manu465-final

December 7, 2021

1 MANU 465 Final Exam - Personality Predictor

1.0.1 Author:

Liam Bontkes, 25530163

1.1 Project Description

This project uses images of shoes to estimate a person's personality. The project uses CNN to

1.2 Importing the Libraries

[ ]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
```

```
2021-12-07 17:59:36.194816: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load
dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open
shared object file: No such file or directory
2021-12-07 17:59:36.194834: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
```

1.3 Data Preprocessing

[117]:

```
from keras.preprocessing.image import ImageDataGenerator

image_data_generator = ImageDataGenerator(rescale=1./255,
                                          shear_range=0.2,
                                          zoom_range=0.2,
                                          horizontal_flip=True)

training_set = image_data_generator.flow_from_directory('training_set',
                                                        target_size=(64, 64),
                                                        batch_size=32,
                                                       ␣
 ↪class_mode='categorical')
```

```
test_set = image_data_generator.flow_from_directory('test_set',
                                                    target_size=(64, 64),
                                                    batch_size=32,
                                                    class_mode='categorical')
```

```
Found 1397 images belonging to 5 classes.
Found 381 images belonging to 5 classes.
```

1.4 CNN Model

1.4.1 Build the CNN Model

[118]:

```
# initialize the model
cnn_model = tf.keras.models.Sequential()
```

[119]:

```
# add and pool first layer
cnn_model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3,␣
 ↪activation='relu', input_shape=[64, 64, 3]))
cnn_model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=1))
```

[120]:

```
# add and pool second layer
cnn_model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3,␣
 ↪activation='relu'))
cnn_model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=1))
```

1.4.2 Flatten the CNN Model

[121]:

```
cnn_model.add(tf.keras.layers.Flatten())
```

1.4.3 Add Connection Layer

[122]:

```
cnn_model.add(tf.keras.layers.Dense(units=128, activation='relu'))
```

1.4.4 Add Output Layers

[123]:

```
# add output node for each category
cnn_model.add(tf.keras.layers.Dense(units=5, activation='sigmoid'))
```

1.4.5 Compile the Model

[124]:

```
cnn_model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
```

1.4.6 Train the Model

[125]:
```
train_history = cnn_model.fit(x=training_set, validation_data=test_set,␣
 ↪epochs=25)
```

```
Epoch 1/25
44/44 [==============================] - 9s 200ms/step - loss: 0.4468 -
accuracy: 0.7359 - val_loss: 0.2773 - val_accuracy: 0.7139
Epoch 2/25
44/44 [==============================] - 9s 206ms/step - loss: 0.1234 -
accuracy: 0.8969 - val_loss: 0.2994 - val_accuracy: 0.7428
Epoch 3/25
44/44 [==============================] - 9s 197ms/step - loss: 0.1035 -
accuracy: 0.9141 - val_loss: 0.2056 - val_accuracy: 0.7927
Epoch 4/25
44/44 [==============================] - 9s 193ms/step - loss: 0.0714 -
accuracy: 0.9499 - val_loss: 0.2216 - val_accuracy: 0.8215
Epoch 5/25
44/44 [==============================] - 9s 199ms/step - loss: 0.0628 -
accuracy: 0.9542 - val_loss: 0.2375 - val_accuracy: 0.8215
Epoch 6/25
44/44 [==============================] - 9s 199ms/step - loss: 0.0559 -
accuracy: 0.9563 - val_loss: 0.2269 - val_accuracy: 0.8241
Epoch 7/25
44/44 [==============================] - 8s 189ms/step - loss: 0.0519 -
accuracy: 0.9664 - val_loss: 0.2398 - val_accuracy: 0.8373
Epoch 8/25
44/44 [==============================] - 8s 180ms/step - loss: 0.0410 -
accuracy: 0.9714 - val_loss: 0.2056 - val_accuracy: 0.8373
Epoch 9/25
44/44 [==============================] - 9s 209ms/step - loss: 0.0422 -
accuracy: 0.9664 - val_loss: 0.2261 - val_accuracy: 0.8294
Epoch 10/25
44/44 [==============================] - 9s 200ms/step - loss: 0.0363 -
accuracy: 0.9742 - val_loss: 0.2274 - val_accuracy: 0.8241
Epoch 11/25
44/44 [==============================] - 9s 193ms/step - loss: 0.0352 -
accuracy: 0.9785 - val_loss: 0.1941 - val_accuracy: 0.8556
Epoch 12/25
44/44 [==============================] - 8s 190ms/step - loss: 0.0422 -
accuracy: 0.9692 - val_loss: 0.2061 - val_accuracy: 0.8661
Epoch 13/25
44/44 [==============================] - 8s 191ms/step - loss: 0.0354 -
accuracy: 0.9742 - val_loss: 0.2369 - val_accuracy: 0.8425
Epoch 14/25
44/44 [==============================] - 8s 185ms/step - loss: 0.0353 -
accuracy: 0.9735 - val_loss: 0.2078 - val_accuracy: 0.8661
Epoch 15/25
```

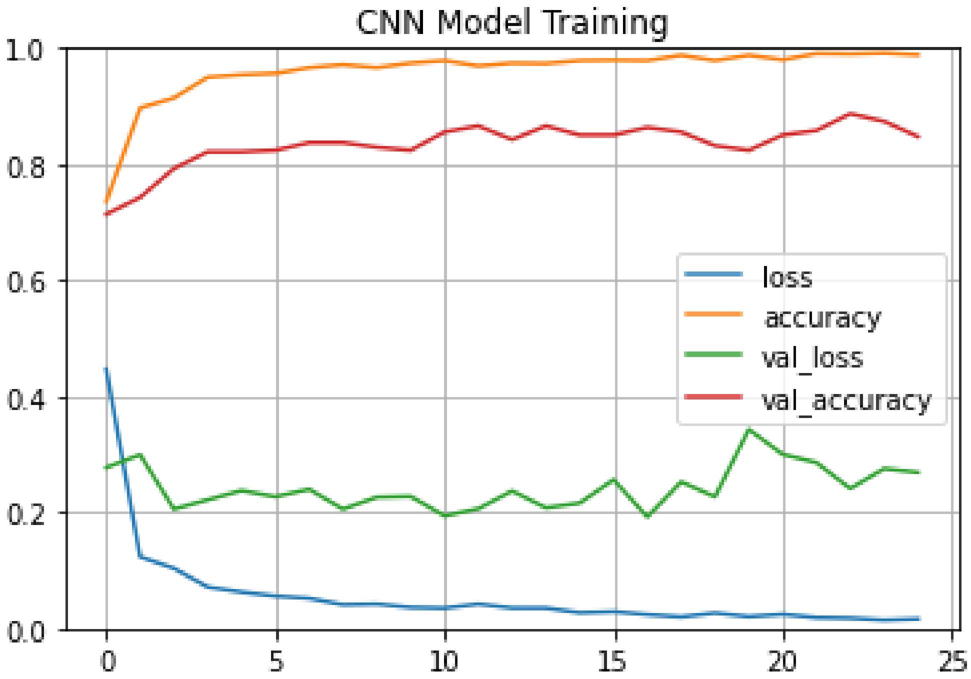
```
44/44 [==============================] - 8s 179ms/step - loss: 0.0272 -
accuracy: 0.9785 - val_loss: 0.2159 - val_accuracy: 0.8504
Epoch 16/25
44/44 [==============================] - 8s 181ms/step - loss: 0.0290 -
accuracy: 0.9792 - val_loss: 0.2563 - val_accuracy: 0.8504
Epoch 17/25
44/44 [==============================] - 8s 183ms/step - loss: 0.0240 -
accuracy: 0.9785 - val_loss: 0.1922 - val_accuracy: 0.8635
Epoch 18/25
44/44 [==============================] - 8s 186ms/step - loss: 0.0200 -
accuracy: 0.9878 - val_loss: 0.2525 - val_accuracy: 0.8556
Epoch 19/25
44/44 [==============================] - 8s 181ms/step - loss: 0.0267 -
accuracy: 0.9785 - val_loss: 0.2268 - val_accuracy: 0.8320
Epoch 20/25
44/44 [==============================] - 8s 190ms/step - loss: 0.0205 -
accuracy: 0.9878 - val_loss: 0.3430 - val_accuracy: 0.8241
Epoch 21/25
44/44 [==============================] - 8s 184ms/step - loss: 0.0244 -
accuracy: 0.9800 - val_loss: 0.3001 - val_accuracy: 0.8504
Epoch 22/25
44/44 [==============================] - 8s 181ms/step - loss: 0.0188 -
accuracy: 0.9907 - val_loss: 0.2859 - val_accuracy: 0.8583
Epoch 23/25
44/44 [==============================] - 8s 181ms/step - loss: 0.0177 -
accuracy: 0.9900 - val_loss: 0.2410 - val_accuracy: 0.8871
Epoch 24/25
44/44 [==============================] - 8s 183ms/step - loss: 0.0145 -
accuracy: 0.9914 - val_loss: 0.2753 - val_accuracy: 0.8740
Epoch 25/25
44/44 [==============================] - 8s 184ms/step - loss: 0.0164 -
accuracy: 0.9885 - val_loss: 0.2689 - val_accuracy: 0.8478
```

[126]:

```python
# plot the accuracy and loss

performance = pd.DataFrame(train_history.history)

performance.plot()
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.title("CNN Model Training")
plt.show()
```



1.5 Make a Prediction

[128]:

```
from keras.preprocessing import image

prediction_image = image.load_img('single_prediction.jpg', target_size=(64, 64))
prediction_image = image.img_to_array(prediction_image)
prediction_image = np.expand_dims(prediction_image, axis=0)

result = cnn_model.predict(prediction_image)

shoe_result = {
    0: 'Athletic',
    1: 'Clogs',
    2: 'Flats',
    3: 'Heels',
    4: 'Loafers'
}

shoe_personality_table = {
    0: 'The person is someone who's very confident, very goal-oriented, and␣
 ↪very organized.',
    1: 'The person is open spirited, and very outdoorsy. They love nature and␣
 ↪the whole regenerative effect of being outdoors.',
    2: 'The person is focused, very modest, and generous.',
    3: 'The person really loves and values beauty.',
    4: 'The person is very responsible, very detail-driven, very much the␣
 ↪person who manages all the details.'
}

personality = 0
for match, shoe in enumerate(result[0]):
    if match:
        personality = shoe
        break

print(shoe_personality_table[personality])
```

```
The person is someone who's very confident, very goal-oriented, and very
organized.
```

1.6 Conclusion

Over the CNN model performs decently well. As shown in the CNN Model Training graph, the accuracy and loss were diverging from the validation accuracy and validation loss. This suggests that the model was over-fitting the data. Despite this, the CNN model correctly classified the single image test shoe (my shoe).

To improve the CNN model's accuracy, a larger and more diverse dataset should be used. Due to the time constraints of the exam, I was unable to do this.