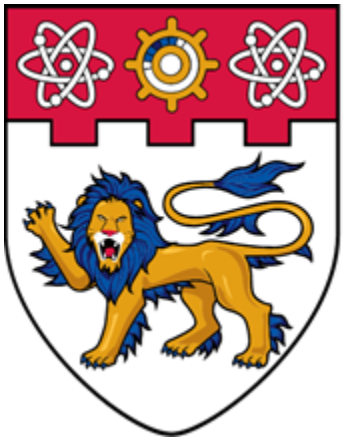
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY



NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4042 Neural Network & Deep Learning (Assignment 2)

Project Report

Name:

S Sri Kalki (U1921575L)

Daniel Loe (U1921408A)

Park Kunyoung (U1821171H)

AY 2021-22 SEMESTER 1

DATE OF SUBMISSION:

15 NOVEMBER 2021

Table of Contents

1. Introduction

Gender classification plays an important role in modern society, we see it in the way of how they are utilized for security and forensic purposes in the way of facial recognition. Additionally, given the rise of social media and social media platforms, social media marketing which is an extension of personalized advertising utilizing demographic and gender info is also on the rise. As the name implies, gender classification is a process whereby a system receives input(s) of a face of a person from a given image and tries to determine the gender of the given person. Current methods do exist already for this classification task however, given its widespread usages and accuracy of the classification being of utmost importance, more could be done to increase the prediction accuracy of a model.

1.1. Project Scope

The main goal of this project thus aims to classify the gender of faces in an image with the highest possible accuracy by means of optimizing model selection and the tuning different hyperparameters.

1.2. Dataset

1.2.1. Adience Dataset

For testing the accuracy of our model, there was an option between 2 datasets from the adience data, one being the faces dataset where it contained face images that were cropped and the second being the aligned dataset where it contained face images that were cropped and aligned. For the purpose of accuracy, we chose the aligned dataset as having faces which were already preprocessed to be aligned will substantially boost the performance of our training and prediction. The 5 different fold_frontal_data.txt (from 0-4) provides us the information we need to do our data preprocessing.

The aligned dataset initially containing 26,580 photos and 2284 subjects was reduced to 12,194 after data preprocessing removing incorrectly labelled records and removing records which did not have a class label f (female) or m (male).

1.2.2. CelebA Dataset

CelebA Dataset is a large-scale face attributes dataset with the following features:
- 202599 face images
- 40 binary attributes annotations for images

1.3. Data Pre-processing

Adience Dataset

To load the aligned dataset into the model for training, we first perform the data preprocessing as follows:

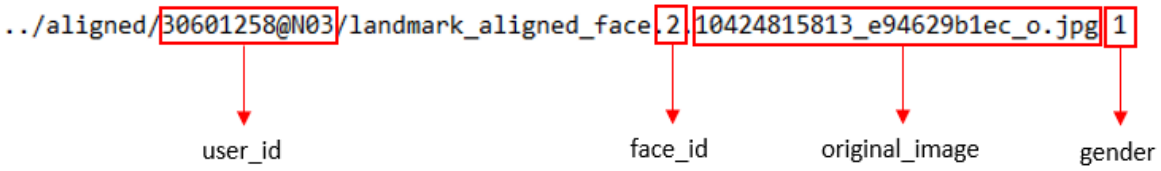
1) Load the 5 different fold_frontal_data.txt files

2) Loop through each fold and concatenate each ‘user_id’, ‘face_id’, ‘original_image’ into a link which identifies the absolute path of where each image is located and change gender label m and f into binary form (0 for f, 1 for m), E.g.

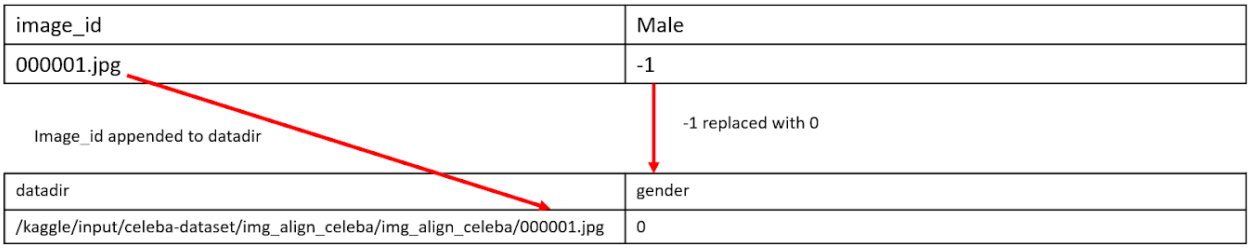
Data from fold_data file:

user_id	original_image	face_id	gender
30601258@N03	10424815813_e94629b1ec_o.jpg	2	m

Output: Path of Image to be used in program



CelebA Dataset



Attributes other than face images are dropped and the target column is set to ‘gender’. The train to test ratio is set to 0.1.

1.4. Libraries

- Wandb
- Tensorflow
- Pandas
- Numpy

2. Review of Existing Techniques

To select the best model to use for the gender classification task, we perform some comparison with several existing model architectures. All training done was done with a 80% train and 20% test data split.

2.1. Hassner CNN

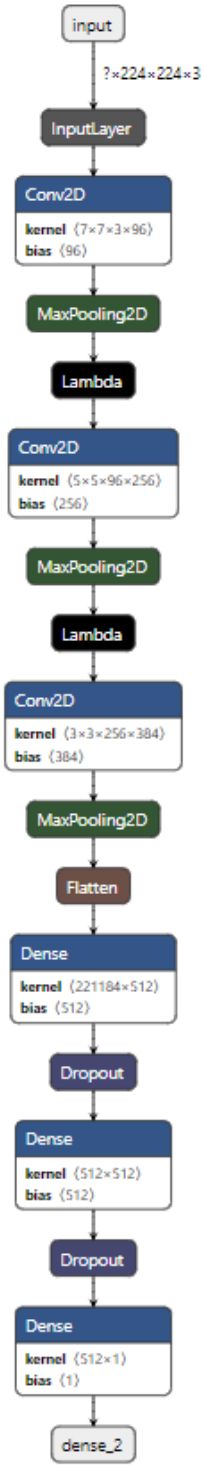


Figure 1: Network Diagram of Hassner Model

Similar to the research paper provided, we first evaluate their model architecture as described by the figure 1

Network Architecture generalized as below:

3 convolutional layers with differing filter, pooling and stride size with rectified linear (ReLu) hidden activation function

Followed by 2 custom fully connected layers with dropout 0.5 and ReLu hidden activation function

A last fully connected layer with sigmoid activation function mapping to the final classes for gender

Configuration Used

```
configs = {
    'epochs': 20,
    'batch_size': 64,
    'seed': 7,
    'learning_rate': 1e-3, #0.001
    'hidden_activation': 'relu',
    'output_activation': 'sigmoid',
    'optimizer': 'adam',
    'loss_function': 'binary_crossentropy',
    'metrics': ['accuracy'],
    'fc_layer_1_neurons' : 512,
    'fc_layer_2_neurons' : 512,
}
```

After training the model, we evaluated its accuracy with model.evaluate() and obtain a validation accuracy of 0.8798688054084778
≈ 0.88 / 88%

2.2. InceptionV3 Model

```
from tensorflow.keras.applications.inception_v3 import preprocess_input
train_image_generated = ImageDataGenerator(preprocessing_function = preprocess_input)
test_image_generated = ImageDataGenerator(preprocessing_function = preprocess_input)
```

The InceptionV3 model will first perform pre-processing of the image input before generating the train and test split. The top inception_v3 layer then accepts the processed inputs of shape (299, 299, 3).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Functional)    (None, 2048)              21802784
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
batch_normalization_94 (Batc (None, 2048)              8192
_________________________________________________________________
dense (Dense)                (None, 1024)              2098176
_________________________________________________________________
batch_normalization_95 (Batc (None, 1024)              4096
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
batch_normalization_96 (Batc (None, 512)               2048
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
batch_normalization_97 (Batc (None, 256)               1024
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 257
=================================================================
Total params: 24,572,705
Trainable params: 2,762,241
Non-trainable params: 21,810,464
```

<u>Network Architecture generalized as below:</u>

InceptionV3 as the first layer using pretrained weights from 'imagenet' and average pooling

Followed by 3 fully connected layers, each with a ReLu hidden activation and a batch normalization layer

A last fully connected layer with sigmoid activation function mapping to the final classes for gender

Configuration Used

```
configs = {
    'epochs': 20,
    'batch_size': 64,
    'seed': 7,
    'learning_rate': 1e-3, #0.001
    'hidden_activation': 'relu',
    'output_activation': 'sigmoid',
    'optimizer': 'adam',
    'loss_function': 'binary_crossentropy',
    'metrics': ['accuracy'],
    'fc_layer_1_neurons' : 1024,
    'fc_layer_2_neurons' : 512,
    'fc_layer_3_neurons' : 256,
}
```

After training the model, we evaluated its accuracy with model.evaluate() and obtain a validation accuracy of 0.876588761806488 **≈ 0.88 / 88%**

2.3. Resnet50V2 Model

```
from tensorflow.keras.applications.resnet_v2 import preprocess_input
train_image_generated = ImageDataGenerator(preprocessing_function = preprocess_input)
test_image_generated = ImageDataGenerator(preprocessing_function = preprocess_input)
```

The Resnet50v2 model will first perform pre-processing of the image input before generating the train and test split. The top resnet50V2 layer then accepts the processed inputs of shape (224. 224. 3).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50v2 (Functional)      (None, 2048)              23564800
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
batch_normalization (BatchNo (None, 2048)              8192
_________________________________________________________________
dense (Dense)                (None, 1024)              2098176
_________________________________________________________________
batch_normalization_1 (Batch (None, 1024)              4096
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
batch_normalization_2 (Batch (None, 512)               2048
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
batch_normalization_3 (Batch (None, 256)               1024
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 257
=================================================================
Total params: 26,334,721
Trainable params: 2,762,241
Non-trainable params: 23,572,480
```

Network Architecture generalized as below:

Resnet50V2 as the first layer using pretrained weights from 'imagenet' and average pooling

Followed by 3 fully connected layers, each with a ReLu hidden activation and a batch normalization layer

A last fully connected layer with sigmoid activation function mapping to the final classes for gender

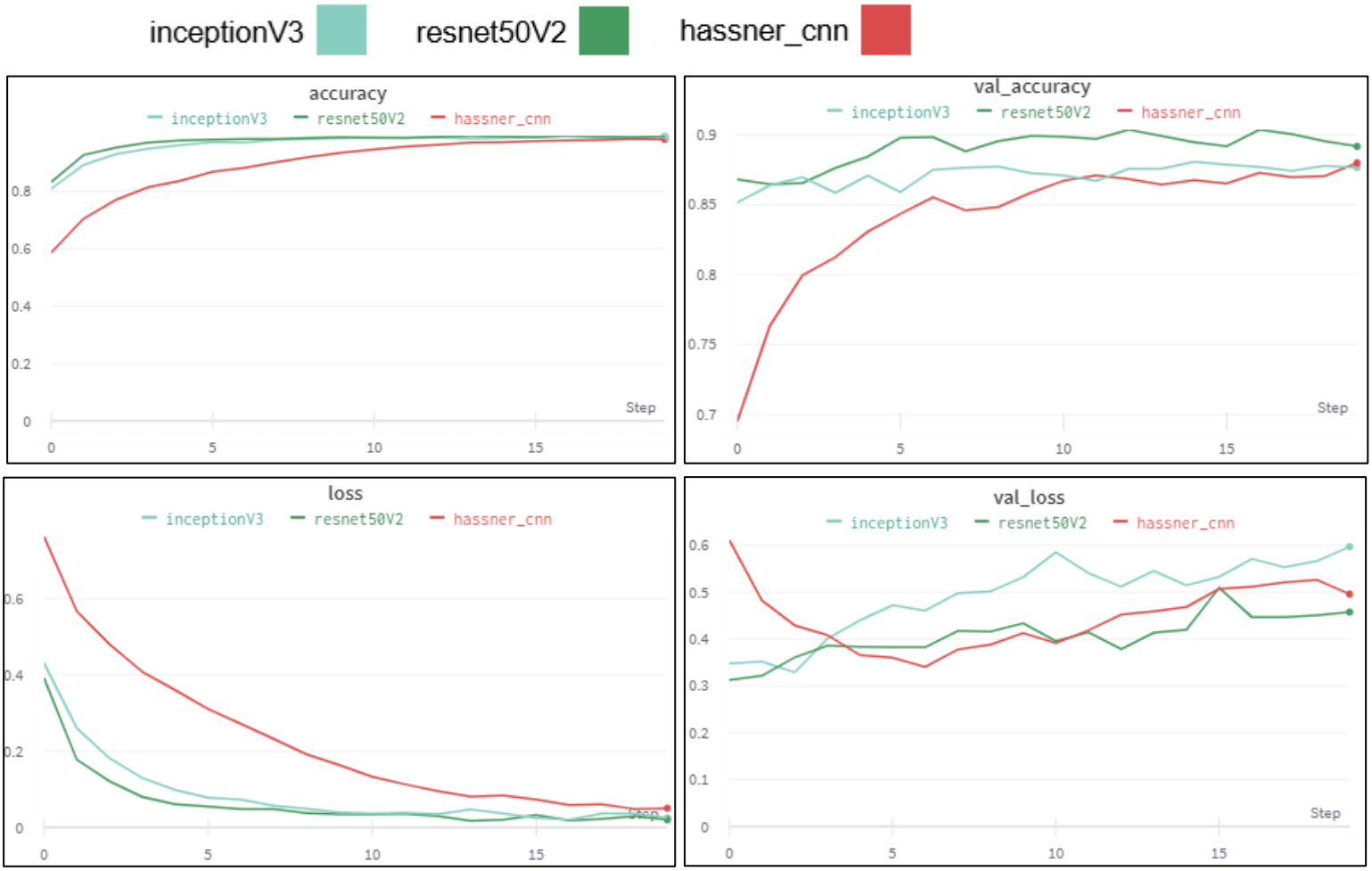
Configuration Used

```
configs = {
    'epochs': 20,
    'batch_size': 64,
    'seed': 7,
    'learning_rate': 1e-3, #0.001
    'hidden_activation': 'relu',
    'output_activation': 'sigmoid',
    'optimizer': 'adam',
    'loss_function': 'binary_crossentropy',
    'metrics': ['accuracy'],
    'fc_layer_1_neurons' : 1024,
    'fc_layer_2_neurons' : 512,
    'fc_layer_3_neurons' : 256,
}
```

After training the model, we evaluated its accuracy with model.evaluate() and obtain a validation accuracy of 0.891758918762207 **≈ 0.89 / 89%**

2.4. Comparison of Model Architectures

2.4.1. Visual Comparison



The Resnet50V2 model seemed to have the highest accuracy/val_accuracy and lowest loss/val_loss values out of the 3 models tested.

2.4.2. Statistical Comparison

Categories	Hassner Model	InceptionV3 Model	Resnet50V2 Model
Accuracy	0.9828	0.9905	0.9932
Val_Accuracy	0.8798	0.8765	0.8917
Loss	0.0486	0.0248	0.0198
Val_Loss	0.3408	0.3289	0.3123

We can see from the table above that in all categories, Resnet50V2 is the clear choice of model to use with the highest accuracy/val_accuracy values and lowest loss/val_loss values, this agrees with our previous observation as well. The Resnet50V2 model will then be used in our later experiments.

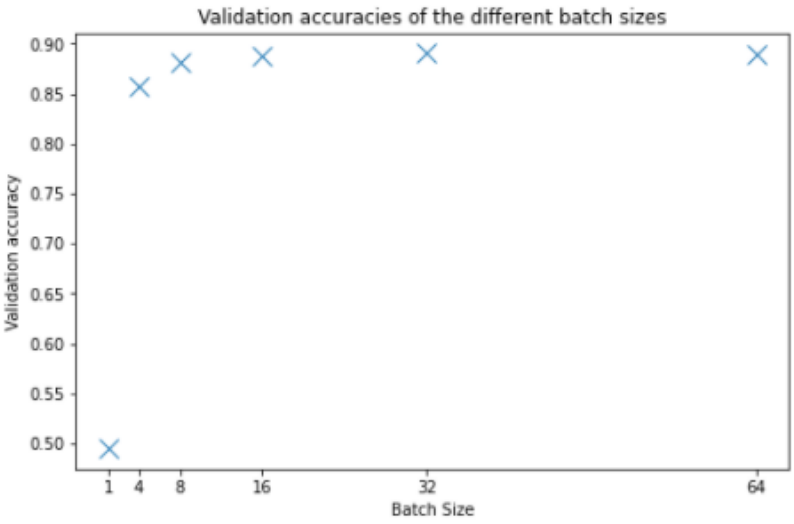
3. Description of methods used

3.1. Hyperparameter Tuning

To obtain the best parameters to use with the selected model, we tested the differing values of hyperparameters and selected the best performing ones. The configuration settings of each parameter were the same with the best hyperparameter found at each tuning being passed on to the next parameter tuning (e.g., best batch size found to be used in learning rate tuning etc.).

3.1.1. Batch Size Tuning

For batch size tuning, the 6 batch size values of **[1, 4, 8, 16, 32, 64]** were tested and evaluated. The results are as follows:

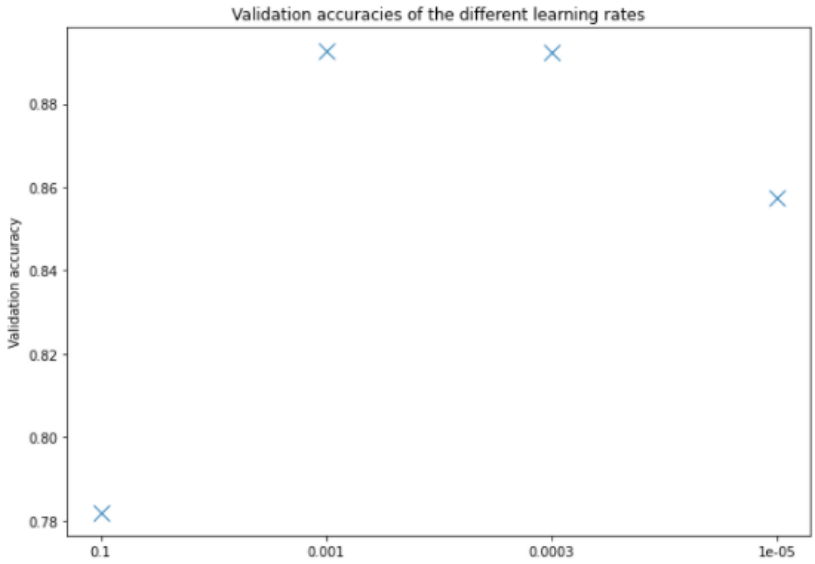


Batch Size	Mean Val_Accuracy
1	0.5179
4	0.8580
8	0.8805
16	0.8874
32	0.8903
64	0.8892

We observed from testing that batch sizes of 16, 32 and 64 have roughly equal validation accuracy however by the total average of val_accuracy obtained, batch size of **32** was determined as the best batch size to use.

3.1.2. Learning Rate Tuning

Utilizing the best batch size of 32 for learning rate tuning, the 4 learning rate values of **[0.1, 0.001, 0.0003, 0.00001]** were tested and evaluated. The results are as follows:



Learning Rate	Mean Val_Accuracy
1e-1 / 0.1	0.7820
1e-3 / 0.001	0.8928
3e-4 / 0.0003	0.8924
1e-5 / 0.00001	0.8573

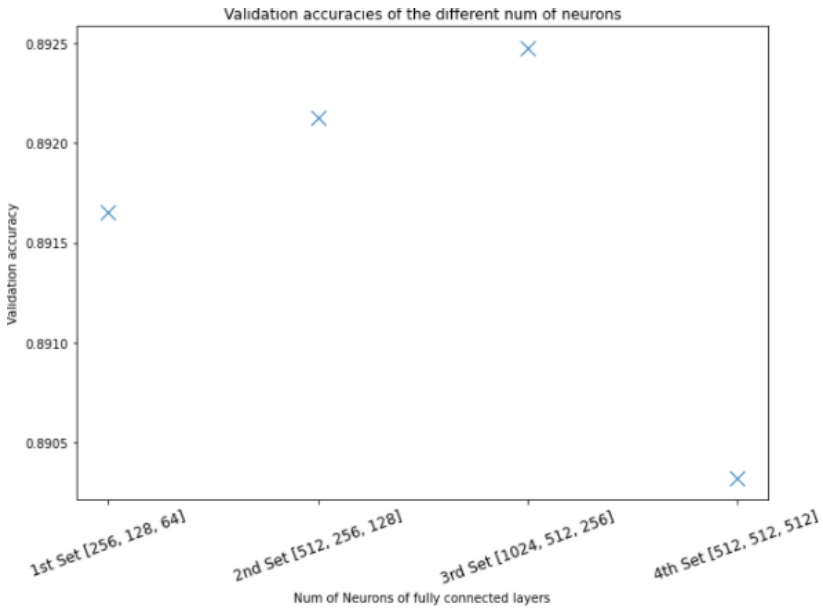
We observed from testing that the learning rate of 0.001 / 1e-3 produces the best validation accuracy by a decent margin, we thus determined **0.001** as the best learning rate to use.

3.1.3. Number of Neurons Tuning

Utilizing the previously tuned batch sizes and learning rates for the tuning of the number of neurons at each fully connected layer, the 4 sets of values described by table 1 were tested and evaluated.

	FC Layer 1	FC Layer 2	FC Layer 3
1st Set	256	128	64
2nd Set	512	256	128
3rd Set	1024	512	256
4th Set	512	512	512

Table 1: Sets of number of neurons at each FC layer

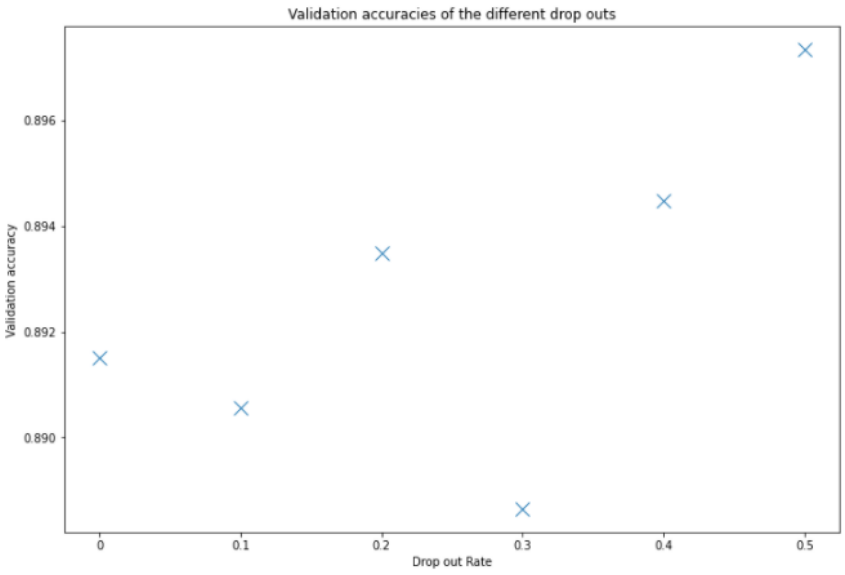


Number of Neurons (FC Layer 1 / FC Layer 2 / FC Layer 3)	Mean Val_Accuracy
1st Set [256 / 128 / 64]	0.8916
2nd Set [512 / 256 / 128]	0.8921
3rd Set [1024 / 512 / 256]	0.8924
4th Set [512 / 512 / 512]	0.8903

We observed from testing that the 3rd set of number of neurons produces the best validation accuracy by a decent margin, we thus determined **1024, 512 and 256** as the best number of neurons to be used at each fully connected layer respectively.

3.1.4. Drop Out Rate Tuning

For dropout rate tuning, the 6 dropout rate values of **[0, 0.1, 0.2, 0.3, 0.4, 0.5]** were tested and evaluated for each fully connected layer. The results are as follows:



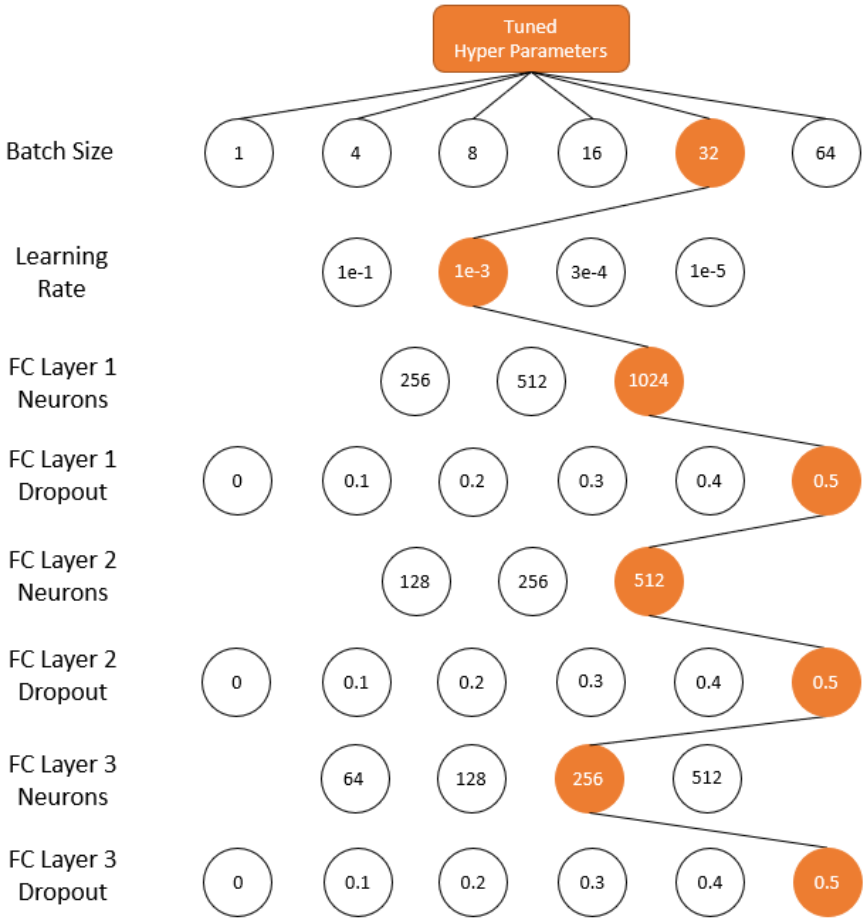
Dropout Rate at each Fully Connected Layer	Mean Val_Accuracy
0	0.8915
0.1	0.8905
0.2	0.8934
0.3	0.8886
0.4	0.8944
0.5	0.8973

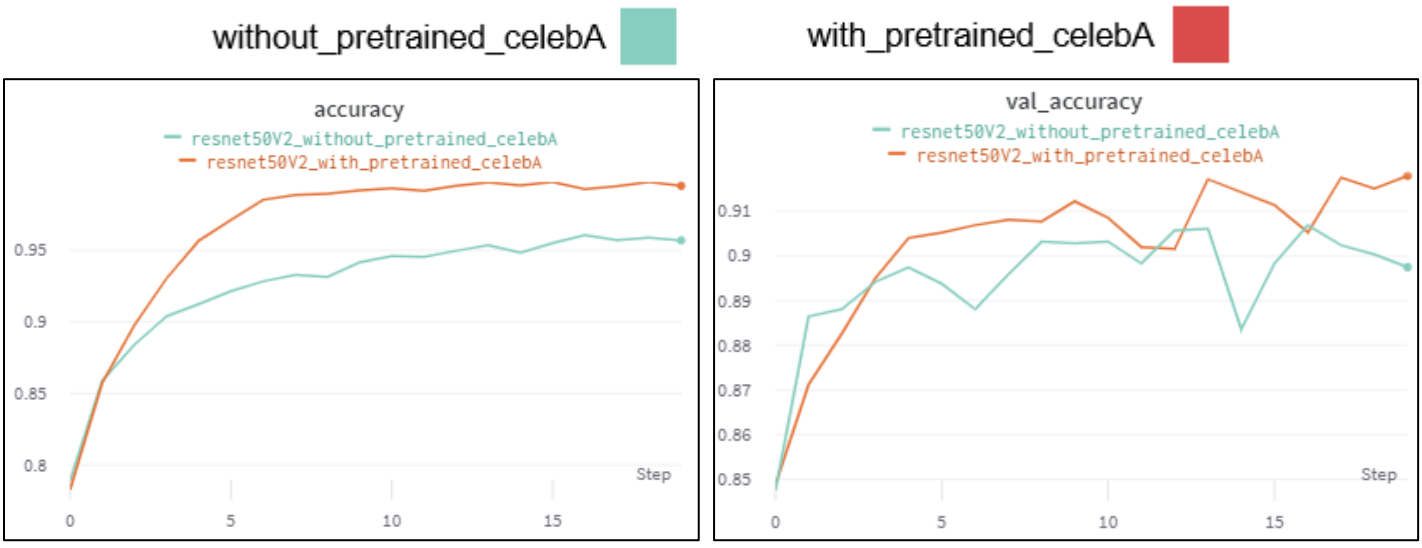
We observed from testing that the dropout rate of 0.5 for each fully connected layer produces the best validation accuracy by a decent margin, we determined **0.5** to be the best learning rate to use at each fc layer.

4. Experiments and Results

4.1. Experiment with and without utilizing pretraining

With our tuned parameters and selected model, we performed the gender classification task once more. The comparison will be between 2 models with the same tuned hyperparameters, however, one will utilize the pretrained weights learned from the CelebA dataset and one without. The experiments will run for the same **20 epochs** as per the previous models. The tuned parameters and comparison results are shown below:





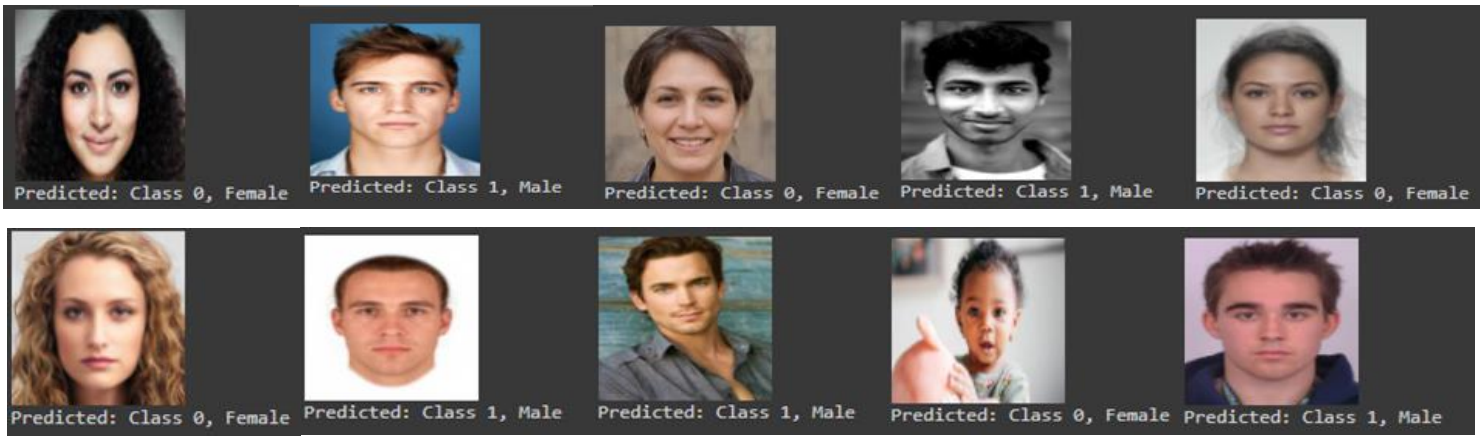
During the comparisons, the pre-training with CelebA produces higher accuracy/val_accuracy compared to that of the initial untuned model and the tuned model without pretraining. However, we observe that the training accuracy of the tuned model is significantly lower than that of the initial model, this could be explained by the use of dropouts for the tuned models which indicates a higher variance at some of the fully connected layers as compared to the initial evaluation model without dropout.

Categories	Initial resnet50V2 model (no tuning)	resnet50V2-without_pretrained_celebA	resnet50V2_ with_pretrained_celebA
Accuracy	0.9932	0.9607	0.9977
Val_Accuracy	0.8917	0.9061	0.9180

Overall, with tuning and pretraining, we are able to obtain an approximate **2%** increase in performance

4.2. Demo Experiment

In this demo experiment, we put our best model to the test against **10 random faces from the internet** and see whether it can predict their gender correctly. The result from this experiment is as follows:



5. Future Considerations

To increase the accuracy further and to solve overfitting issues, some considerations for future work could be to utilize and tune other parameters such as:

1) **L2 Regularization** - Penalty term discouraging weights from attaining large values
2) **Random Sampling (K-Fold Cross Validation)** – Testing Technique which enables targeted tuning of parameters based on certain evaluations on unseen data
3) **Learning Rate Scheduler** – Adjust weights during training to improve adaptability of model

6. Conclusion

In conclusion, we performed automatic gender classification of face images utilizing convolutional neural networks (CNN) on a variety of models on the Adience dataset. We experimented with the different models (Hassner, Inception, Resnet) and different hyperparameters to eventually obtain the best model and tuned parameters for the purpose of achieving the highest accuracy of gender classification. We also experimented using pre-trained weights from the CelebA dataset and achieved the highest val_accuracy observed from all experiments conducted of 0.918 **≈ 0.92 / 92%**.