.

For random search, in a max trial of 3, the best latent size and learning rate are 320 and 0.000010032 respectively.

3.3.4 Discussion

Learning Rate

Learning rate determines how much the model changes in response to the model error. Large learning rate may cause learning of sub-optimal weights and unstable training while too small learning rate causes long training process or model to get stuck. In all of our experiments with MobileNet, ResNet50 and EfficientNet, the optimal learning rate is approximately 0.00001.

Number of Frozen Layers

Varying number of frozen layers gives permission to pre-trained models to acclimatise to features in the actual training data. However, a too low number of frozen layers does not give time for pre-trained models to sufficiently conduct feature extraction from the training data. This is evident in our experiment with MobileNet where lower validation accuracy is reported for a low number of frozen layers.

Maximum Trials of Random Search

Due to resource and time constraint, a maximum trial of 3 is implemented for EfficientNet tuning. Given a higher number of trials, there is a strong possibility of finding the optimal parameters that would yield the best result.

Advantages and DIsadvantages of Transfer Learning

Benefits of transfer learning includes reducing the need for massive amounts of data to be prepared, lesser time and resources for training deep learning models. The major drawback of transfer learning would be if pre-trained data and actual data have many dissimilarities in nature, which would cause differences in validation and poor accuracy as a consequence.

Application of Transfer Learning in real world

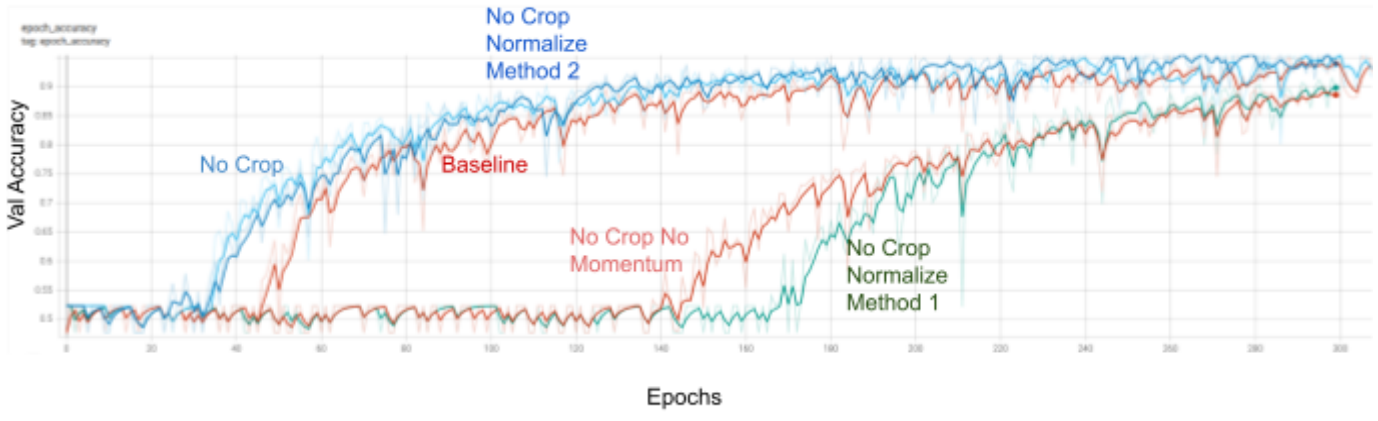
Having enough time and resources, as well as massive amounts of data are often hurdles for training deep learning models. Transfer learning comes in handy in many real world circumstances when there are limited amounts of the aforementioned. Today, one useful example of transfer learning is medical imaging, for detecting kidney issues in an ultrasound using models trained on ImageNet.

3.4 Model Components

3.4.1 Description

For this experiment, we tweak different components of the baseline model to see which factors contribute to the success of the baseline model in order to get a better understanding of the model and attempt to improve. By investigating which components of the baseline model contribute to its success, we may have a direction to investigate to attempt to improve the model. Some aspects we thought of improving were to improve the training time or convergence (in terms of epochs), to reduce the model's size, or to reduce computation cost.

3.4.2 Results and Discussion



Model Changes	Val Accuracy Epoch 300	Description
Baseline	0.9305	Baseline
No Crop	0.9659	Removed random crop and used the entire 256 x 256 as input.
No Crop, No Momentum	0.8839	No Crop and removed momentum from SGD optimizer
No Crop, Normalise Method 1	0.9011	No Crop and change the per image normalisation to normalisation to [-1, 1] range (maps [0, 255] to [-1,1])
No Crop, Normalise Method 2	0.9393	No Crop and change the per image normalisation to per channel normalisation to 0 mean 1 variance (mean and variance scaling adapted from training data)

From this experiment, the factor that improved the baseline the most is the centre crop removal. This means we used the entire 256 x 256 of the image as the input to the model instead. The increase in input size provides the model with more information to classify the image with. This makes the model converge earlier in terms of epoch, but the performance of the baseline reaches a similar level with sufficient training epochs. The increase in the input size has the tradeoff of increasing the amount of model's parameter and computation cost. The sizes from ~8 Million parameters in the baseline to ~11 Million parameters in the No Crop model. The investigation for cropping preprocessing was to determine if the crop is needed for the model to perform well, as the dimension of the image is not a power of 2 due to the cropping step, it may restrict some model architecture to be applied.

The removal of the momentum very negatively affects the model's performance. Momentum helps the optimizer escape local minima as well as help the model converge faster, without momentum the model only reaches 0.88 validation accuracy after 300 epochs of training. The investigation for momentum in the optimizer was to try to see whether changing to another optimizer would be possible. Another popular choice of optimizer besides SGD is Adam optimizer which may provide better training time. But seeing that removing the momentum from the baseline setup negatively affects the model's performance, it may be hard to find suitable parameters for the Adam optimizer which give an equivalent learning schedule to the SGD set up, which is very effective. We have done some experiments with Adam optimizer but failed to find suitable parameters and the results were inconclusive on whether Adam optimizer would help improve the model training.

Next we compared different methods of image preprocessing. We compared 3 different methods: per image normalisation, normalise to [-1, 1] range, and normalise to 0 mean and 1 variance. Per Image normalisation computes the mean and variance of all the pixels of each image individually, then normalises all the pixel values in all channels to achieve a global mean of 0 and variance of 1. Normalising to [-1, 1] range is done by mapping from [0, 255] to [-1, 1] by the equation image_normed = (image / 127.5) - 1.0. Normalisation to 0 mean and 1 variance is done by calculating the data mean and variance of each colour channel on the training set then for each image, image_normed = (image - mean) / variance. From the results, the normalisation to [-1, 1] range performs the worst with only 0.88 validation accuracy after epoch 300. It might have failed to capture the variability in the image and their central values due to static normalisation with maximum uint8 value. The per colour channel normalisation to 0 mean and 1 variance perform similarly to the per image normalisation used in the baseline reference. Even though the mean and variance of this method is calculated once using the training set, the performance is similar to calculating the statistics for each image individually. The advantage of this implementation is it is able to capture the variability (signals/information) in the image more meaningfully than normalisation to [-1. 1] range while only performing statistics calculation once compared to the per image normalisation. Since statistics calculating requires reduction steps, it may not be very efficient to parallelize using GPU hardware. The investigation for different image processing methods is to see which preprocessing method is best suited for this image classification task. The normalisation methods experimented are both popular methods used. The normalisation to [-1, 1] range was an early normalisation method which is also used for other types of bounded data. While the per channel normalisation are only seen more recently in literature, and it is shown through the experiment that the per image normalisation and per channel normalisation are way better suited for this image classification task.

3.5 Improvement Attempts

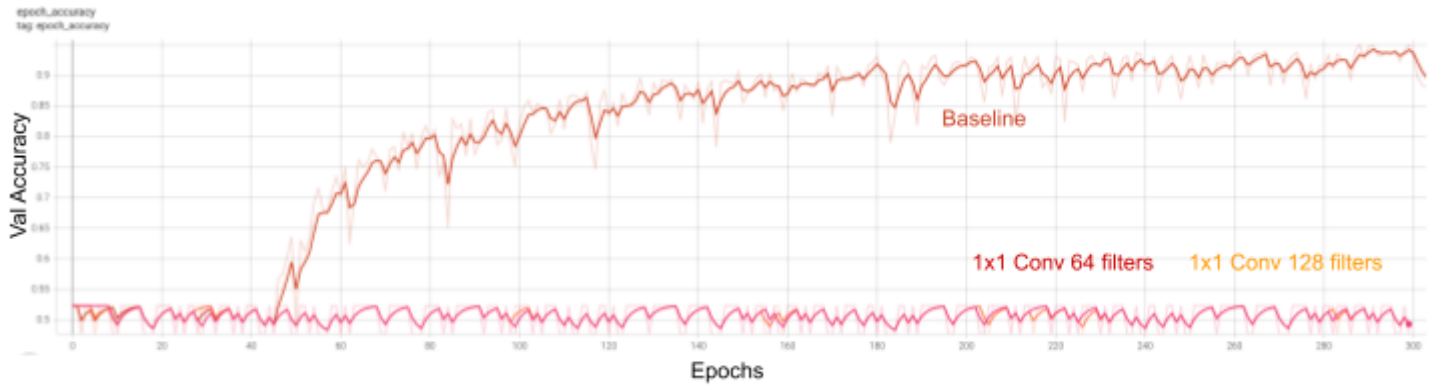
3.5.1 Description

We aim to improve the training time (in epochs) and to reduce the model's size. We must analyse the current model architecture to determine where it could be improved. The model used in baseline is a CNN where it uses convolution layers to extract image local features, then it uses fully connected layers to perform classification on the detected features.

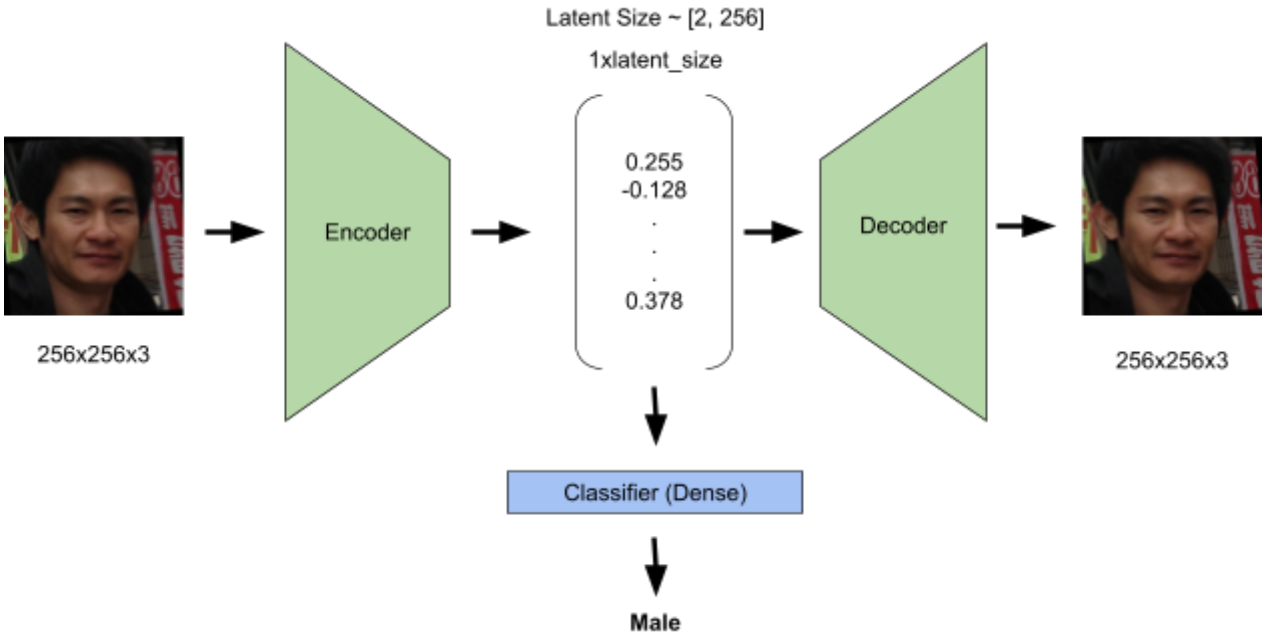
```
Model: "baseline_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 227, 227, 3)]     0

 conv1 (Conv2D)              (None, 56, 56, 96)        14208

 pool1 (MaxPooling2D)        (None, 27, 27, 96)        0

 tf.nn.local_response_normal  (None, 27, 27, 96)       0
 ization (TFOpLambda)

 tf.compat.v1.pad (TFOpLambd  (None, 31, 31, 96)       0
 a)

 conv2 (Conv2D)              (None, 27, 27, 256)       614656

 pool2 (MaxPooling2D)        (None, 13, 13, 256)       0

 tf.nn.local_response_normal  (None, 13, 13, 256)      0
 ization_1 (TFOpLambda)

 tf.compat.v1.pad_1 (TFOpLam  (None, 15, 15, 256)      0
 bda)

 conv3 (Conv2D)              (None, 13, 13, 384)       885120

 pool3 (MaxPooling2D)        (None, 6, 6, 384)         0

 tf.nn.local_response_normal  (None, 6, 6, 384)        0
 ization_2 (TFOpLambda)

 flat1 (Flatten)             (None, 13824)             0

 fc1 (Dense)                 (None, 512)               7078400

 do1 (Dropout)               (None, 512)               0

 fc2 (Dense)                 (None, 512)               262656

 do2 (Dropout)               (None, 512)               0

 fc3 (Dense)                 (None, 2)                 1026

=================================================================
Total params: 8,856,066
Trainable params: 8,856,066
Non-trainable params: 0
```

3.5.2 Pointwise Convolution

From the model summary, the majority of the weights and computation are in the connection between the feature extractor and the classifier. We can try to decrease the size of the extracted features in order to reduce the model's size. One of the possible methods is to use a point-wise convolution layer or convolution layer with 1x1 kernel size to reduce the feature space to a lower dimension. We attempted training a few of these models but with time constraints we were not able to find suitable parameters to achieve any meaningful results. The model was not able to learn with the current set of parameters and accuracy doesn't change from the start. The model details can be found in the code.

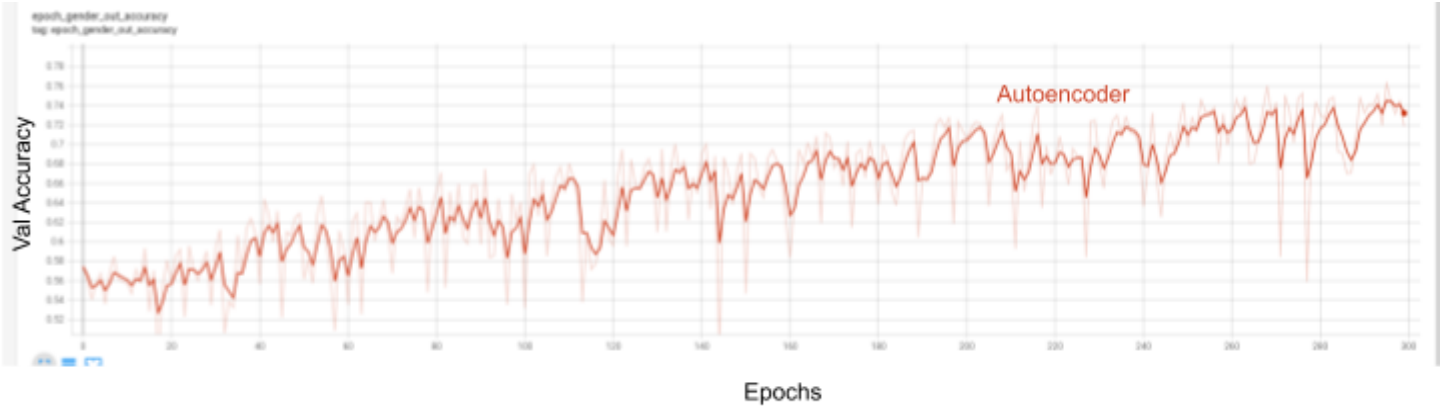


3.5.3 Autoencoder

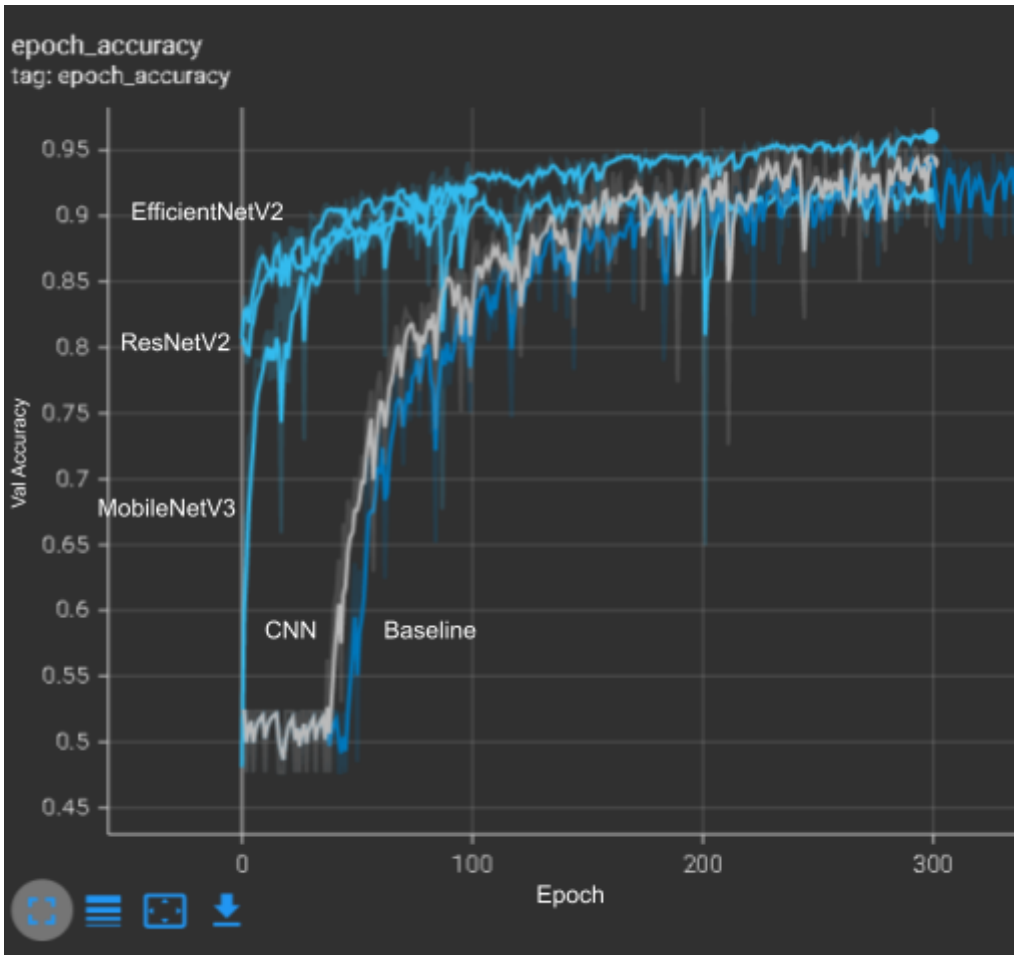


Due to the large feature set output from the feature extractor, the connection between the feature extractor and classifier is very large and costly. Autoencoder is a class of neural network which is trained to encode data into a small feature space. It consists of an encoder and a decoder portion, the encoder takes an input image and encodes it into a small feature (or latent) space, then the decoder takes the encoded representation as input and tries to reconstruct the output. Since the encoded representation must contain enough meaningful information to reconstruct the input, then it may be possible to classify the image based on its encoded representation which is a small set of values. By classifying on a small set of values, it effectively reduces the connection from the feature input to the classifier, decreasing the model size as well as computation cost. For classification inference, the decoder can also be removed to reduce unnecessary computations. Furthermore by training the classifier jointly with the encoder-decoder pair, the gradients may combine and help the model converge with lower iterations. (Kingma & Welling, 2014)

For autoencoders, we have also added more data augmentation for training such as random rotate, translation, shear, and noises for a more diverse training set set. But due to time and resource constraints, we were not able to find parameters for the model yet. The best model was only able to achieve a validation accuracy of 0.73. The model shown has LATENT_SIZE = 128, with total parameter of ~3 Million, even though the model was able to vaguely reconstruct the image, the encoded features may not capture the gender data very meaningfully to allow the classifier to distinguish between male and female subjects. Currently the encoder is just a simple CNN feature extractor but it may be possible to use the gender label to guide the learned representation in a more meaningful way than through the classifier loss.



4 Conclusion



Model	Epoch	Validation Accuracy
Baseline	300	0.9498
Modified Baseline (CNN)	300	0.9517
MobileNetV3	100	0.9244
ResNet50V2	300	0.9213
EfficientNetV2	300	0.9641

The graph and table above shows the best models in each category of experiments we executed. We first reproduced the results from a reference baseline paper. We then applied transfer learning to observe other state-of-the-art models performances on the chosen dataset. MobiletNetV3 was tested for ideal learning rate which at peak gave mediocre accuracies and then experimented with freezing of layers which reasoned for a significant accuracy boost. ResNetV2 performed with consistency relative to its learning rate and a good learning rate produced subsequently good accuracy. EfficientNetV2, built with compound scaling of CNN, established even better and consistent results with adjustment to learning rate causing slight increase in accuracy. In the end, all the transfer models tuned perform relatively well to generate almost similar accuracies, albeit with differences in terms of model configuration, training time and its respective application purposes. Thus showing that the gender classification from frontal image is a solved problem. We then investigated the baseline reference model to utilise as a starting point for improvement due to low model size. After gaining insights into the baseline model components, we analyse to find potential direction for improvements in model size and training time (in epoch). We suggested and experimented with two different potential methods for improvement to the baseline model. One being to reduce the model size by using pointwise convolution to reduce the feature space size and the other being to use autoencoder architecture to encode the features to a small latent space then perform classification using latent space representation. Overall, due to time and resource constraints we were not able to search for suitable parameters for the models suggested to give a conclusive result. Future works may include adjusting the parameters and architecture on the pointwise and autoencoder-based models to give meaningful results. There are also many variations of feature extraction such as the inception module proposed by GoogleNet(Zsegedy, et al, 2014) and many variants of autoencoder architecture such as variational autoencoder (Kingma and Welling, 2014) which may yield better results.

5 Reference

1. Boesch, G. (2022, June 29). *Deep Residual Networks (ResNet, ResNet50) - 2022 Guide - viso.ai*. Viso.ai. https://viso.ai/deep-learning/resnet-residual-neural-network/

2. Pujara, A. (2020, July 4). MobileNet Convolutional neural network Machine Learning Algorithms | Analytics Vidhya. Medium; Analytics Vidhya. https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470

3. Tan, M., & Le, Q. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. https://arxiv.org/pdf/1905.11946.pdf

4. Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., & Adam, H. (2019). Searching for MobileNetV3. *ArXiv.org*. https://doi.org/10.48550/arXiv.1905.02244

5. Szegedy, C., Vanhoucke, V., Ioffe, S., & Shlens, J. (n.d.). Rethinking the Inception Architecture for Computer Vision. https://arxiv.org/pdf/1512.00567v3.pdf

6. Kingma, D., & Welling, M. (2014). Auto-Encoding Variational Bayes. https://arxiv.org/pdf/1312.6114.pdf

7. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90. https://doi.org/10.1145/3065386