The Simpsons Characters

Sijie Huang, Hanying Shen, Yijia Sun

1. Ask a Question

- Simpson character image dataset

- To distinguish the characters of Homer Simpson and Lisa Simpson

- Use CNN and Transfer Learning to do the binary image classification

2. Get the data.

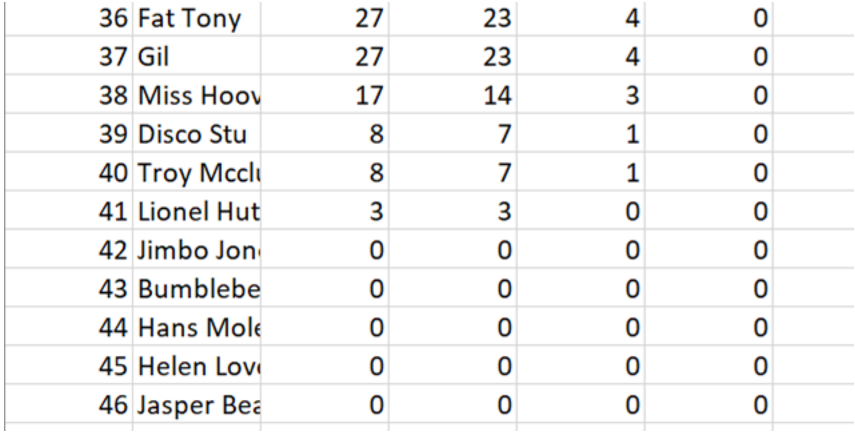
- Kaggle: https://www.kaggle.com/alexattia/the-simpsons-characters-dataset
- 42 classes with 20933 images

- Homer Simpson: 2247 images
- Lisa Simpson: 1354 images

- Data cleaning: add labels to images before training the model
- Related dataset: test set (to test the model at the end)

3. Explore the Data

- Problem with data?
  - The original dataset has 42 classes, and many of them have less than 100 images
  - Too many classes will lead to less accuracy
- How to improve this dataset in the future?
  - Capture more images
  - Reduce to less classes



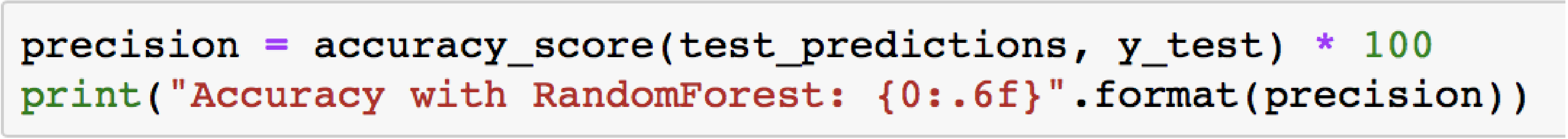
36	Fat Tony	27	23	4	0
37	Gil	27	23	4	0
38	Miss Hoov	17	14	3	0
39	Disco Stu	8	7	1	0
40	Troy Mccl	8	7	1	0
41	Lionel Hut	3	3	0	0
42	Jimbo Jon	0	0	0	0
43	Bumblebe	0	0	0	0
44	Hans Mol	0	0	0	0
45	Helen Lov	0	0	0	0
46	Jasper Bea	0	0	0	0

4. Model the Data

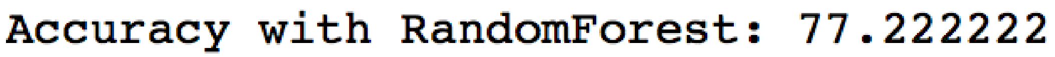
- Image dataset → CNN and Transfer Learning, SVM
- Accuracy: Training and Validation Loss, Confusion Matrix, ROC

5. Select a model and train it.

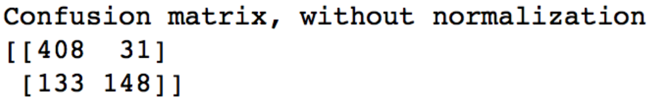
- Random Forest Classifier
- SVM (Support Vector Machine)
- CNN
- Transfer Learning



```
precision = accuracy_score(test_predictions, y_test) * 100
print("Accuracy with RandomForest: {0:.6f}".format(precision))
```

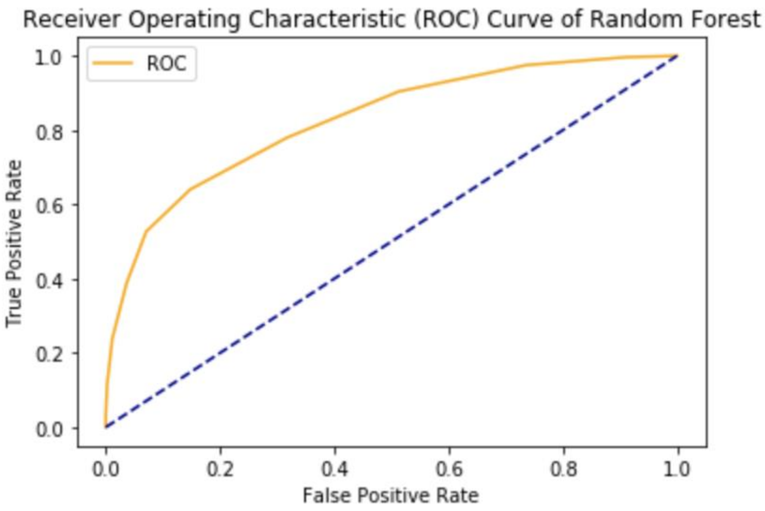


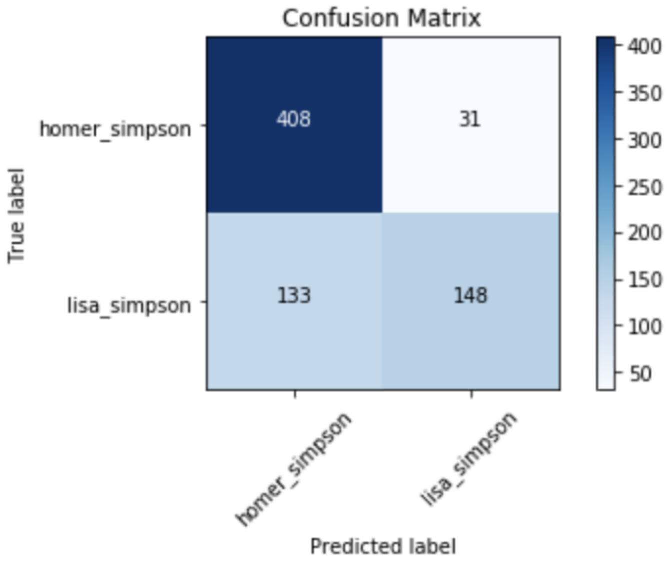
```
Accuracy with RandomForest: 77.222222
```



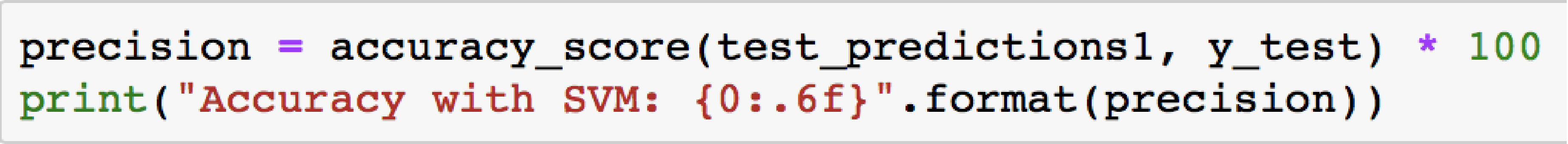
```
Confusion matrix, without normalization
[[408  31]
 [133 148]]
```

Random Forest Classifier



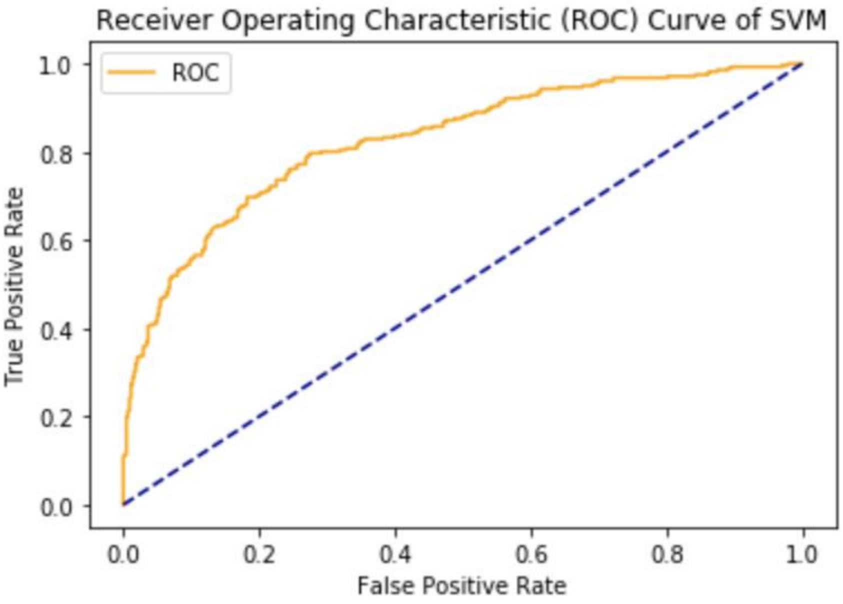


AUC test score: 0.73

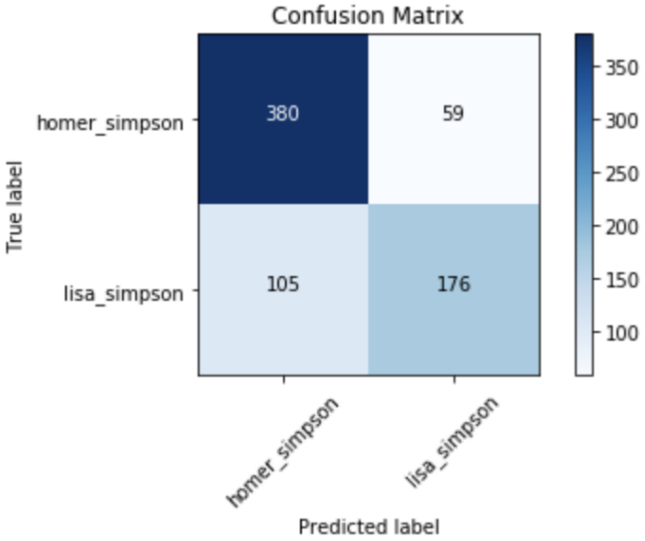


```
precision = accuracy_score(test_predictions1, y_test) * 100
print("Accuracy with SVM: {0:.6f}".format(precision))
```

```
Accuracy with SVM: 77.222222
```

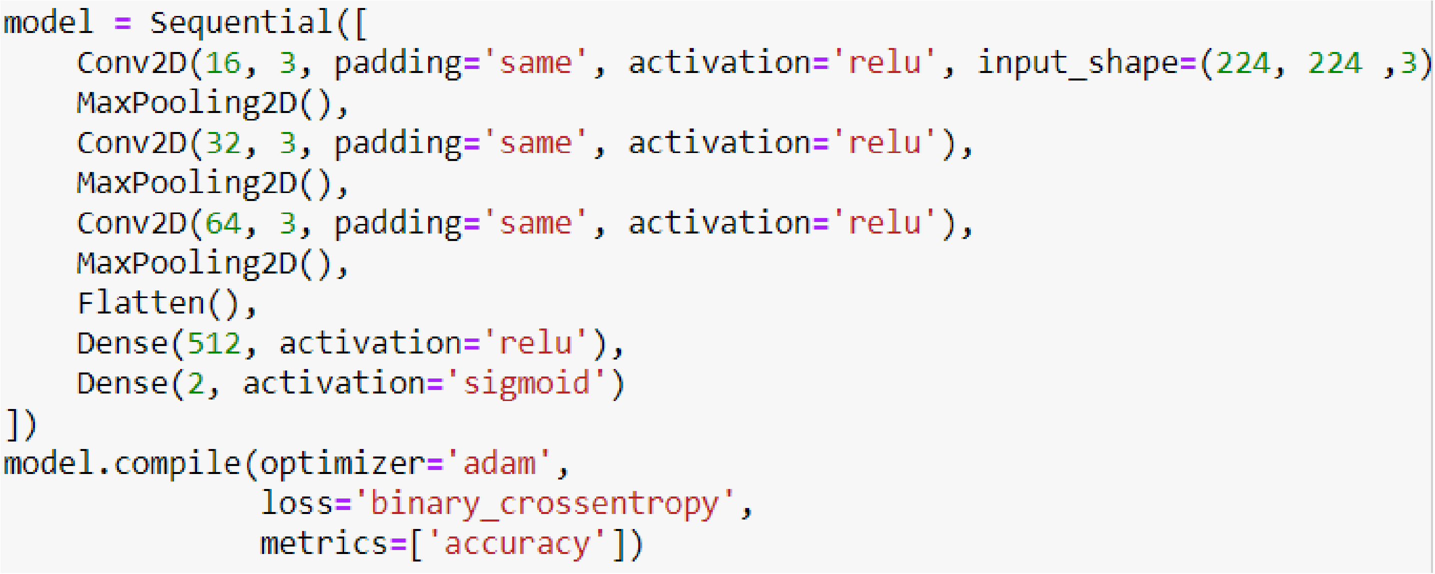


```
Confusion matrix, without normalization
[[380  59]
 [105 176]]
```

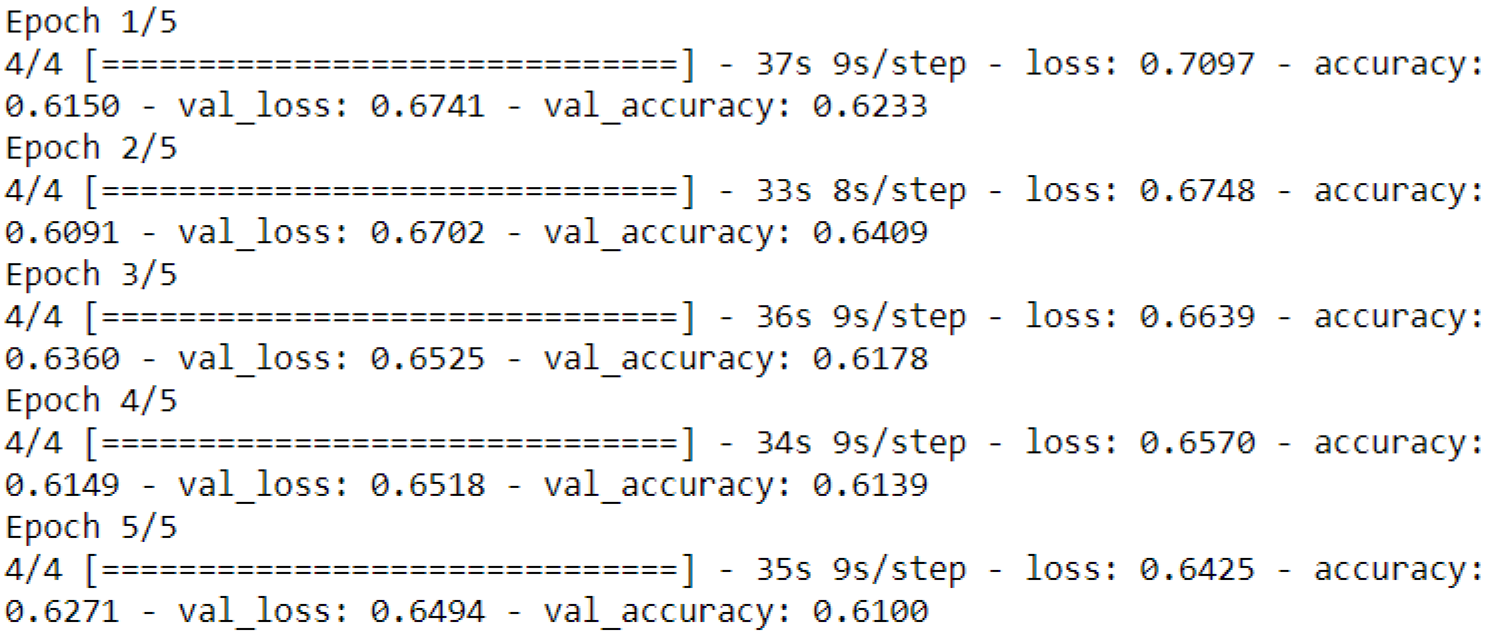


AUC test score: 0.75

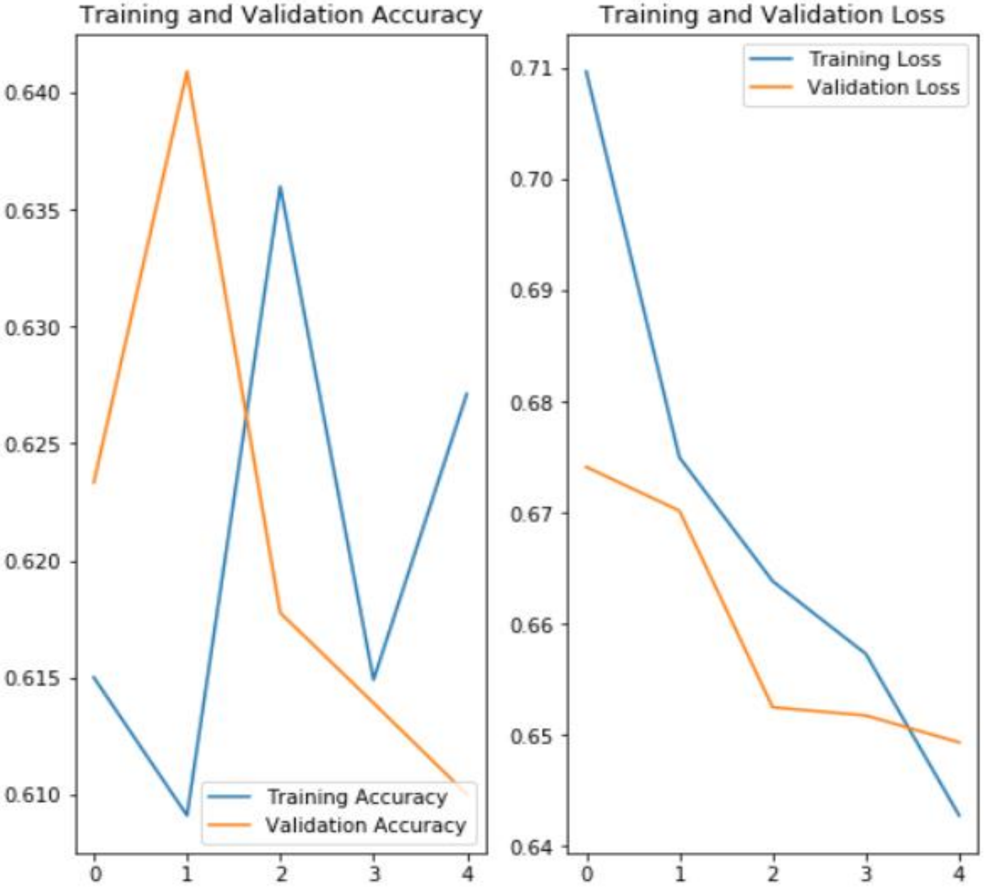
CNN

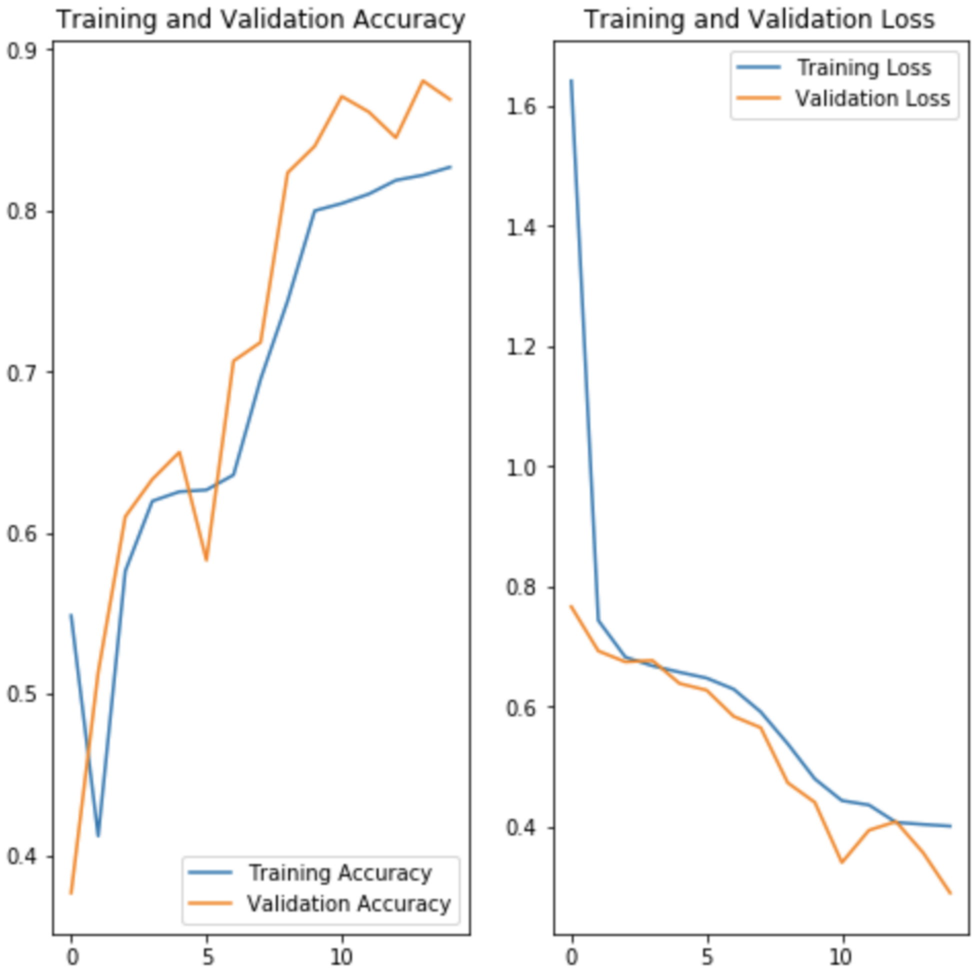


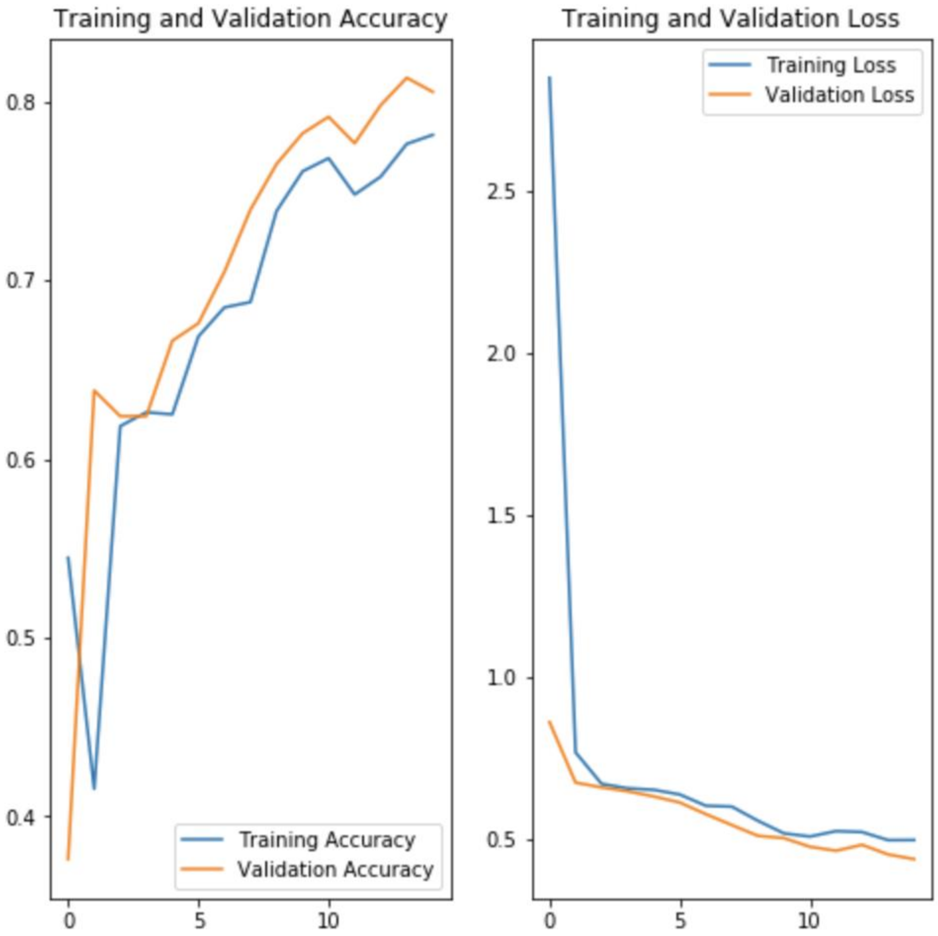
```
model = Sequential([
    Conv2D(16, 3, padding='same', activation='relu', input_shape=(224, 224 ,3)
    MaxPooling2D(),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(2, activation='sigmoid')
])
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```



```
Epoch 1/5
4/4 [==============================] - 37s 9s/step - loss: 0.7097 - accuracy:
0.6150 - val_loss: 0.6741 - val_accuracy: 0.6233
Epoch 2/5
4/4 [==============================] - 33s 8s/step - loss: 0.6748 - accuracy:
0.6091 - val_loss: 0.6702 - val_accuracy: 0.6409
Epoch 3/5
4/4 [==============================] - 36s 9s/step - loss: 0.6639 - accuracy:
0.6360 - val_loss: 0.6525 - val_accuracy: 0.6178
Epoch 4/5
4/4 [==============================] - 34s 9s/step - loss: 0.6570 - accuracy:
0.6149 - val_loss: 0.6518 - val_accuracy: 0.6139
Epoch 5/5
4/4 [==============================] - 35s 9s/step - loss: 0.6425 - accuracy:
0.6271 - val_loss: 0.6494 - val_accuracy: 0.6100
```

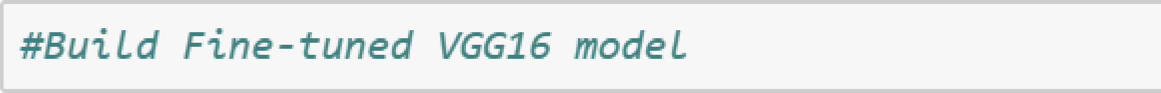

Training and Validation Accuracy
0.640
0.635
0.630
0.625
0.620
0.615
0.610
0
1
2
3
4
Training Accuracy
Validation Accuracy
Training and Validation Loss
0.71
0.70
0.69
0.68
0.67
0.66
0.65
0.64
0
1
2
3
4
Training Loss
Validation Loss


Training and Validation Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
Training Accuracy
Validation Accuracy
Training and Validation Loss
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
5
10
Training Loss
Validation Loss

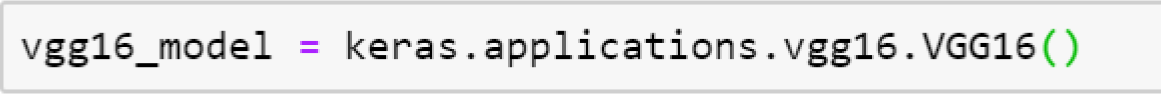

Training and Validation Accuracy
0.8
0.7
0.6
0.5
0.4
0
5
10
Training Accuracy
Validation Accuracy
Training and Validation Loss
Training Loss
Validation Loss
2.5
2.0
1.5
1.0
0.5
0
5
10

6. Fine-tune the model.

Import VGG16 model

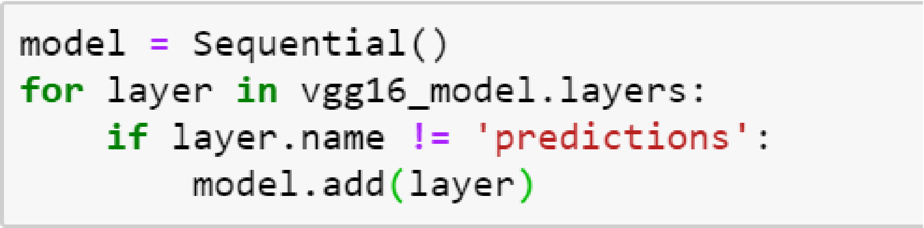


```
#Build Fine-tuned VGG16 model
```



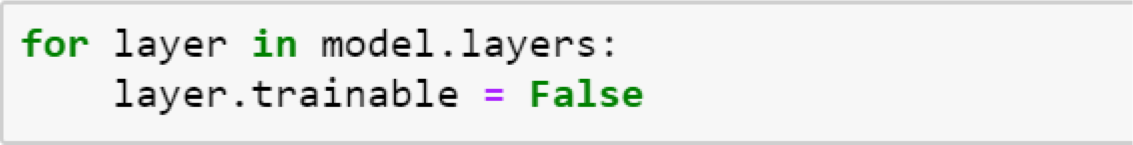
```
vgg16_model = keras.applications.vgg16.VGG16()
```

Freeze all layers besides `predictions`

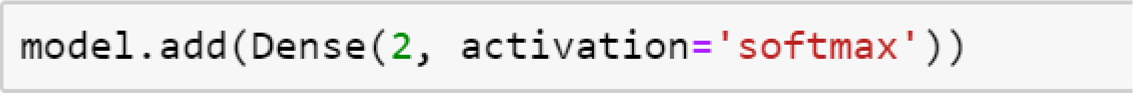


```
model = Sequential()
for layer in vgg16_model.layers:
    if layer.name != 'predictions':
        model.add(layer)
```

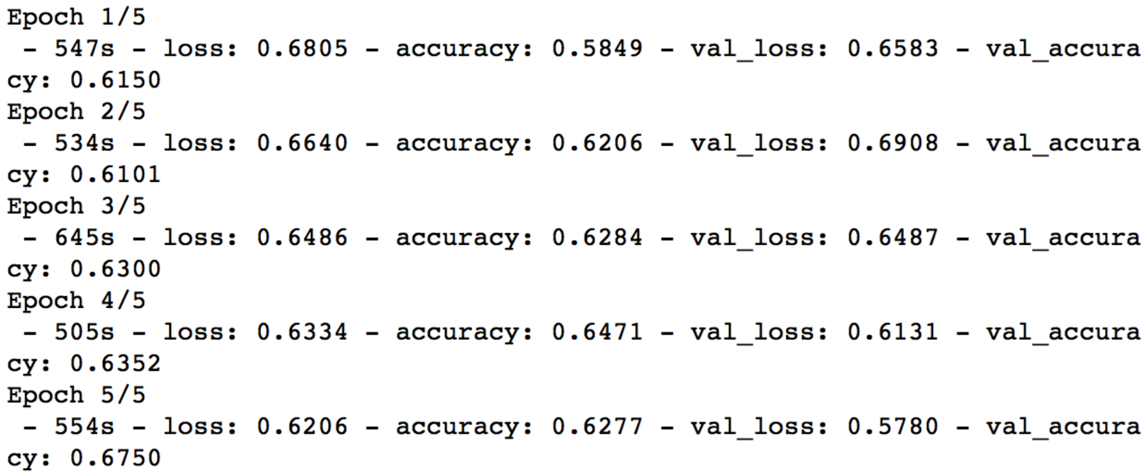
Add our layer to the model



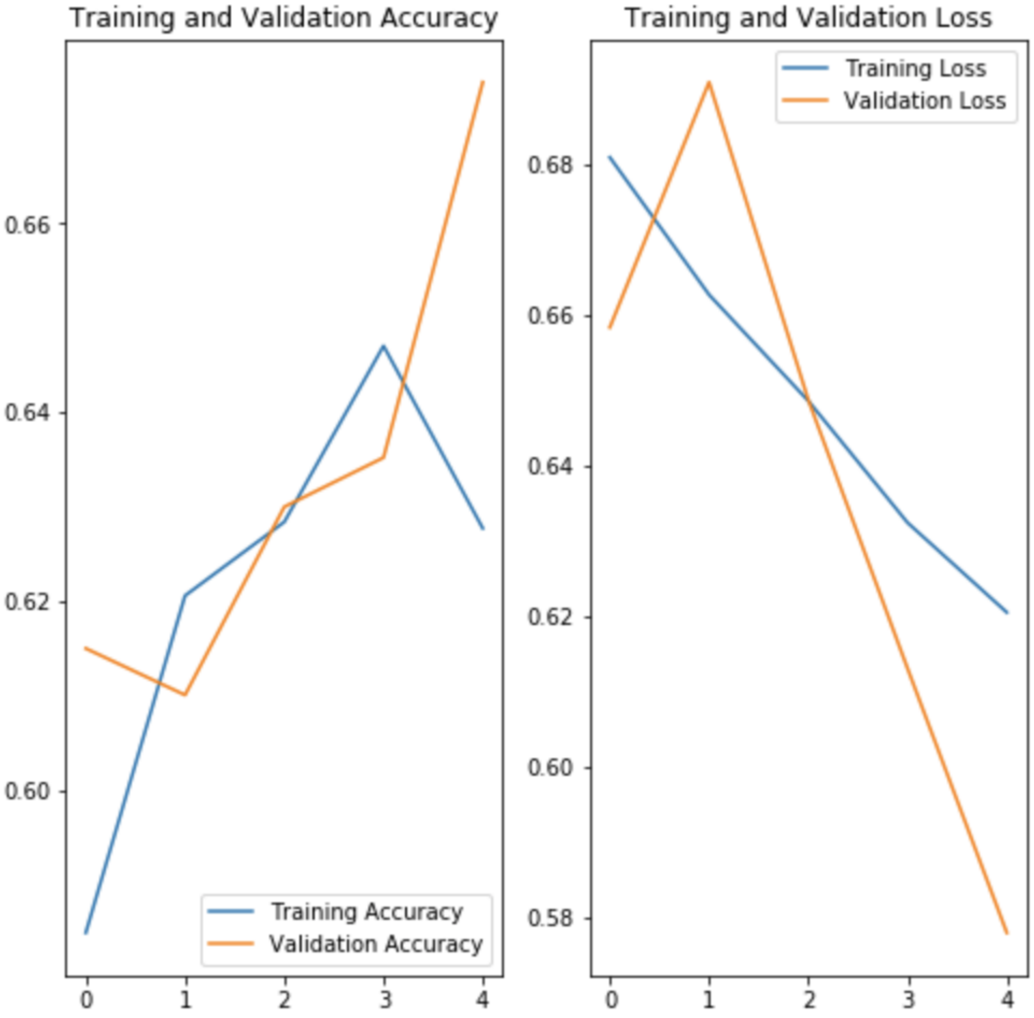
```
for layer in model.layers:
    layer.trainable = False
```



```
model.add(Dense(2, activation='softmax'))
```

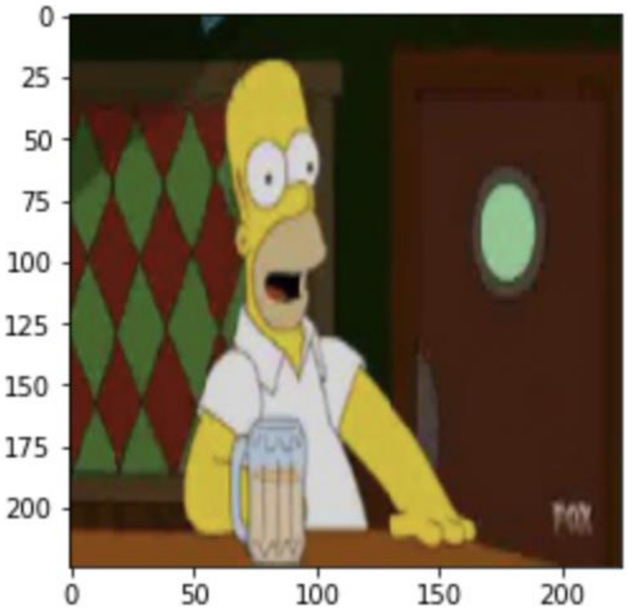


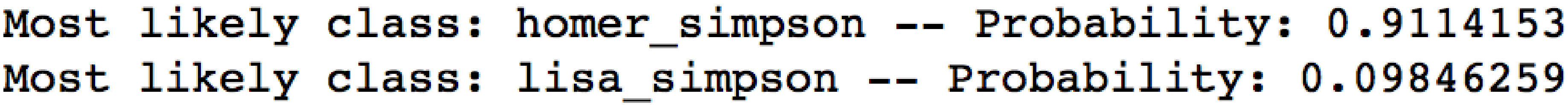
```
Epoch 1/5
 - 547s - loss: 0.6805 - accuracy: 0.5849 - val_loss: 0.6583 - val_accura
cy: 0.6150
Epoch 2/5
 - 534s - loss: 0.6640 - accuracy: 0.6206 - val_loss: 0.6908 - val_accura
cy: 0.6101
Epoch 3/5
 - 645s - loss: 0.6486 - accuracy: 0.6284 - val_loss: 0.6487 - val_accura
cy: 0.6300
Epoch 4/5
 - 505s - loss: 0.6334 - accuracy: 0.6471 - val_loss: 0.6131 - val_accura
cy: 0.6352
Epoch 5/5
 - 554s - loss: 0.6206 - accuracy: 0.6277 - val_loss: 0.5780 - val_accura
cy: 0.6750
```


Training and Validation Accuracy
0.60
0.62
0.64
0.66
0
1
2
3
4
Training Accuracy
Validation Accuracy
Training and Validation Loss
0.58
0.60
0.62
0.64
0.66
0.68
0
1
2
3
4
Training Loss
Validation Loss

Solution

- Upload an image
- Resize it
- Use the model to predict who is it in the image





```
Most likely class: homer_simpson -- Probability: 0.9114153
Most likely class: lisa_simpson -- Probability: 0.09846259
```