Project: Sneaker Description Generator

Student Name: Victor Kwok

Purpose

The goal of this product is to generate a product description with just an image of a sneaker.

The technical approach

The project is structured in a CNN, then GPT-2 fashion. A UI interface was not created for this project, so currently the user must manually input the path of the image to the desired image. When the user passes the image into the application, the image would first go through the 2 different CNNs. One for Style classification, and one for color classification. The Style classification CNN classifies what type of sneaker the image is. Output could vary from Air Jordan 1, Dunk Low to Air Max 1. However, due to incomplete datasets from the website, the dataset does not have all the style that is owned by Nike. The Color classification CNN classifies what the primary color of the sneaker is. Producing output such as Red, blue, white, black, orange, etc. However similar to the style issue, the range of colors is limited to the colors only found on the website.

The fine-tuned GPT-2 is trained with a question-and-answer custom dataset, where the prompt is a sentence formatted in this way: "[Q] create a product description for {Sneaker Name}, style: {Style}, colourway: {Colour}" and the Answer would be the corresponding production description. The output produced by the two different CNNs will be then reformatted into an input for the GPT-2. For example, if the two CNNs produce the keywords: Air Max 1 and Orange, then the keywords would be reformatted into a prompt: "[Q] create a product description for Air Max 1, style: Air Max 1, colourway: Orange" The fine-tuned GPT-2 would then generate an appropriate sequence of sentences based on the input.

Implementation

CNN:

Both CNNs are trained in the same way using TensorFlow. Both CNN has 3 main activation function layer with the first two being Relu and the last layer being SoftMax. For both CNN the batch size is set to 125 and the epochs are set to 10.

```
# Set random seed
tf.random.set_seed(42)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPool2D
from tensorflow.keras.optimizers import Adam

# Create model
model = Sequential([
    Conv2D(10, 3, activation="relu", input_shape=(240, 240, 3)),
    MaxPool2D(pool_size=2),
    Conv2D(10, 3, activation="relu"),
    MaxPool2D(pool_size=2),
    Flatten(),
    Dense(19, activation="softmax"),

])

# Compile the model
model.compile(loss="categorical_crossentropy",
              optimizer=Adam(),
              metrics=["accuracy"])
```

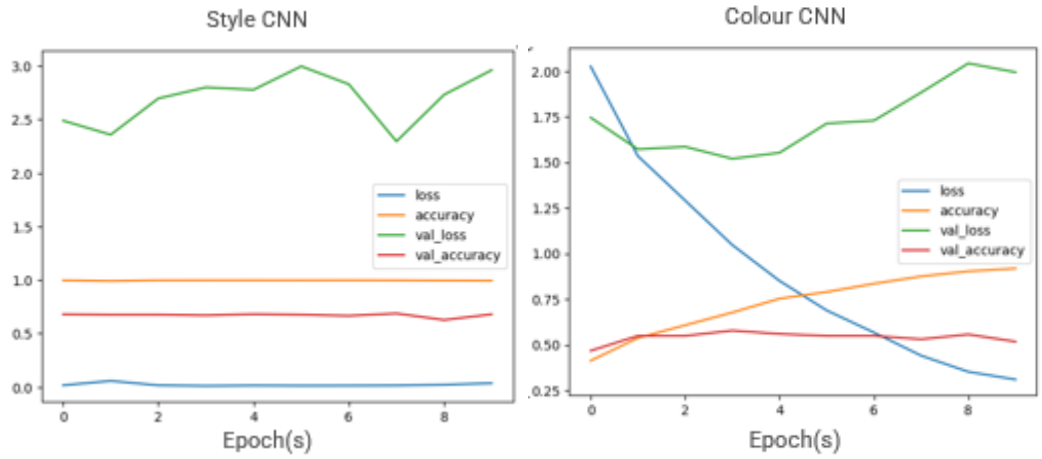
```
# Fit the model
history = model.fit(train_data,
                    batch_size=125,
                    epochs=10,
                    steps_per_epoch=len(train_data),
                    validation_data=validation_data,
                    validation_steps=len(validation_data))
```

FineTuned-GPT2:

Use the GPT2Tokenizer, and GPT2LMHeadModel from the transformers to access the GPT2, and use the trainer from the transformers as well as the custom dataset created by reformatting the data that was web scraped from the website to finetune the GPT2. Then use the save model function to save the model.

Results-

CNNs



The graphs show the validation loss, validation accuracy, train loss, and train accuracy. For both CNNs, the training accuracy ended up being extremely high. For the style CNN, the training accuracy scored a near-perfect accuracy as can be seen in the graph on the left, with the training accuracy being on the borderline of 1.0, which symbolized 100% accuracy, throughout all 10 Epochs. This is abnormally high. For Color CNN, the accuracy started off fairly low with around 0.4 training accuracy in its first epoch. However, during its 10^{th} epoch, the training accuracy reached an abnormal level similar to the style CNN, where the training accuracy is around 0.96, indicating a 96% accuracy.

Both validation accuracies were subpar, around 55 to 65% accuracy. The extremely high and mediocre difference might indicate an overfitting within both models. This is further supported by the outcome of the testing data.

Style CNN Test loss and accuracy:

```
29/29 [==============================] - 4s 138ms/step - loss: 2.3266 - accuracy: 0.6761

[2.3266329765319824, 0.676086962223053]
```

Color CNN Test loss and accuracy:

```
24/24 [==============================] - 5s 220ms/step - loss: 2.1899 - accuracy: 0.4993

[2.1899476051330566, 0.49932339787483215]
```

Both CNNs' Test accuracy was similar to their respective validation accuracy and nowhere near the >90% accuracy that the test accuracy produced

FineTuned-GPT2:

```
[Q] create a product description for Nike Dunk, style: Nike Dunk, colourway: Black
[A] The Nike Dunk Low Retro 'Move To Zero' (DZ0394-101) is a collaboration between streetw
```

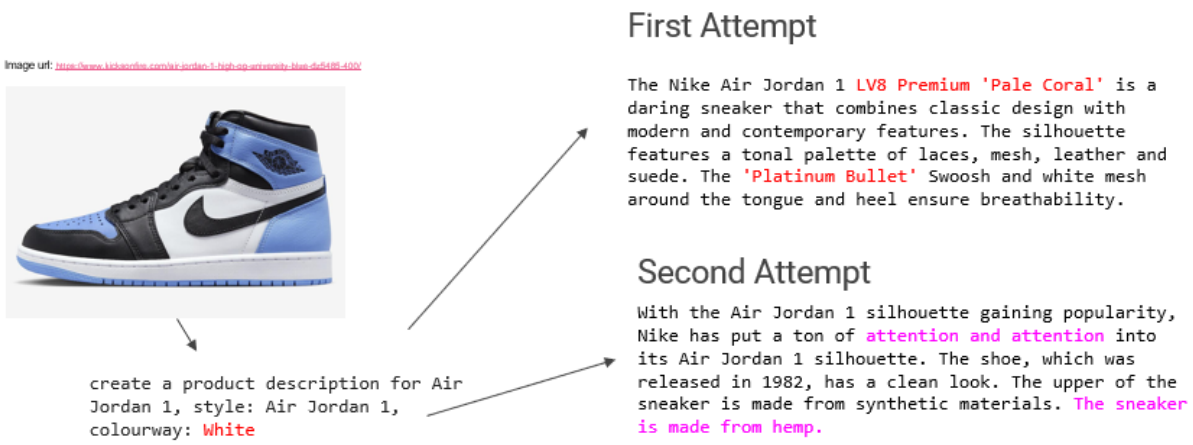
```
[Q] create a product description for Air Max 270, style: Air Max 270, colourway: black
[A] The Nike Air Max 270, formerly known as the Air Max 270, has been released in two new colours. The
```

```
[Q] create a product description for Air Jordan 3, style: Air Jordan 3, colourway: orange
[A] The Air Jordan 3 – Next Nature (DZ2323-001) is one of the most anticipated sneaker release
```

For the FineTuned-GPT2, manual testing was conducted. Random Prompts were manually typed in as input to see what the FineTuned-GPT2 was able to generate. The results were surprisingly better than expected due to the extreme amount of missing product descriptions found on the website that was web scraped from. In all cases within the manual testing, the FineTuned-GPT2 was able to successfully generate the correct style, and some broad yet related descriptions of it, however, the performance on generating the correct color description of the sneaker did not perform as well, as the description generate the wrong color from time to time. In addition to that, random add-ons to the name such as the style number also appear during some of the manual testing.

Overall:

Manually testing the application as a whole leads to the following result:



For styles, it has decent accuracy with popular styles such as Jordan 1, Dunk Low, Air Max 90. and low accuracy on less popular styles. However, for Color, it just had low accuracy overall. As for the ending product description, similar results as the stand-alone FineTuned-GPT2 testing was found. There were random add-ons to the name and some small sentence error but overall decent.

Limitations-

All the images that were web scraped were images of the side view of the sneaker facing left, therefore any images that are not faced the same way would affect the performance of the application.

Another limitation previously mentioned is that there are Missing styles that are less popular or were not found in the website that was web scraped, therefore a sneaker that has one of the missing styles would result in a wrong output. In addition to that the dataset is unequal, Some styles or colours have way more images than others. E.g. White has around 300 images while the Navy only has around 20.

How to Run it:

Change the “Testing2.jpg” to the location of your desired image, then press run all.