Comparison of Imaged-Based Gender Classification Techniques

Project Report

Oscar Chen, Savith Jayasekera, Prince Okoli

April 11, 2019

Abstract

This paper implemented and compared multiple machine learning models focused on predicting the gender from a dataset of face images. The models were based on six different supervised classification algorithms as well as a deep learning model.

The algorithms included K Nearest Neighbor (KNN), support vector machines (SVM), random forest, logistic regression, linear discriminant analysis, and convolutional neural network (CNN) (the deep-learning model).

The SVM resulted in the highest accuracy among the traditional machine learning techniques with an overall accuracy of 73.3%. The best performing CNN model resulted in an overall accuracy of 90.0%. Both the SVM model and CNN models resulted in similar accuracies for their prediction of female versus male faces. The SVM model correctly predicted male and female faces 74.4% and 72.4% of the time respectively, and the CNN model correctly predicted male and female faces 90.1% and 88.8% of the time.

Introduction

With the increase in available user information through social media, automated gender prediction has become relevant to a variety of domains; these include human computer interaction, targeted advertising, and content-based searching [1]. Gender classification from facial images is therefore a thoroughly researched area in machine learning. Existing research have implemented gender classifiers based on traditional machine learning algorithms and ANNs. In this paper, we implemented multiple machine learning models with the goal of exploring the differences in model performance. Specifically, the following topics were explored:

- The accuracy of each model when trained on a data set of similar size. We determined the performance and “how quickly” the model learns.
- The performance of the models when predicting male faces compared to predicting female faces. We explored any possible differences between the predictions

Related Work

Gender classification using a hybrid structure which includes convolution neural network and extreme learning machine was implemented by Duan et al. [2]. The model was trained on the MORPH-II image database and resulted in an accuracy of 88.2% which was higher than all the previous models trained on the same database.

Nazir et al. [3] proposed a method of using discrete cosine transform for feature extraction and training a KNN model with the extracted features. Using the Stanford university medical student frontal facial images database, the proposed technique was able to achieve an accuracy of 99.3% for gender classification.

Moghaddam and Yang [4] investigated the performance of a gender classifier based on SVM algorithm. It was trained on a low-resolution image set from the FERET face database. The paper concluded that the SVM model yielded 3.4% error rate and was shown to be superior to traditional pattern classifiers such as linear, quadratic and nearest-neighbor.

Basha and Jahangeer [5] explored a method for gender classification of face images using a random forest model. The model was trained on the ORL database containing 400 face images. They reported a classification accuracy of 100% outperforming other techniques such as SVM, linear discriminate analysis, KNN, and fuzzy c-means.

Gender classification has been an area of interest since the beginning of the field of artificial intelligence dating back to the 1950s; Rosenblatt, a pioneer of the formative years of neural networks, used gender classification based on images of men and women to showcase his revolutionary perceptron [6]. While the perceptron failed due to its simplicity of being one-layer neural network, the core theory of behind the device was the foundation of ANNs. With the increase in available data and computing power, a resurgence of the use of image processing techniques for gender classification has occurred since the mid-2000s. The authors of the project acknowledge that the proposed techniques have already been explored in detail; our goal is to gain an appreciation of the practical aspects of machine learning techniques by delving deeper into the implementation of different models.

Methodology

This project is focused on understanding and practicing the implementation of different machine learning algorithms. As such, it aspired to address the following research questions:

Question 1: How accurate are the machine learning models at predicting gender?

This question explored the accuracy of the models used during the project. The models were trained and tested on the same training and testing sets from the data set. Each model was evaluated using the following metrics:

$$Overall\ Accuracy\ = \frac{TP}{TP + FN}$$

Where:

TP is number of males correctly classified.

TN is number of females correctly classified.

FP is number of females incorrectly classified as male.

FN is number of males incorrectly classified as female.

Note that **male is positive,** and **female is negative**.

Question 2: How do the models perform when detecting male faces compared to female faces?

This question was aimed to determine if the models perform better at detecting either gender. Two important metrics of binary classification was used to answer this research question. Namely, the metrics were specificity and precision which were calculated as shown below:

$$Recall\ for\ positive\ (male) = Sensitivity = \frac{TP}{TP + FN}$$

$$Recall\ for\ negative\ (female) = Specificity = \frac{TN}{TN + FP}$$

$$Precision\ for\ negative\ (female) = \frac{TP}{TP + FP}$$

$$Precision\ for\ positive\ (male) = \frac{TN}{TN + FN}$$

Data Preparation with PySpark

We were given 452,261 images from the IMDB dataset. These images have been cropped to create images of people's head with 40% margin. A large portion of the images had multiple people present, and therefore resulting in wrongly labelled cropped images as each image created multiple cropped headshots.

The data labels were given in a Matlab file. We parsed the content using Scipy library and stored in a Pandas dataframe.

Using PySpark we excluded any cropped images with multiple people present from the data, as their labels may be wrong. We then noticed there were several wrongly cropped

images with no faces obviously present, so using PySpark we removed these from our data set as well. Furthermore, using PySpark we removed images that were stretched out to fill up the frame of the image (see figures). After these clean up operations with PySpark, we had 181,626 images remaining in the dataset.

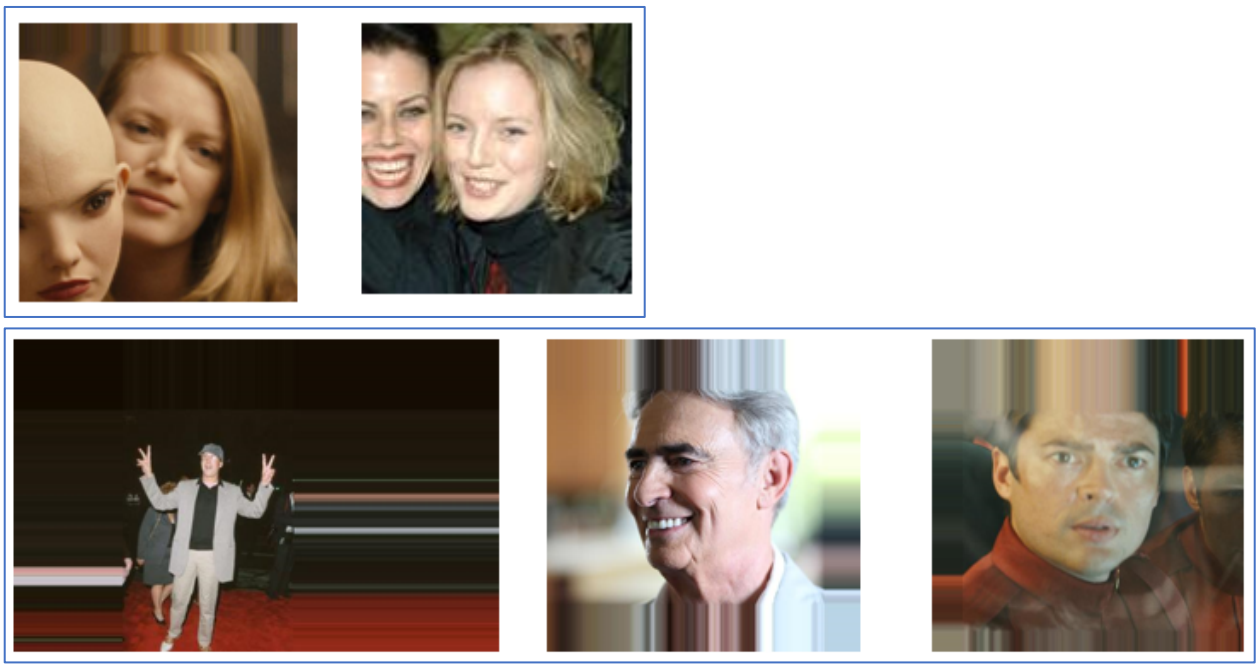


Figure 1 Samples of Imaged Removed

However, at this point, there was more male than female data points. Using random sampling, we further discarded a number of male data points to eliminate sample bias.

Our final data set has a total of 163,336 images, the data set is perfectly balanced with male and female genders each having 81,668 samples.

To be able to experiment our models in stages in a time-efficient manner, we created smaller datasets. We created dataframes containing 1 percent, 5 percent, 10 percent, and 20 percent of the final dataset. For each of the dataset, we further divided it into training set and test set at 95%-5% split.

The dataframes contain the following columns:

- 'path': relative file path to the jpg image file
- 'id': an unique id associated with the person
- 'name': name of the person
- 'dob': date of birth
- 'gender': the person's biological gender
- 'score1': face score of the person
- 'score2': face score of a second person if present
- 'pic_date': date the picture was taken

- ‘region’: the region on the original image where the cropped image comes from
- ‘age’: calculated age of the person at the time the photo was taken

We are only interested in the ‘path’ and ‘gender’, but we kept the other information in the dataframes for future studies involving the same dataset.

Data Preprocessing for CNN

We used a data generator from Keras library to feed images into our training model. For most of the training experiments, the images were resized to 100 pixel by 100 pixels, gray scale, and normalized. The batch size we used for training was 64. The training data set was further divided into training set and validation set with a 90%-10% split. The chart below highlights how data is used in the training and testing process.

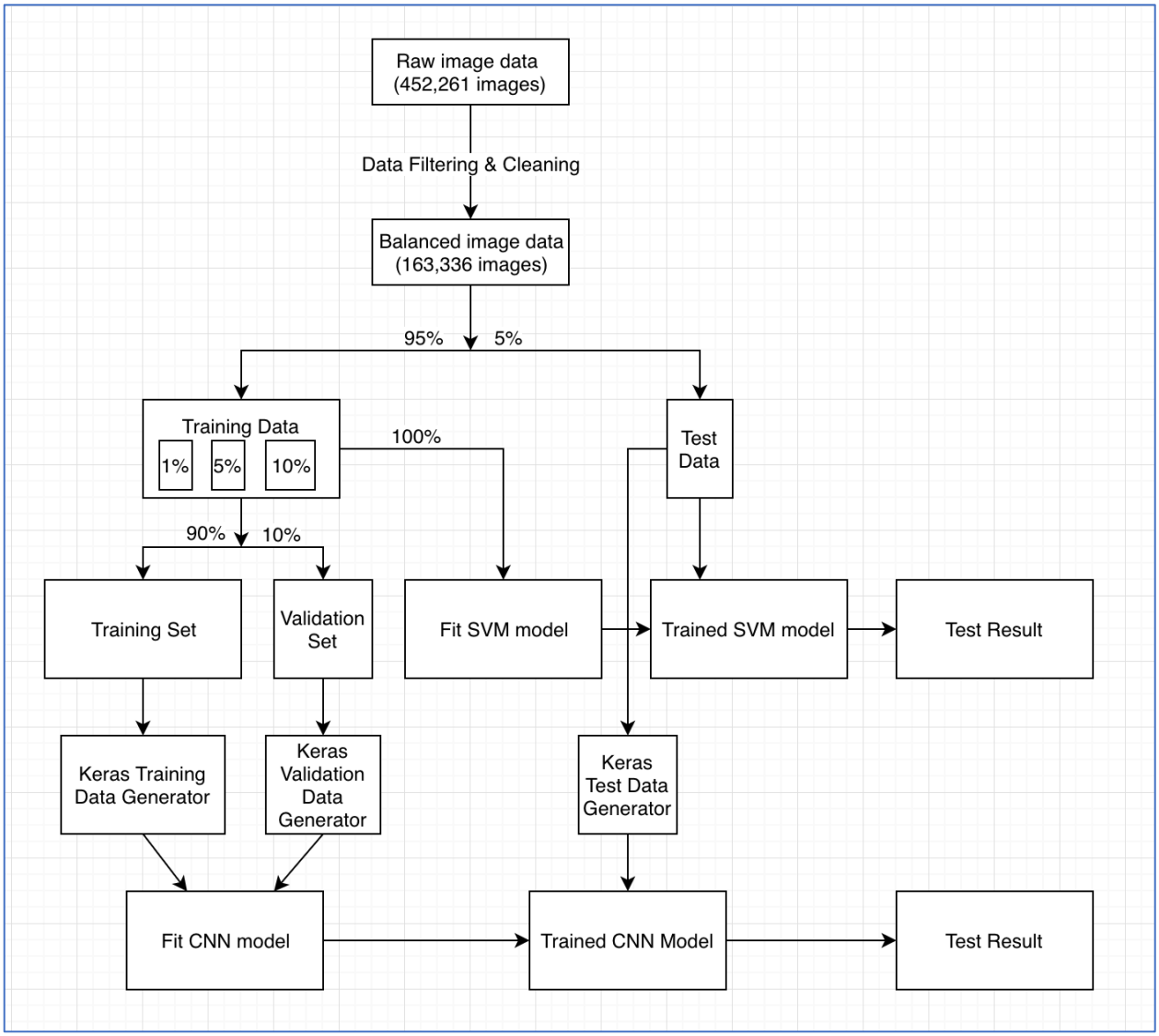


Figure 2 Overall Data Process

Image Preprocessing for SVM and other Traditional Machine Learning Techniques

The images were converted to greyscale and then to 2-dimensional tensors (matrices), where each pixel is a feature of the resulting dataset. The images were also resized so that all the images were the same size. The resulting dataset was also normalized with respect to intensity by dividing each intensity by 255.

Efficiency of the cleaning process

The cleaning process was executed several times with various PySpark nodes and tasks, and the summary of the results are presented below. In general, there was no considerable difference because the processing applied through the metadata of the dataset.

Table 1 Comparison of Speed of Nodes in Spark

Entire dataset (100%)		
Nodes	Tasks	Duration (seconds)
1	5	0.67
5	5	0.59
5	1	0.65
3	3	0.62
1	1	0.61
10% of dataset		
Nodes	Tasks	Duration (seconds)
1	5	0.63
5	5	0.59

Convolutional Neural Networks

We used Keras library to construct the neural network models, all layers are standard layers provided by the Keras library, as well as implementation of the activation functions, loss function, optimizer, and kernels.

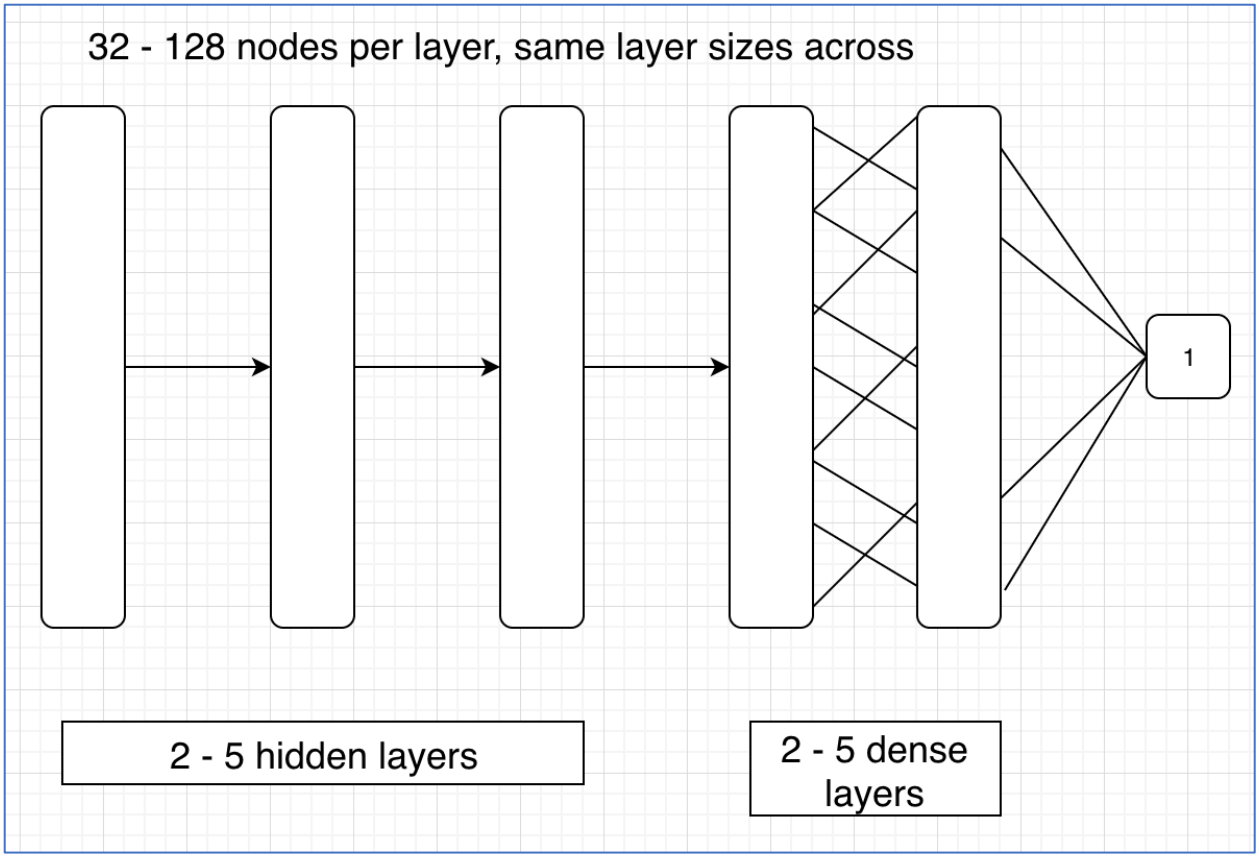


Figure 3 Simple Conv Net Design for Hyperparameter Tuning Trials

We started with a relatively simple convolutional neural network architecture where layers of identical sizes are used. The networks consist of 2 to 5 hidden layers, followed by 2 to 5 dense layers, followed by an output layer. Except for the output layer, each of the layers either have 32, 64, 96, or 128 nodes per layer, and have the same number of nodes across all layers.

The hidden layers was padded to ensure size does not change, the hidden layers had 3X3 kernel, and using relu activation function. A 2X2 max pooling kernel was used after each hidden layer.

The dense layers also uses relu activation function. On the final output layer, sigmoid activation function was used as the age classification was treated as a binary classification problem.

We used Adam optimizer with a constant default learning rate of 0.001, and validation accuracy as training metrics. Binary cross-entropy was used as loss function.

A combination of hyperparameters were tested with this simple architecture, the validation accuracy and loss for each trial are shown in the graphs below.

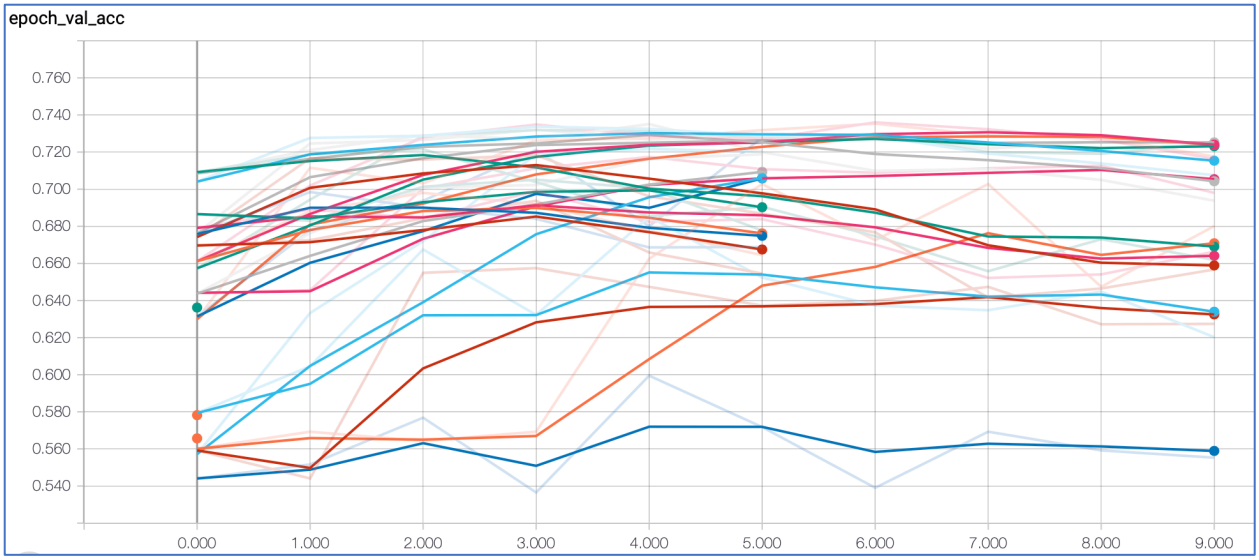


Figure 4 Validation Accuracy for Simple CNN Trials

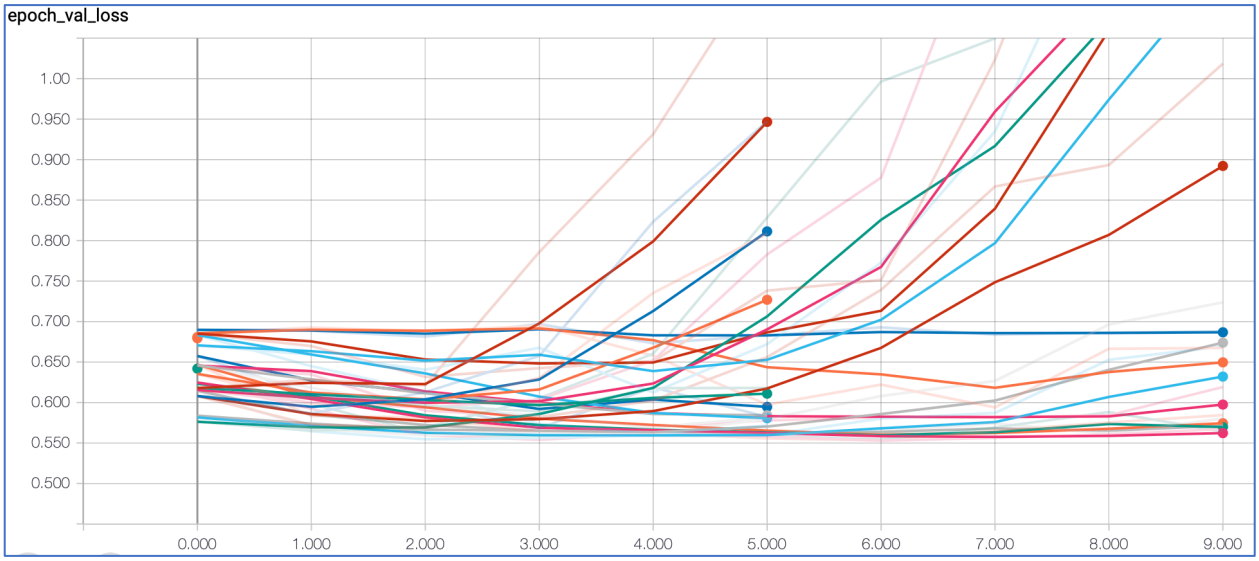
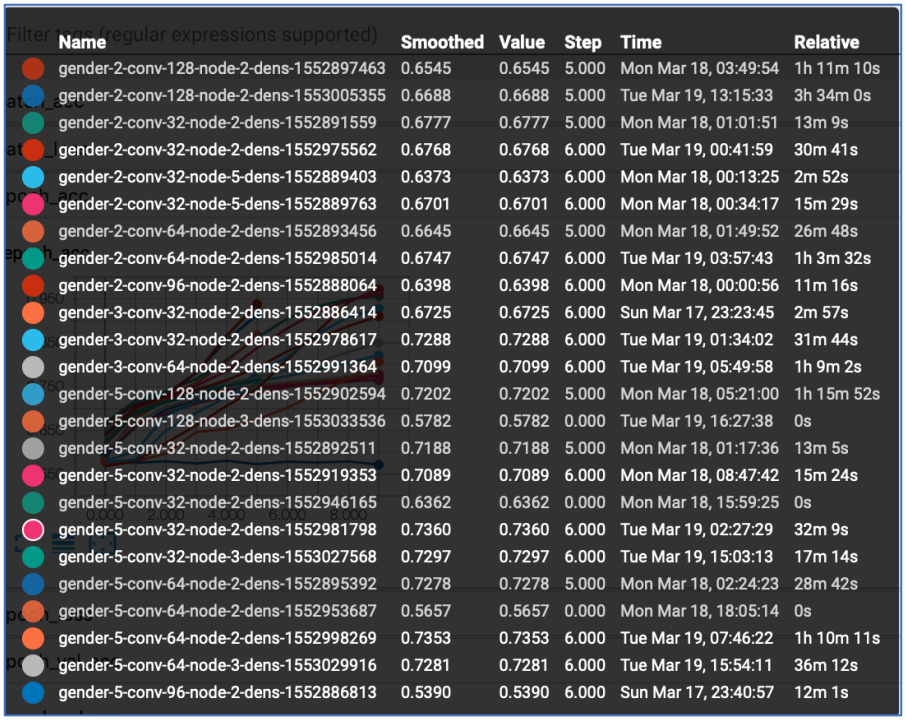


Figure 5 Validation Loss for Simple CNN Trials

As shown, we first ran 5 epochs, and selected the models with relatively high validation accuracies and low losses, from there we trained these select models again to up to 10 epochs to see if they continue to improve.

Disappointingly, all of these models started to show performance degradation after 6 epochs. The table below shows all of the trials we have ran with this architecture and the validation accuracy at 6th epoch. The best validation accuracy achieved was 73.6% using 5 hidden layers, 2 dense layers, and each layer has 32 nodes.



Name	Smoothed	Value	Step	Time	Relative
gender-2-conv-128-node-2-dens-1552897463	0.6545	0.6545	5.000	Mon Mar 18, 03:49:54	1h 11m 10s
gender-2-conv-128-node-2-dens-1553005355	0.6688	0.6688	5.000	Tue Mar 19, 13:15:33	3h 34m 0s
gender-2-conv-32-node-2-dens-1552891559	0.6777	0.6777	5.000	Mon Mar 18, 01:01:51	13m 9s
gender-2-conv-32-node-2-dens-1552975562	0.6768	0.6768	6.000	Tue Mar 19, 00:41:59	30m 41s
gender-2-conv-32-node-5-dens-1552889403	0.6373	0.6373	6.000	Mon Mar 18, 00:13:25	2m 52s
gender-2-conv-32-node-5-dens-1552889763	0.6701	0.6701	6.000	Mon Mar 18, 00:34:17	15m 29s
gender-2-conv-64-node-2-dens-1552893456	0.6645	0.6645	5.000	Mon Mar 18, 01:49:52	26m 48s
gender-2-conv-64-node-2-dens-1552985014	0.6747	0.6747	6.000	Tue Mar 19, 03:57:43	1h 3m 32s
gender-2-conv-96-node-2-dens-1552888064	0.6398	0.6398	6.000	Mon Mar 18, 00:00:56	11m 16s
gender-3-conv-32-node-2-dens-1552886414	0.6725	0.6725	6.000	Sun Mar 17, 23:23:45	2m 57s
gender-3-conv-32-node-2-dens-1552978617	0.7288	0.7288	6.000	Tue Mar 19, 01:34:02	31m 44s
gender-3-conv-64-node-2-dens-1552991364	0.7099	0.7099	6.000	Tue Mar 19, 05:49:58	1h 9m 2s
gender-5-conv-128-node-2-dens-1552902594	0.7202	0.7202	5.000	Mon Mar 18, 05:21:00	1h 15m 52s
gender-5-conv-128-node-3-dens-1553033536	0.5782	0.5782	0.000	Tue Mar 19, 16:27:38	0s
gender-5-conv-32-node-2-dens-1552892511	0.7188	0.7188	5.000	Mon Mar 18, 01:17:36	13m 5s
gender-5-conv-32-node-2-dens-1552919353	0.7089	0.7089	6.000	Mon Mar 18, 08:47:42	15m 24s
gender-5-conv-32-node-2-dens-1552946165	0.6362	0.6362	0.000	Mon Mar 18, 15:59:25	0s
gender-5-conv-32-node-2-dens-1552981798	0.7360	0.7360	6.000	Tue Mar 19, 02:27:29	32m 9s
gender-5-conv-32-node-3-dens-1553027568	0.7297	0.7297	6.000	Tue Mar 19, 15:03:13	17m 14s
gender-5-conv-64-node-2-dens-1552895392	0.7278	0.7278	5.000	Mon Mar 18, 02:24:23	28m 42s
gender-5-conv-64-node-2-dens-1552953687	0.5657	0.5657	0.000	Mon Mar 18, 18:05:14	0s
gender-5-conv-64-node-2-dens-1552998269	0.7353	0.7353	6.000	Tue Mar 19, 07:46:22	1h 10m 11s
gender-5-conv-64-node-3-dens-1553029916	0.7281	0.7281	6.000	Tue Mar 19, 15:54:11	36m 12s
gender-5-conv-96-node-2-dens-1552886813	0.5390	0.5390	6.000	Sun Mar 17, 23:40:57	12m 1s

Figure 6 Peak Validation Accuracy Results for Simple CNN Trials

Next, we modified our networks architecture in an attempt to capture more info from the training images, so that the networks do not saturate after only 6 epochs. We came up with an architecture demonstrated in the image below.

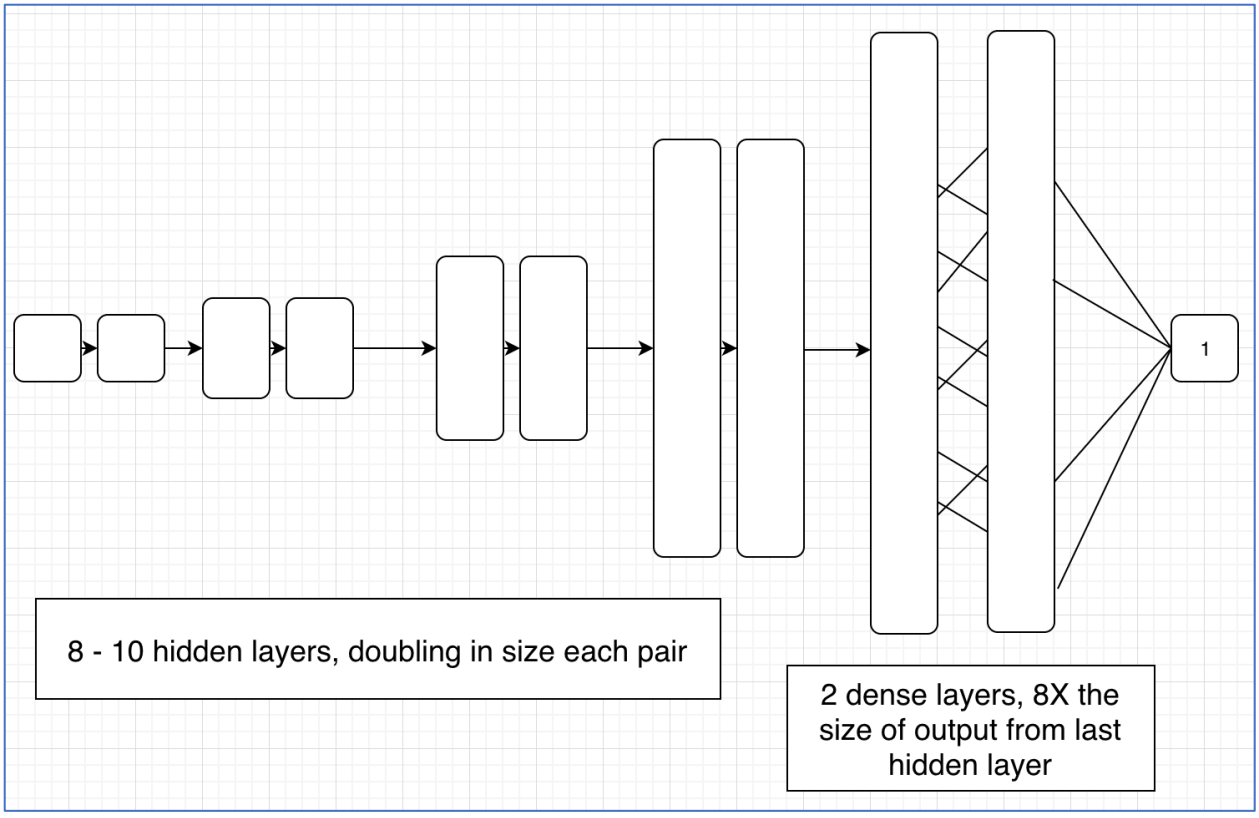


Figure 7 More Complex CNN Design for Hyperparameter Tuning

The original paper authors used a pre-trained VGG-16 model, we took some inspiration from the VGG-16 architecture and incorporated into our design. Our architecture has the following distinct properties:

- The VGG-16 starts with small hidden layer, and doubles the layer size every other layer. We took this feature and incorporated into our design, as you can see that we start with two hidden layers with 32 nodes, and then doubling the number of nodes every pair thereafter until we get to the dense layers.
- We do not have as many layers in our design as the VGG-16 architecture. The VGG-16 was developed as a classifier with 1000 classes (1000 node soft-max output layer), we do not believe our network needs quite as much complexity since our goal is binary classification. After some experimentation, we narrowed down to 8-10 hidden layers structure, with 2 dense layers, and a single node sigmoid output layer at the end.
- The dense layers would be considerably larger than the hidden layers. After some testing, we settled on having the dense layer's size being 4 times the size as the preceding hidden layer, or 8 times the size as the output from the preceding hidden layer.
- In the implementation of VGG-16, we did not see batch normalization and drop out layers, they may have been removed after training. We added batch

normalization between each of the layers, which seem to make this architecture much easier to train.

- We also added a drop out layer after each pair of hidden layer, with a setting of 0.25. A single drop out layer at the end of the dense layer was used, with a setting of 0.5. Several trials were run and the results are shown in the graphs below.

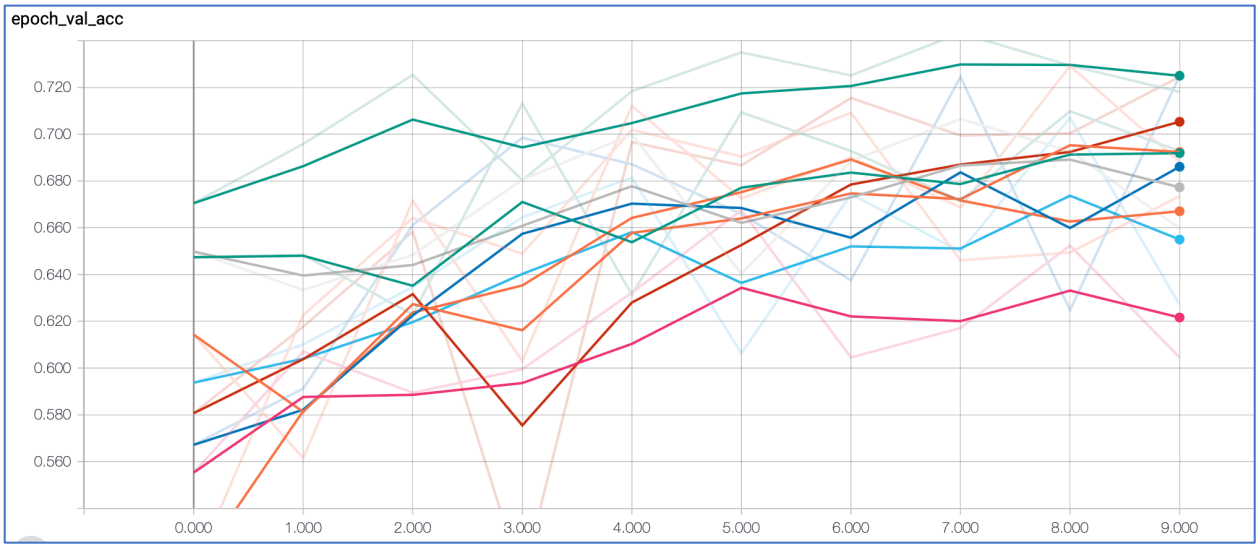


Figure 8 Validation Accuracy for CNN Trials

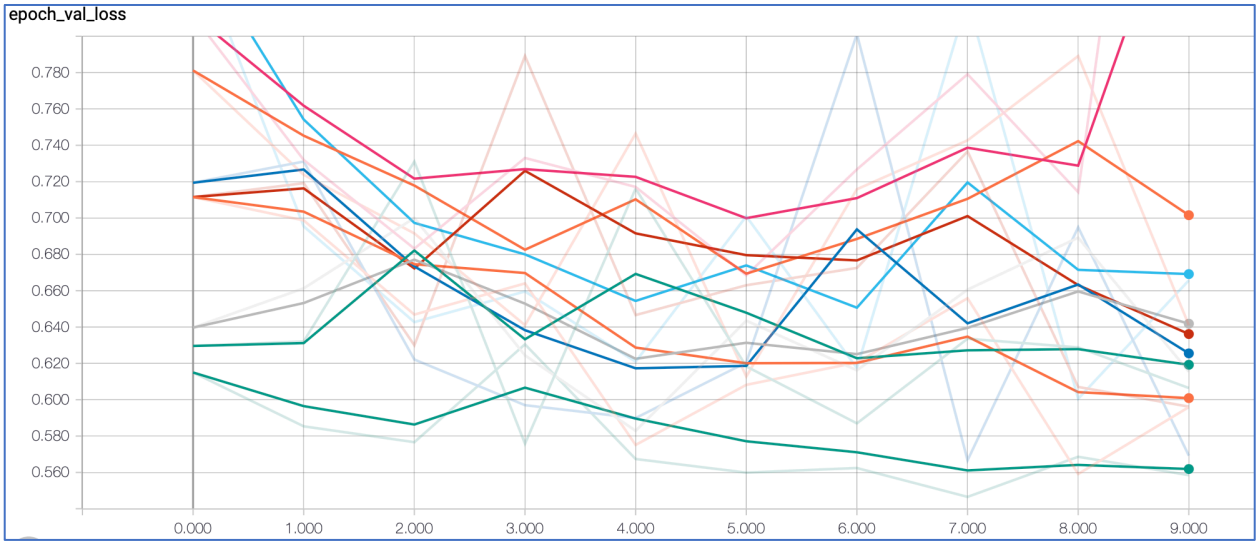


Figure 9 Validation Loss for CNN Trials

Name	Smoothed	Value	Step	Time	Relative
BN-3-conv-32-node-2-dens-1553635516	0.6171	0.6171	7.000	Tue Mar 26, 15:37:13	10m 10s
BN-3-conv-32-node-2-dens-1553636464	0.6716	0.6716	7.000	Tue Mar 26, 16:39:54	51m 11s
BN-3-conv-32-node-2-dens-1553641383	0.7065	0.7065	7.000	Tue Mar 26, 18:09:10	57m 27s
BN-4-conv-32-node-2-dens-1553646613	0.6687	0.6687	7.000	Tue Mar 26, 19:37:31	58m 19s
BN-4-conv-32-node-2-dens-1553651832	0.7245	0.7245	7.000	Tue Mar 26, 23:25:18	3h 1m 27s
BN-4-conv-32-node-2-dens-1553710479	0.7432	0.7432	7.000	Wed Mar 27, 14:30:34	1h 57m 54s
BN-5-conv-32-node-2-dens-1553695243	0.6994	0.6994	7.000	Wed Mar 27, 09:23:42	1h 11m 38s
BN-5-conv-32-node-2-dens-1553721908	0.6461	0.6461	7.000	Wed Mar 27, 18:09:09	2h 23m 7s
BN-5-conv-32-node-3-dens-1553703610	0.6498	0.6498	7.000	Wed Mar 27, 11:46:06	1h 14m 24s

Figure 10 Validation Accuracy Achieved During CNN Trials

The most significant trend was that most of these models have not shown performance peaking after 10 epochs, and we believe they will continue to improve with more training on larger data set. We incrementally carried out each of these trials, testing different hyperparameters along the way. The following highlights how we compared the hyperparameters and how we found our final model.

Larger dataset improves the overall performance of the networks

BN-3-conv-32-node-2-dens-1553635516 vs BN-3-conv-32-node-2-dens-1553636464: same hyperparameters were used in both models, but the second model is trained on 5 times the image set. The first model was trained on only 1 percent of the dataset, the latter trained on 5 percent of the dataset.

Figure 11 CNN: Comparison of Dataset Size

For the subsequent trials, we will continue to use the larger 5 percent dataset.

Dense layer having 8X of size of the output of the preceding hidden layer seems to work

BN-3-conv-32-node-2-dens-1553641383 vs BN-3-conv-32-node-2-dens-1553636464: The first model follows a "convention" of using a dense layer that is 8 times the size of the output from the preceding hidden layer, the latter is using a dense layer that is 16 times. The latter showed lower accuracy and higher loss, although not significantly.

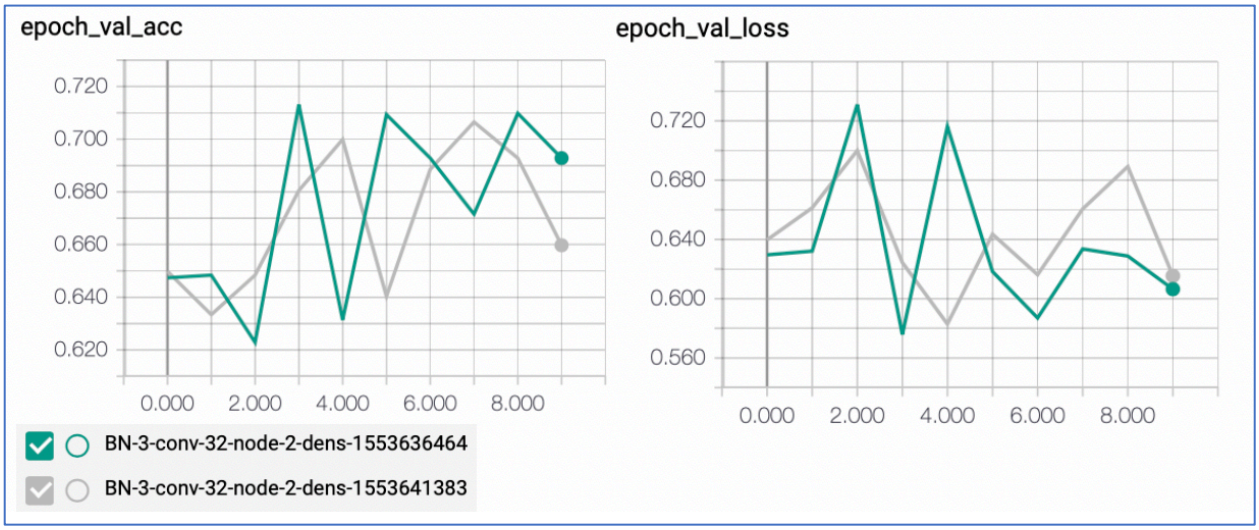


Figure 12 CNN: Comparison of 8X vs 16X Dense Layer Sizes

Increasing the number of hidden layers improves the model

BN-3-conv-32-node-2-dens-1553641383 vs BN-4-conv-32-node-2-dens-1553646613 vs BN-5-conv-32-node-2-dens-1553695243: They each use 3, 4, and 5 pairs of hidden layers. Because of we are sticking to the convention of having dense layers that are 8 times the sizes of the output of the preceding hidden layer, the latter models also have twice as large dense layers. The resulting validation accuracy and loss shown improvements.

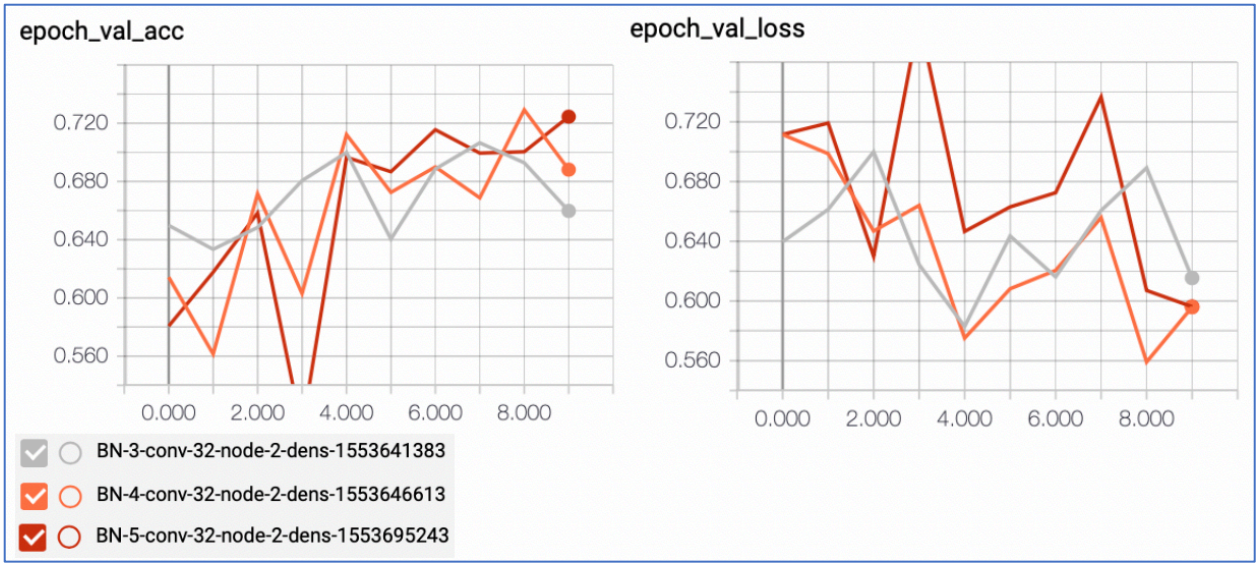


Figure 13 CNN: Comparison of Number of Hidden Layers

Increasing size of images for training improves the model but at expensive the training time

While the size of the image matter depending on the feature and goal of the models, having very large image does not necessarily improve performance of the models. BN-4-conv-32-node-2-dens-1553646613 vs BN-4-conv-32-node-2-dens-1553651832: the latter was trained on images with size of 224 pixels instead of 120 pixels, the overall accuracy and loss improved very slightly but took much longer to train.

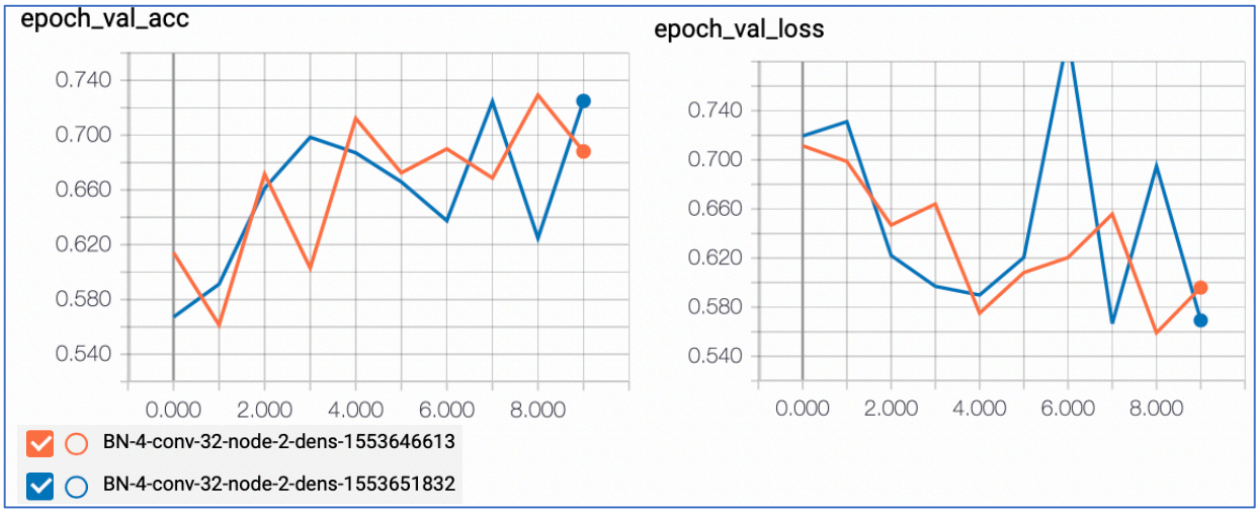


Figure 14 CNN: Comparison of 120 vs 224 Image Sizes for Training

Increasing the number of dense layers did not improve the model

BN-5-conv-32-node-2-dens-1553695243 vs BN-5-conv-32-node-3-dens-1553703610: Same settings except the latter model had an extra dense layer, the resulting validation accuracy and loss are slightly worse.

Figure 15 CNN: Comparison of Number of Dense Layers

The effects of training data set size and number of epochs on complexity of the networks explored

So far, the best trials were BN-5-conv-32-node-2-dens-1553695243 and BN-4-conv-32-node-2-dens-1553646613. The only difference between these two are the number of hidden layers.

Figure 16 CNN: Comparison of Training on Smaller Dataset

They both showed good performance overall and continued trend of improvement after 10 epochs. These two models have very similar design, except that the first one has 4 pairs of hidden layers, with dense layers having size of 1024 neurons; and the latter has 5 pairs of hidden layers, with dense layers having size of 2048 neurons.

We wanted to see if the two networks would start to differentiate with larger dataset. The next two trials were using larger 10 percent data set on these same two networks for comparison.

Figure 17 CNN: Comparison of Training on Larger Dataset

It seems to indicate that the larger and more complex model performed a bit worse with the increased training set size. However we are not convinced. The next trial is a comparison of the same architecture, but training for much longer up to 30 epochs.

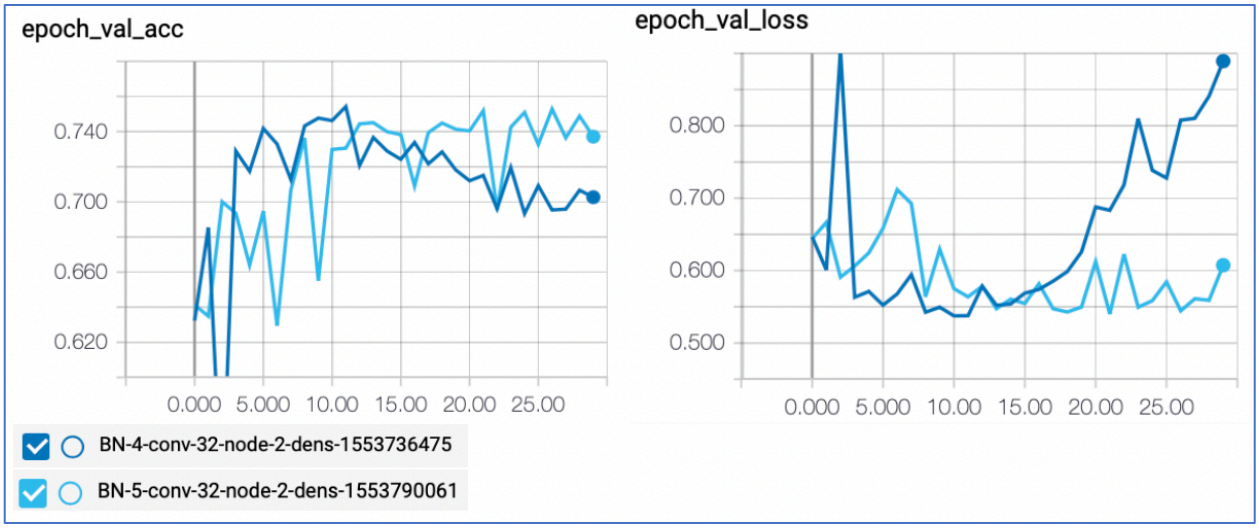
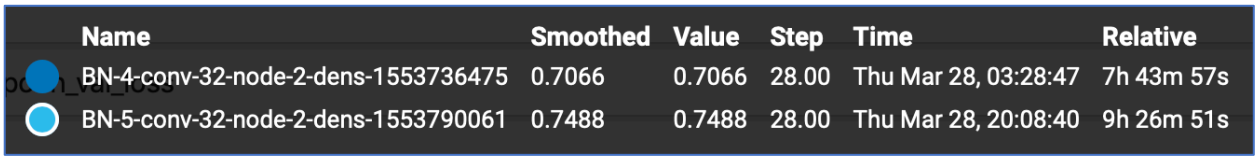


Figure 18 CNN: Comparison of Two Models Trained to Up to 30 Epochs

As the graphs have shown here, even though in the first 10 epochs, the smaller less complex network performed better, as more repeated training took place, the larger more complex network is able to continue to improve its parameters in prolong epochs while the simpler network started to deteriorate after 12 epochs. We believe the larger network having more parameters take longer to train.

The smaller network with 4 pairs of hidden layers and 1024 neurons in dense layers appear to have saturated at 11th epoch and shown a degradation in validation accuracy and loss thereafter. The larger network with 5 pairs of hidden layers and 2048 neurons in dense layers appear to have continued to improve gradually all through 30 epochs. Their validation accuracy is shown below.



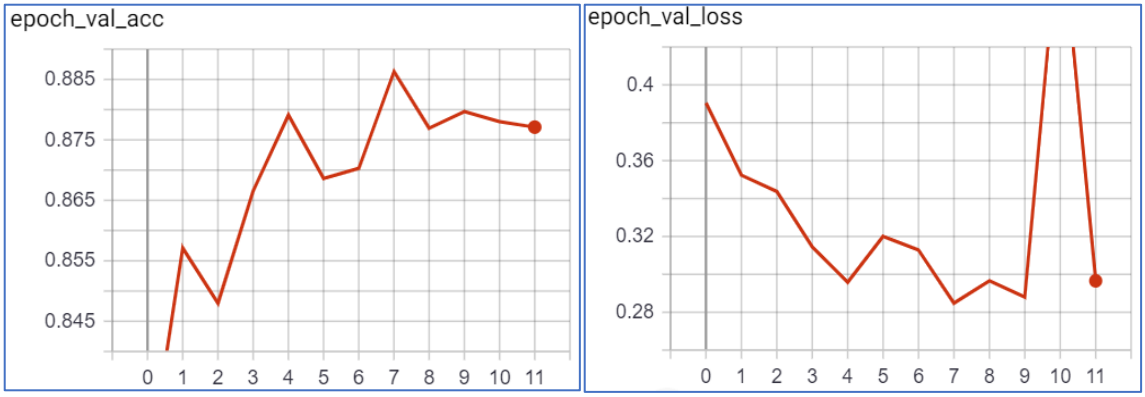
Name	Smoothed	Value	Step	Time	Relative
BN-4-conv-32-node-2-dens-1553736475	0.7066	0.7066	28.00	Thu Mar 28, 03:28:47	7h 43m 57s
BN-5-conv-32-node-2-dens-1553790061	0.7488	0.7488	28.00	Thu Mar 28, 20:08:40	9h 26m 51s

Figure 19 CNN: Peak of Validation Accuracy of the Two Models Compared

We believe the latter design is appropriate for training on larger dataset, which is the design we used for subsequent training to produce a final model. At this point, we implemented some transfer training features from the Keras library, and we are able to save weights from the best epoch and resume training with different hyperparameters. We eventually slowed down learning rate to 0.0001 (from 0.001), increased the hidden layer dropout to 0.5 (from 0.25), increased the dense layer dropout to 0.7 (from 0.5), and were able to sustain further training to gain another 1% improvement in validation accuracy. In the end, we decided on the following parameters to train our model:

- 10 hidden layers in total (or 5 pairs)
- Hidden layer start with 32 nodes in the first two layers, doubling in sizes every second hidden layer, up to 512 in the last two hidden layers.
- 2 densely connected layers in total, having 2048 nodes each.
- 1 output layer with 1 node.
- Batch normalization between each of the hidden layer.
- All of the hidden layers use 3X3 convolutional kernel, with padding that ensure the convolution does not change size.
- The first hidden layer uses 3X3 max pooling, with all subsequent hidden layers having 2X2 max pooling.
- Relu activation function is used in all of the hidden layers and dense layers. Sigmoid activation is used in the output layer.
- We used drop out of 0.35 after each pair of the hidden layer. We used a drop out of 0.6 at the end of 2nd dense layer.
- We used Adam optimizer with constant learning rate of 0.0008, binary cross-entropy loss function, and accuracy as metrics.

The validation accuracy and loss for the training is shown in the graphs below.



Name	Smoothed	Value	Step	Time	Relative
V2-5-conv-32-node-2-dens-1554852878	0.8863	0.8863	7	Wed Apr 10, 05:09:50	10h 6m 52s
V2-5-conv-32-node-2-dens-1554852878	0.2848	0.2848	7	Wed Apr 10, 05:09:50	10h 6m 52s

Figure 20 CNN: Final Model Trained From Scratch

We took the weights from the 7th epoch where the model had highest validation accuracy of 0.886, with the lowest validation loss of 0.285, and compiled the final model.

Traditional machine learning classifiers

The following classifiers were used for this project:

- Linear Discriminant Analysis (LDA) classifier
- Multinomial Logistic regression (MLR) classifier
- Gaussian Naïve Bayes (GNB) classifier
- K-Nearest Neighbour (KNN) classifier
- Random Forest (RF) classifier
- Support Vector Machine (SVM) classifier

Hyperparameter tuning

The hyperparameters for the classifiers, especially the SVM classifier, were iteratively tuned in order to achieve better performances.

For KNN, Euclidean distance was immediately used as the measurement for nearness, and the square root of the number of instances was chosen as the k.

For the SVM classifier, the radial basis function kernel were both tested out. Also, the budget and gamma hyperparameters were iteratively tuned for optimum performance.

Table 2 SVM Hyperparameter Tuning

Budget parameter	Gamma	Overall Accuracy
1	0.0001	68.9%
1	0.0005	70.9%
1	0.0009	71.9%
10	0.0001	72.5%
10	0.0005	73.3%
10	0.0009	72.9%
100	0.0001	72.8%
100	0.0005	69.8%
100	0.0009	71.3%

Evaluation

Traditional machine learning classifiers

The first important outcome of the learning, as can be seen in the confusion matrices, is that none of the classifiers were simply classifying all instances as just one class; an outcome that can still lead to spuriously high accuracy, depending on the distribution of the classes within the dataset.

For 10% of the total dataset, linear discriminant analysis classifier had the weakest performance, with both low recall and precision. The Gaussian naïve Bayes classifier, which has a solid track record of impressive performance in text classification, also performed poorly. The KNN classifier performed poorly based on its overall accuracy, however it classified female faces very accurately compared to the other classifiers but was weighed down by its poorer accuracy with male faces. Interestingly it was more precise with its classification of male faces than most of the other classifiers.

In fact, four of the six classifiers, which are LDA, GNB, KNN, and RF, all struggled with classifying both genders with similar accuracies. They all generally tended to classify female faces more accurately than male faces. MLR and SVM, on the other hand, both classified male faces with similar accuracies as female faces.

The SVM classifier produced the highest overall accuracy, 73%, out of all the six traditional ML classifiers. It also produced high precisions for both genders, 74% for female and 73% for male.

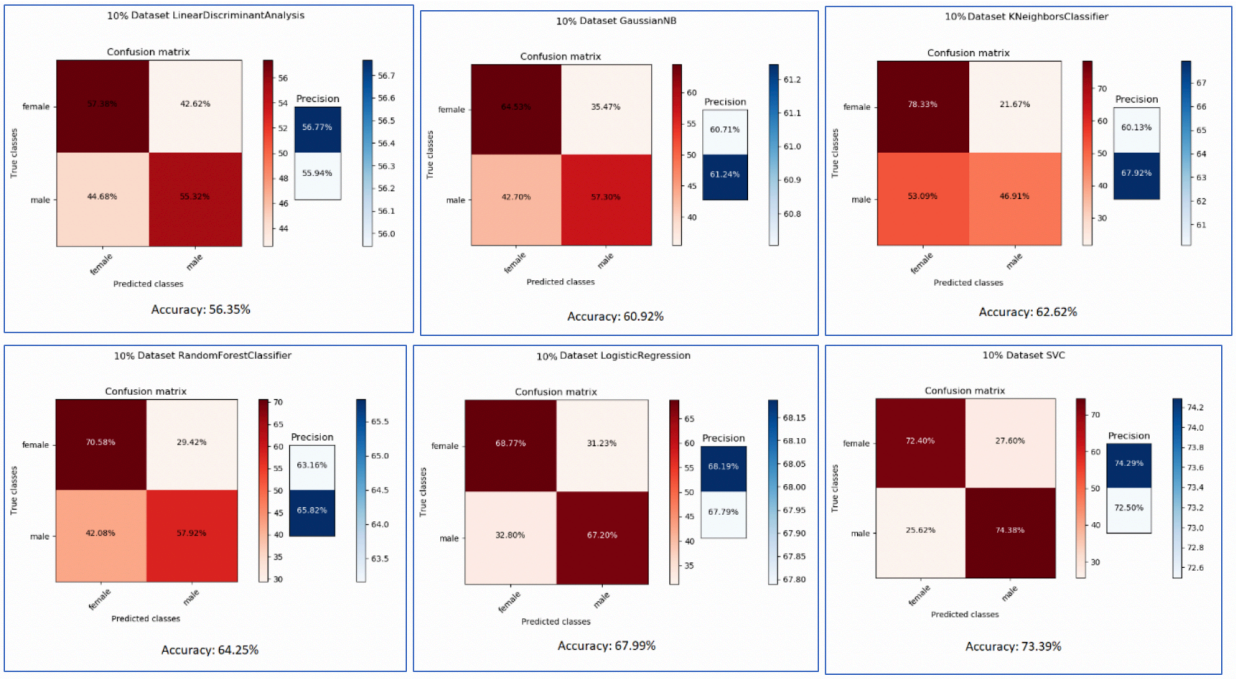
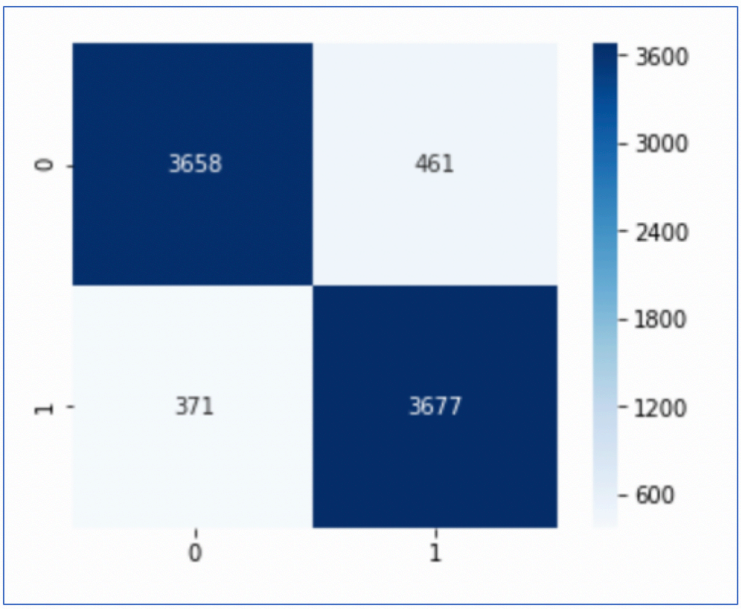


Figure 21: Comparison of Traditional Machine Learning Techniques

Final Convolutional Neural Network

At the evaluation stage, the test dataset, which is not part of the training set or validation set, was used for the purpose of evaluation.

A total of 8,167 images were used for testing. The confusion matrix is illustrated in the graph below.



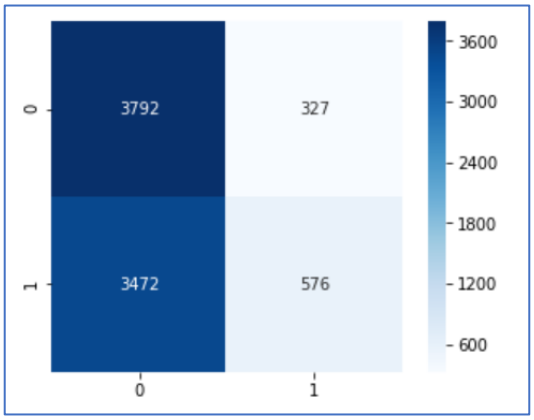
- Specificity: 0.89
- Recall: 0.91
- Precision (Male): 0.89
- Precision (Female): 0.91
- Accuracy: 0.90

Figure 22 CNN: Final Trained Model Evaluation Metrics

Final CNN Model Trained with Smaller Training Sets

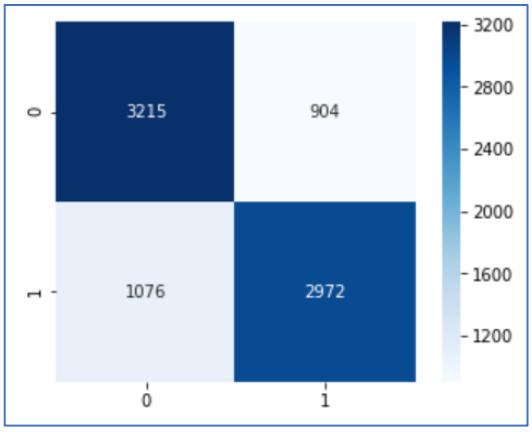
To demonstrate the effect of training sample set on the CNN model, we trained multiple instances of the model from scratch using different number of images.

CNN model trained with ~1500 images (1% of filtered data set):



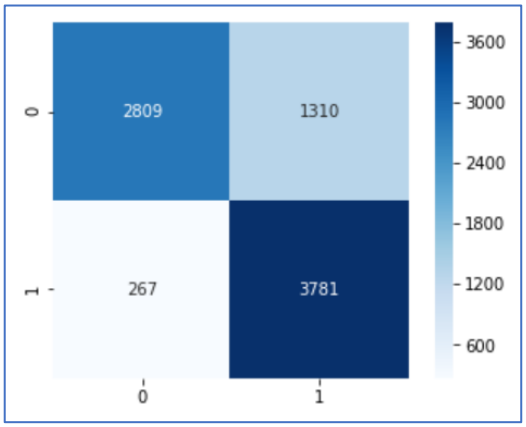
- Specificity: 0.92
- Recall: 0.14
- Precision (Male): 0.64
- Precision (Female): 0.52
- Accuracy: 0.53

CNN model trained with ~7800 images (5% of filtered data set):



- Specificity: 0.78
- Recall: 0.73
- Precision (Male): 0.77
- Precision (Female): 0.75
- Accuracy: 0.76

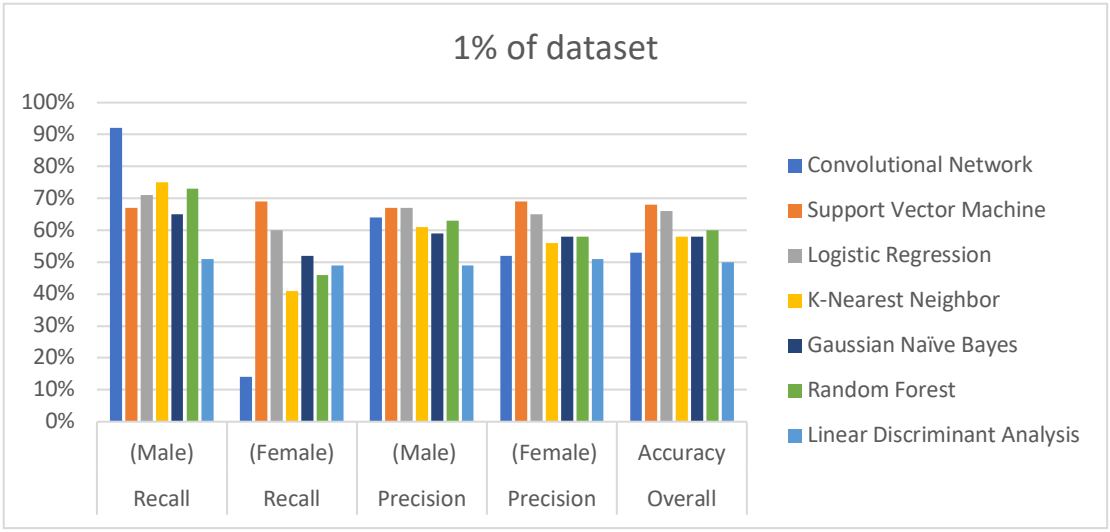
CNN model trained with ~15000 images (10% of filtered data set):

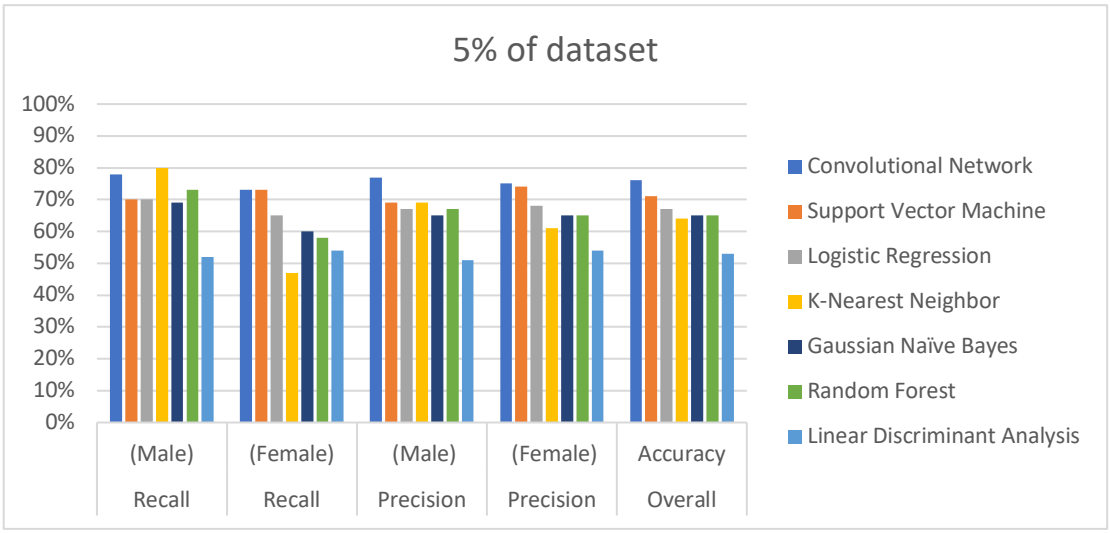


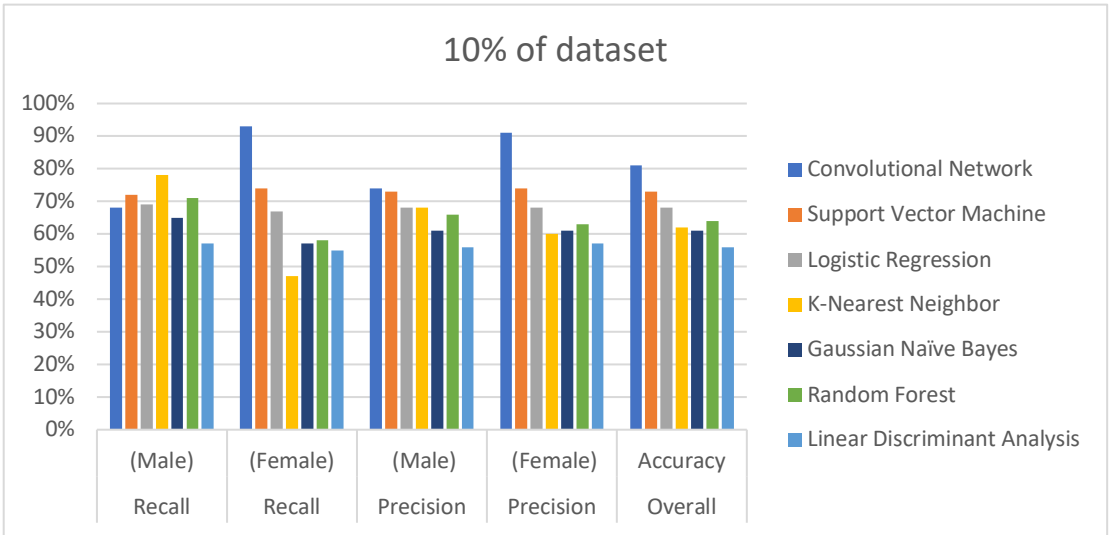
- Specificity: 0.68
- Recall: 0.93
- Precision (Male): 0.74
- Precision (Female): 0.91
- Accuracy: 0.81

Comparison

The learning experiments show that the CNN classifier performs more poorly than the traditional ML classifiers when the dataset is small. However, as the dataset increases, CNN displays a decisive advantage as the dataset increases as can be seen in the figurers below.


1% of dataset
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
(Male) Recall
(Female) Recall
(Male) Precision
(Female) Precision
Accuracy Overall
Convolutional Network
Support Vector Machine
Logistic Regression
K-Nearest Neighbor
Gaussian Naïve Bayes
Random Forest
Linear Discriminant Analysis


5% of dataset
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
(Male) Recall
(Female) Recall
(Male) Precision
(Female) Precision
Accuracy Overall
Convolutional Network
Support Vector Machine
Logistic Regression
K-Nearest Neighbor
Gaussian Naïve Bayes
Random Forest
Linear Discriminant Analysis


10% of dataset
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
(Male) Recall
(Female) Recall
(Male) Precision
(Female) Precision
Accuracy Overall
Convolutional Network
Support Vector Machine
Logistic Regression
K-Nearest Neighbor
Gaussian Naïve Bayes
Random Forest
Linear Discriminant Analysis

Conclusion

The SVM resulted in the highest accuracy among the traditional machine learning classifiers with an overall accuracy of 73.3%. The best performing CNN model resulted in an overall accuracy of 90.0%.

Both the SVM model and CNN models resulted in similar accuracies for their prediction of female versus male faces. The SVM model correctly predicted male and female faces 74.4% and 72.4% of the time respectively, and the CNN model correctly predicted male and female faces 90.1% and 88.8% of the time.

The traditional machine classifiers generally performed better than the CNN model when the dataset was small (less than 2000 images).

Appendix

Comparison of all classifiers

Training Set Size	Model	Recall (male)	Recall (female)	Precision (Male)	Precision (Female)	Overall Accuracy
1,500	Convolutional Network	0.92	0.14	0.64	0.52	0.53
	Support Vector Machine	0.67	0.69	0.67	0.69	0.68
	Logistic Regression	0.71	0.60	0.67	0.65	0.66
	K-Nearest Neighbor	0.75	0.41	0.61	0.56	0.58
	Linear Discriminant Analysis	0.51	0.49	0.49	0.51	0.50
	Gaussian Naïve Bayes	0.65	0.52	0.59	0.58	0.58
	Random Forest	0.73	0.46	0.63	0.58	0.60
7,800	Convolutional Network	0.78	0.73	0.77	0.75	0.76
	Support Vector Machine	0.70	0.73	0.69	0.74	0.71
	Logistic Regression	0.70	0.65	0.67	0.68	0.67
	K-Nearest Neighbor	0.80	0.47	0.69	0.61	0.64
	Linear Discriminant Analysis	0.52	0.54	0.51	0.54	0.53

	Gaussian Naïve Bayes	0.69	0.60	0.65	0.65	0.65
	Random Forest	0.73	0.58	0.67	0.65	0.65
15,000	Convolutional Network	0.68	0.93	0.74	0.91	0.81
	Support Vector Machine	0.72	0.74	0.73	0.74	0.73
	Logistic Regression	0.69	0.67	0.68	0.68	0.68
	K-Nearest Neighbor	0.78	0.47	0.68	0.60	0.62
	Linear Discriminant Analysis	0.57	0.55	0.56	0.57	0.56
	Gaussian Naïve Bayes	0.65	0.57	0.61	0.61	0.61
	Random Forest	0.71	0.58	0.66	0.63	0.64
150,000	Convolutional Network	0.89	0.91	0.89	0.91	0.90

References

[1] S. A. Khan, M. Ahmad, M. Nazir, and N. Riaz, "A comparative analysis of gender classification techniques," *Middle-East Journal of Scientific Research*, vol. 20, no. 1, pp. 1–13, 2014.

[2] M. Duan, K. Li, C. Yang, and K. Li, "A hybrid deep learning cnn–elm for age and gender classification," *Neurocomputing*, vol. 275, pp. 448–461, 2018.

[3] M. Nazir, M. Ishtiaq, A. Batool, M. A. Jaffar, and A. M. Mirza, "Feature selection for efficient gender classification," in *Proceedings of the 11th WSEAS international conference*, 2010, pp. 70–75.

[4] B. Moghaddam and M.-H. Yang, "Learning gender with support faces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 707–711, 2002.

[5] A. F. Basha and G. S. B. Jahangeer, "Exploring a novel method for face

image gender classification using random forest and comparing with other machine learning techniques," *International Journal of Computer Science Issues (IJCSI)*, vol. 11, no. 6, p. 8, 2014.
[6] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.
[7] R. Rothe, R. Timofte, and L. Van Gool, "Dex: Deep expectation of apparent age from a single image," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 10–15.