AI-based celebrity classification using CNN

Jaea Cho

0032077157

CNIT 48300

Jin Wei-Kocsis

May 4th, 2022

Abstract

The classification of images is a common practice today, especially for social networks and personal electronic devices. This study developed a convolutional neural network (CNN) model that is capable of classifying any individual with sufficient data. This project uses Tensorflow to classify 4 different people: Angelina Jolie, Barack Obama, Lionel Messi, and Yuna Kim. Imagery is retrieved from web crawling, and the data is separated into two parts: a training set and a test set. Each class imports 60 images total of 240. 200 images are used for training and 40 images are used for testing. Using random images from the training set, it was possible to assess if each generation was successful. To preprocess the data for a model, data were rescaled and turned into batches. The model was created with the typical CNN structure, but to fit the machine learning model better, the max-pooling method was used. In order to prevent overfitting, hidden units in each layer were reduced. The trained model is used to make predictions after all the data is imported and the model has been built. Using random images of 3 different people, the model succeeded in making the prediction. A high accuracy value of 0.91 is found for the training data, but it ends at 0.63 for the accuracy value.

Keywords: Artificial Intelligence, image crawling, TensorFlow, classification

Overview and Motivation

In the field of biometrics, facial recognition has been studied extensively. Face verifications are used for a variety of purposes, including access to mobile applications, payment methods, and second authentication methods. A number of mobile applications have also been developed that attempt to classify a person's face, but facial authentication remains a complex process, as there are still many errors. Since the use of facial recognition is increasing and becoming more diverse, it seems crucial to develop a simple but powerful classification model. This project aims to extract different faces from the internet and classify them into the right classes. As the training data is already classified prior to training, the classification process helps to accelerate the training. With enough data, any person can be classified into the right category. Although categorical class mode was used in this project, other options such as binary class and sparse class could lead to additional results. Given the face data of just one individual, the binary class model could determine if it is that person or not.

Literature Review

Relevant Work

There is much to be learned from the face of a person. When looking at the face, one can determine the person's age, gender, and emotional state. Specifically, robots can be taught to speak with a particular attitude and language based on their owner's age. Age classification, especially utilizing real-world or wild face images, can be a challenging task, yet its relevance arises from its wide range of applications. In this paper, pre-trained CNN is used to improve the performance of the designed CNNs architectures. The difference between face recognition and

The main difference between age classification and face recognition is that the benchmarks used in age classifications are smaller than those used in face recognition. As a result, there is a critical overfitting problem, but this problem can be solved by applying transfer learning techniques. A comprehensive performance evaluation is conducted in this paper and a comparison is made with six pre-trained CNN architectures. The study shows that not only does the number of training pictures and subject matter affect performance, but also the pre-training task. (Mallouh, 2019)

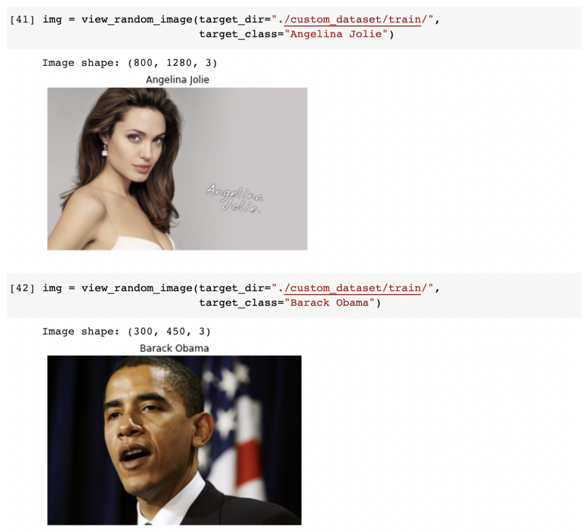
Contributions of the Project

This paper has given me the insight that overfitting a model, which performs well on training data, but not so well on unobserved data, cannot be effectively used in a real-life situation. Consequently, I have tried ConvNet with maximum pooling, and have also augmented the data set. The augmentation of the training data resulted in a reduction of the accuracy value too much, and this decision was not made. The overfitting was also reduced to some extent by simplifying the model. Methods that I utilized included obtaining more data, simplifying the model, and using data augmentation. Further studies will be carried out through the application of a transfer learning method in order to leverage the patterns.

This paper presents a model that provides simple, but meaningful results for the classification of human faces. The time required for the entire operation is very short, making it possible to extract very meaningful information. Using that technology, simple quiz games could be created using fewer graphics processing units or an application that could help to predict who celebrities look like.

Technology Details

Gathering training images for the image crawling. In order to crawl images from the website, an image downloader library was installed. The directories were divided into training directories and test directories. The ratio was 5:1. Following the creation of the train/test directory, data was divided into the initial ratio set by a function. Four different classes of images are then crawled. To verify that all the images are successfully crawled into the right category, random images are extracted from each class.

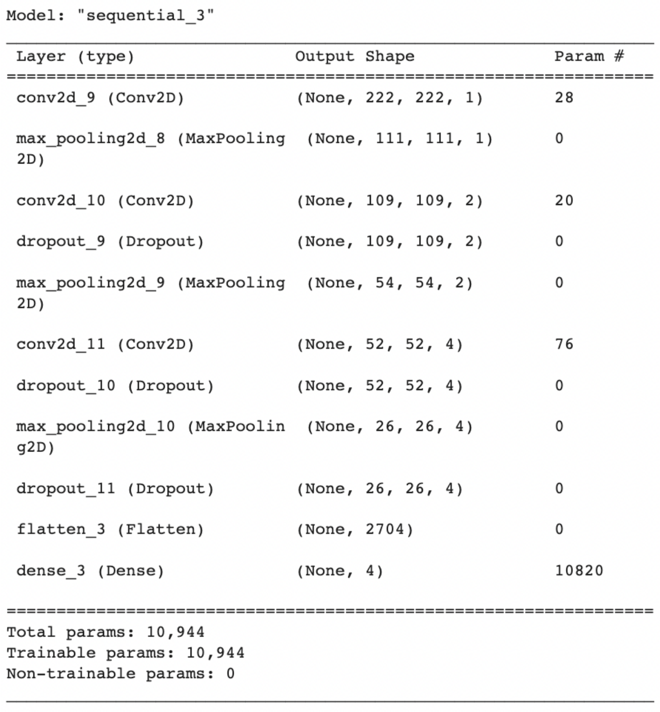


```
[41] img = view_random_image(target_dir="./custom_dataset/train/",
                             target_class="Angelina Jolie")

     Image shape: (800, 1280, 3)
```

```
[42] img = view_random_image(target_dir="./custom_dataset/train/",
                             target_class="Barack Obama")

     Image shape: (300, 450, 3)
```

Fig. 1. Random extract crawled images

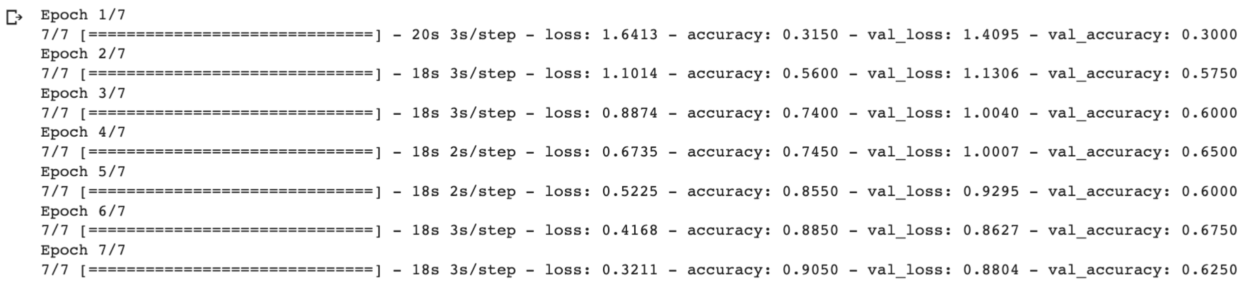
Building the model. Convolutional neural networks consist of input images, input layer, convolution layer, hidden activation, pooling layer, fully connected layer, an output layer, and output activation. For the input image, as shown in Figure 1, crawling is used. The input layer functions as a preprocessor for subsequent layers based on the target image. In code, typical values can be represented as : input_shape = [batch_size, image_height, image_width, color_channels]. Processed inputs will get represented as a tensor. For the convolutional layer, I used Conv2D and it will extract the most important features from the target images. Relu was used as hidden activation and max pooling was used to reduce the dimension of learned image features. The fully connected layer refines learned features, and the output layer outputs learned features in the form of target labels. Since the class mode is categorical, softmax output activation was used.



```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 222, 222, 1)       28

 max_pooling2d_8 (MaxPooling  (None, 111, 111, 1)      0
 2D)

 conv2d_10 (Conv2D)          (None, 109, 109, 2)       20

 dropout_9 (Dropout)         (None, 109, 109, 2)       0

 max_pooling2d_9 (MaxPooling  (None, 54, 54, 2)        0
 2D)

 conv2d_11 (Conv2D)          (None, 52, 52, 4)         76

 dropout_10 (Dropout)        (None, 52, 52, 4)         0

 max_pooling2d_10 (MaxPoolin  (None, 26, 26, 4)        0
 g2D)

 dropout_11 (Dropout)        (None, 26, 26, 4)         0

 flatten_3 (Flatten)         (None, 2704)              0

 dense_3 (Dense)             (None, 4)                 10820

=================================================================
Total params: 10,944
Trainable params: 10,944
Non-trainable params: 0
_________________________________________________________________
```

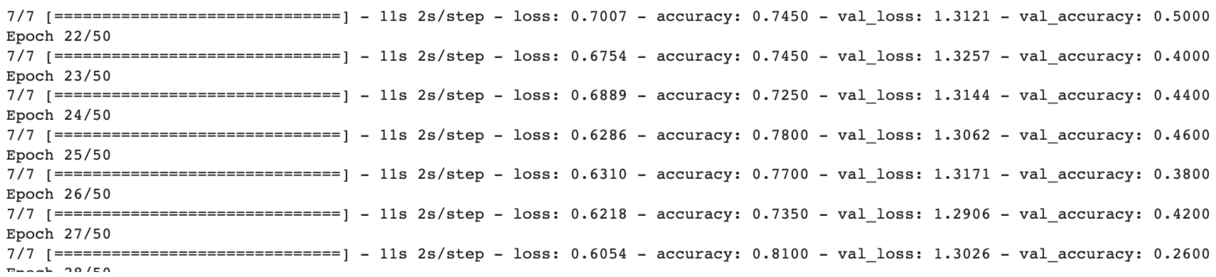
Fig. 2. Summary of the model

The role of data augmentation is to avoid overfitting as explained above. Figure 3.1 shows that the accuracy of the training data differs from the accuracy of the real data. Figure 3.2. shows the training set with augmentation, but the problem here is that too many epochs are needed to gain 90 percent accuracy for the training set. Still then, the accuracy of the real data is not so high.



```
Epoch 1/7
7/7 [==============================] - 20s 3s/step - loss: 1.6413 - accuracy: 0.3150 - val_loss: 1.4095 - val_accuracy: 0.3000
Epoch 2/7
7/7 [==============================] - 18s 3s/step - loss: 1.1014 - accuracy: 0.5600 - val_loss: 1.1306 - val_accuracy: 0.5750
Epoch 3/7
7/7 [==============================] - 18s 3s/step - loss: 0.8874 - accuracy: 0.7400 - val_loss: 1.0040 - val_accuracy: 0.6000
Epoch 4/7
7/7 [==============================] - 18s 2s/step - loss: 0.6735 - accuracy: 0.7450 - val_loss: 1.0007 - val_accuracy: 0.6500
Epoch 5/7
7/7 [==============================] - 18s 2s/step - loss: 0.5225 - accuracy: 0.8550 - val_loss: 0.9295 - val_accuracy: 0.6000
Epoch 6/7
7/7 [==============================] - 18s 3s/step - loss: 0.4168 - accuracy: 0.8850 - val_loss: 0.8627 - val_accuracy: 0.6750
Epoch 7/7
7/7 [==============================] - 18s 3s/step - loss: 0.3211 - accuracy: 0.9050 - val_loss: 0.8804 - val_accuracy: 0.6250
```

Fig. 3.1. Without Augmentation



```
7/7 [==============================] - 11s 2s/step - loss: 0.7007 - accuracy: 0.7450 - val_loss: 1.3121 - val_accuracy: 0.5000
Epoch 22/50
7/7 [==============================] - 11s 2s/step - loss: 0.6754 - accuracy: 0.7450 - val_loss: 1.3257 - val_accuracy: 0.4000
Epoch 23/50
7/7 [==============================] - 11s 2s/step - loss: 0.6889 - accuracy: 0.7250 - val_loss: 1.3144 - val_accuracy: 0.4400
Epoch 24/50
7/7 [==============================] - 11s 2s/step - loss: 0.6286 - accuracy: 0.7800 - val_loss: 1.3062 - val_accuracy: 0.4600
Epoch 25/50
7/7 [==============================] - 11s 2s/step - loss: 0.6310 - accuracy: 0.7700 - val_loss: 1.3171 - val_accuracy: 0.3800
Epoch 26/50
7/7 [==============================] - 11s 2s/step - loss: 0.6218 - accuracy: 0.7350 - val_loss: 1.2906 - val_accuracy: 0.4200
Epoch 27/50
7/7 [==============================] - 11s 2s/step - loss: 0.6054 - accuracy: 0.8100 - val_loss: 1.3026 - val_accuracy: 0.2600
```

Fig. 3.2. With Augmentation

Instead, I chose to drop out layers to prevent overfitting without augmentation. The model accuracy and loss graph are as follows.

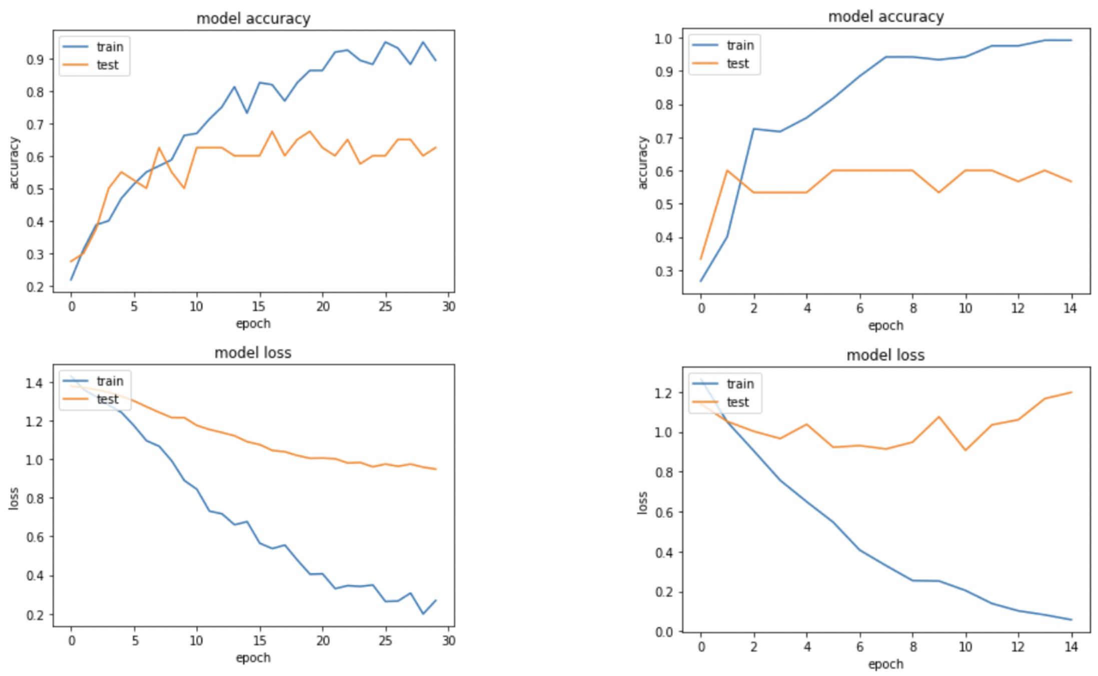
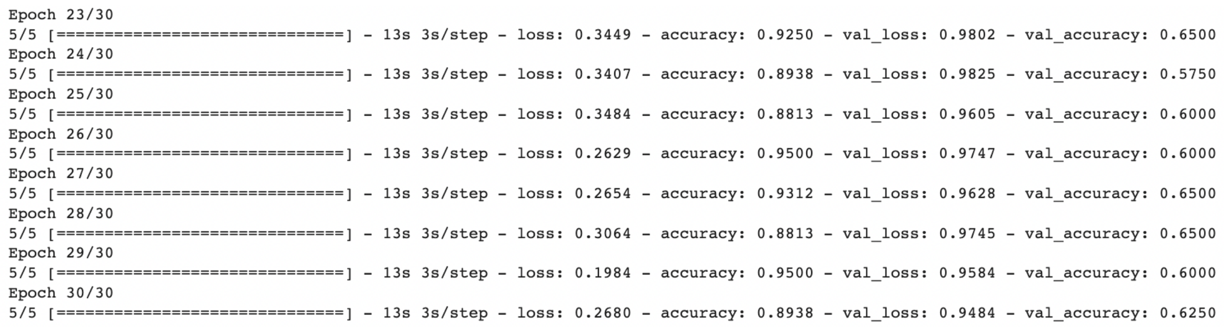


Fig. 4. Without augmentation(left), with augmentation(right)

Validating the model. In order to validate the model, raw sample data were chosen from the internet to verify the model. The image is decoded in to a tensor with 3 ensured colour channels. Image is then resized and rescaled and is loaded to preprocess. After another dimension is added to axis 0, model can not make a prediction of the custom image tensor. The model brings out the prediction and with the 'pred_and_plot' function and the predictions are visualized.

Performance Evaluations

The main goal of the project was to successfully generate and match the person's name with the class. Two out of three generations were successful as seen in outcome 2. The model itself did not have high accuracy as it was between 60~70% at most times. The training data's accuracy was close to 100%. In figure 6, it can be seen that the val_loss is gradually decreasing implying that the effor of reducing overfitting was successful.



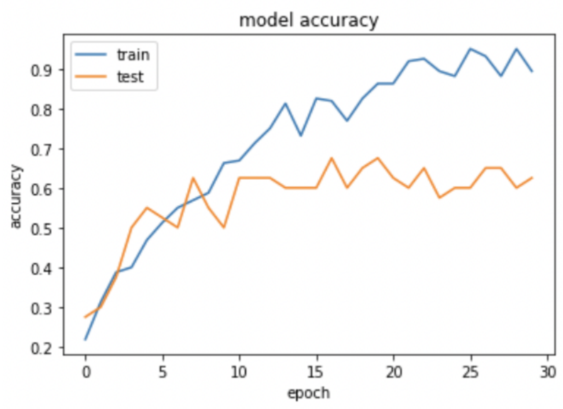
```
Epoch 23/30
5/5 [==============================] - 13s 3s/step - loss: 0.3449 - accuracy: 0.9250 - val_loss: 0.9802 - val_accuracy: 0.6500
Epoch 24/30
5/5 [==============================] - 13s 3s/step - loss: 0.3407 - accuracy: 0.8938 - val_loss: 0.9825 - val_accuracy: 0.5750
Epoch 25/30
5/5 [==============================] - 13s 3s/step - loss: 0.3484 - accuracy: 0.8813 - val_loss: 0.9605 - val_accuracy: 0.6000
Epoch 26/30
5/5 [==============================] - 13s 3s/step - loss: 0.2629 - accuracy: 0.9500 - val_loss: 0.9747 - val_accuracy: 0.6000
Epoch 27/30
5/5 [==============================] - 13s 3s/step - loss: 0.2654 - accuracy: 0.9312 - val_loss: 0.9628 - val_accuracy: 0.6500
Epoch 28/30
5/5 [==============================] - 13s 3s/step - loss: 0.3064 - accuracy: 0.8813 - val_loss: 0.9745 - val_accuracy: 0.6500
Epoch 29/30
5/5 [==============================] - 13s 3s/step - loss: 0.1984 - accuracy: 0.9500 - val_loss: 0.9584 - val_accuracy: 0.6000
Epoch 30/30
5/5 [==============================] - 13s 3s/step - loss: 0.2680 - accuracy: 0.8938 - val_loss: 0.9484 - val_accuracy: 0.6250
```

Figure. 6. Model fitting history

In an effort to reduce overfitting, I made the simpler model and dropped out some layers, added max pooling, and drop outs. Although the model did not perform as well on the test set as on the training set, it might be because of a lack of data. For the greater collection of data, it would take a larger GPU and a longer processing time, so it was unnecessary.

Outcome 1

According to the model accuracy graph, train data accuracy is higher than test data accuracy. Although it fluctuates, it still follows the growth graph and the loss curves and model accuracy curves are closer to each other. On the other hand, the loss graph follows the opposite path. The training loss and the test loss are both decreasing.



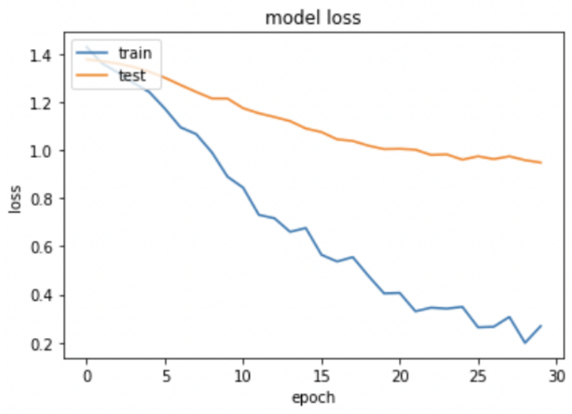


Figure. 7. Model accuracy and loss graph

A few differences between the previous model is that there is one more convolutional layer with fewer filters, a larger drop out, and an additional max pooling layer. I also trained the data set for a longer period of time with more epochs to increase accuracy. While there has been a decrease in the overall accuracy by a small percentage, the model does not show overfitting, and it shows closer curves.

Outcome 2

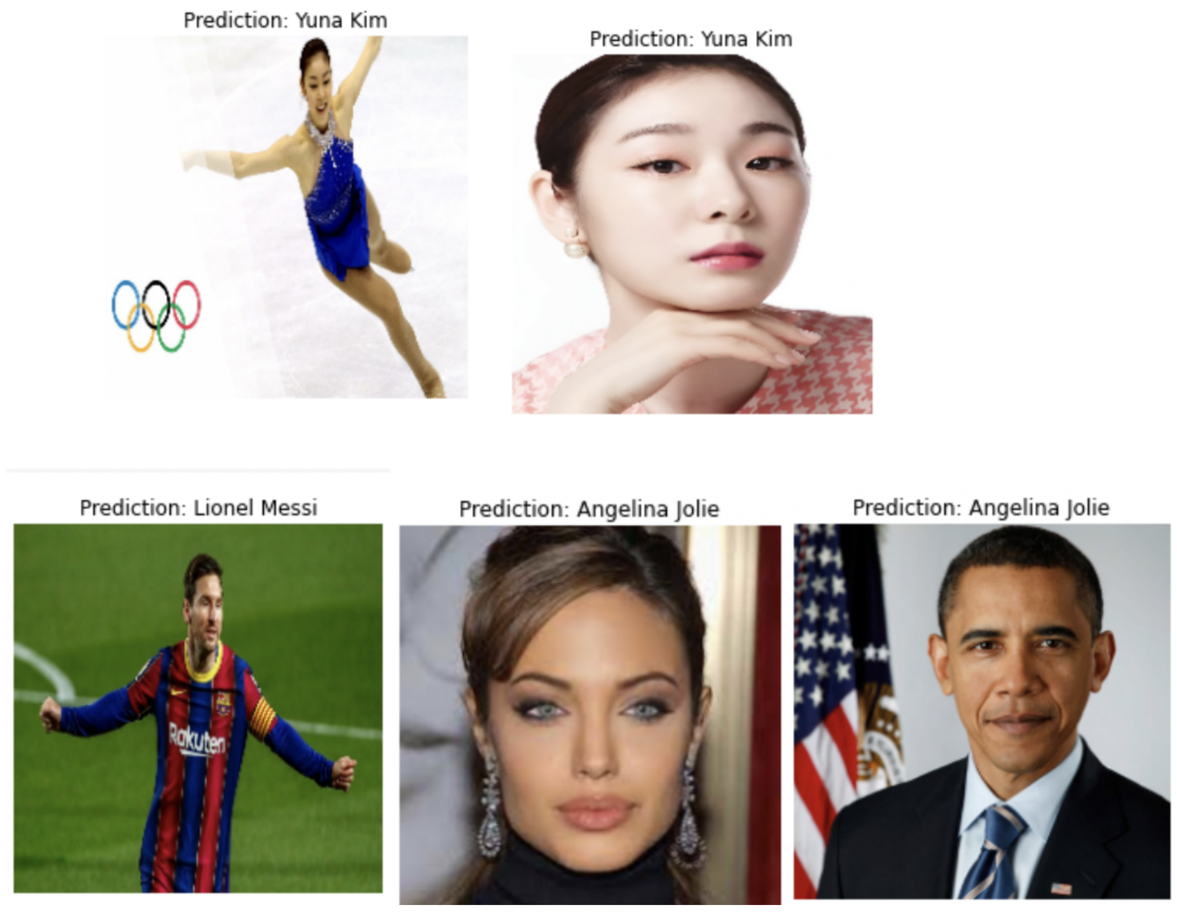


Figure. 8. Model validation visualization

The visualization of the model prediction can be seen in Figure 8. There is a prediction error with sample image of Barack Obama and the model made a prediction that the image is Angelina Jolie. This is because the model is performing at ~65% accuracy on the test dataset.

Conclusions

Even though the accuracy did not meet expectations, the intended result came out successfully. The sub-goal of this research was to crawl the data, use visualization to confirm the datasets and results. The main goal was to guess unknown images to right class and show through visualization. This study could be furthered on using real-world photo taken with various devices.

Finally, I am to include a brief discussion by gathering all the relevant dat provided along this research. Next, I would like to present itemize analysis in order to conclude the review of this research and to exhibit future directions.

- Hyper-parameter selection has a significant impact on CNN performance. Various modifications have been made to the model, changing the hyper-parameter slightly. There were significant differences in the results of each model.
- Crawled images showed low accuracy level compared to mnist dataset. Using this model, it can be defined that the model can be applied to a real-world dataset with appropriate preprocessing.
- An analysis of the model with and without augmentation was conducted, and the model without augmentation delivered a better result in contradiction to the intended outcome. There need to be further study upon why augmentation did not help overfitting of data.
- In future studies, it may be possible to alleviate overfitting through transfer learning in light of the lack of training data.

References

Abu Mallouh, Qawaqneh, Z., & Barkana, B. D. (2019). Utilizing CNNs and transfer learning of pre-trained models for age range classification from unconstrained face images. Image and Vision Computing, 88, 41–51. https://doi.org/10.1016/j.imavis.2019.05.001

Yuda, Aroef, C., Rustam, Z., & Alatas, H. (2020). Gender Classification Based on Face Recognition using Convolutional Neural Networks (CNNs). Journal of Physics. Conference Series, 1490(1), 12042–. https://doi.org/10.1088/1742-6596/1490/1/012042