APPAREL CLASSIFICATION PROJECT

Abstract

To see the application of keyword extraction and classify the given apparel classes successfully using Computer Vision techniques and applications. This project will help us see how search engine can work using predictive modelling to display the correct results to the user.

Introduction

Taking the example of Google Images, when you search for anything, it returns a number of correct results depending on your search and those images shown are displayed using numerous concepts. One such concept can be called “keyword extraction”. The term “keyword extraction” refers to the resulting output (in this case the images) to be matching to the given keyword input. Using this input, Google Images gives us the resulting output. Similarly, this concept can especially be seen in an app called Pinterest. Pinterest is an image-based social media network where users and businesses can promote and explore their interests by pinning images and videos on virtual boards that usually have a common theme. Pinterest differs from many other social media platforms in that it is image-based, allowing a following to develop among those with interests that have a strong visual component. Pinterest encourages its users to credit the sources for their images and provide a link back to them. Categorization is key. Pinterest is primarily used for commerce, most pins on the platform represent an idea, product, or service that a consumer wants to try or buy.

So, the idea of classification and keyword extraction is widely used in image related websites and companies.

Using a dataset from Kaggle, we will try to correctly predict the classes using Computer Vision techniques and applications.

Dataset Details

Link: https://www.kaggle.com/datasets/trolukovich/apparel-images-dataset

Context: This dataset has been created to practice multi-label classification.

Content: The dataset consists of 5993 images and includes next categories:

- black_dress: 450
- black_pants: 871
- black_shirt: 715
- black_shoes: 766
- black_shorts: 328
- blue_dress: 502
- blue_pants: 798
- blue_shirt: 741
- blue_shoes: 523
- blue_shorts: 299

Methodology and Process

The method that we will be following for this multi-class classification problem is as follows:

1) We will read different folders as different variables in our python file.

```
folder1 = 'black_dress'
folder2 = 'black_pants'
folder3 = 'black_shirt'
folder4 = 'black_shoes'
folder5 = 'black_shorts'
folder6 = 'blue_dress'
folder7 = 'blue_pants'
folder8 = 'blue_shirt'
folder9 = 'blue_shoes'
folder10 = 'blue_shorts'
```

2) Using the cv2 and os library, we will iterate through the all the image folders and read all the images in an empty list. Sample code for one folder is:

```
import os
import cv2
black_dress = []
for filename in os.listdir(folder1):
  imgg = plt.imread(os.path.join(folder1,filename))
  resized_img = cv2.resize(imgg , (350,350))
  black_dress.append(resized_img)
```

3) Then we will convert these lists to numpy arrays and check their shape. For example:

```
black_dress = np.array(black_dress)
np.shape(black_dress)
```

```
(450, 350, 350, 3)
```

4) Then we will display sample images from each folder to make sure our images have been read properly. For example, black_shirt folder will have images like:

5) Then we will create our X and y variables to be used to fit our model on by concatenating all the image lists to X variable and creating a y variable by generating an array of numbers for each image in each folder and then combing them.

```
x = np.concatenate([black_dress, black_pants, black_shirt, black_shoes, black_shorts, blue_dress, blue_pants, blue_shirt, blue_sh
```

```
x.shape
```

```
(5993, 350, 350, 3)
```

```
y1 = np.zeros(450)
y2 = np.ones(871)
y3 = np.full(shape=(715), fill_value = 2)
y4 = np.full(shape=(766), fill_value = 3)
y5 = np.full(shape=(328), fill_value = 4)
y6 = np.full(shape=(502), fill_value = 5)
y7 = np.full(shape=(798), fill_value = 6)
y8 = np.full(shape=(741), fill_value = 7)
y9 = np.full(shape=(523), fill_value = 8)
y10 = np.full(shape=(299), fill_value = 9)
y = np.concatenate([y1 , y2, y3, y4, y5, y6, y7, y8, y9, y10])
y = y.reshape(-1 , 1)
```

```
y.shape
```

```
(5993, 1)
```

6) We will convert our y variable to categorical to get our target variable.

```
from tensorflow.keras.utils import to_categorical
y = to_categorical(y)
```

```
y.shape
```

```
(5993, 10)
```

7) Now, we will create a sequential model. The sequence will contain of two Conv2D layers and then one MaxPooling2D layer. After testing a number of activation functions, we came to the conclusion that relu works the best as it solves the vanishing gradient problem. We have kept the padding same in order to prevent compression of images. After this we Flatten the images and apply a Dense layer with the activation function relu and in the final Dense layer, we use softmax activation function to generate probabilities. This concludes our CNN model.

```
from keras.layers import Dense, Flatten , Conv2D , MaxPooling2D
from keras.models import Sequential
```

```
model = Sequential()

model.add(Conv2D(64, (3, 3), activation='relu', padding='same',input_shape=(350 , 350 , 3)))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

8) We use the Adam optimizer as it helps with momentum and adaptive learning rate. We use the loss function Categorical Cross Entropy because this is a multi-class classification problem and this loss function will help us optimize our weights better. The metric we use for evaluating our model is accuracy. The model gives us approximately 98% accuracy on running 5 epochs.

```
model.compile(optimizer = 'Adam' , loss = 'CategoricalCrossentropy' , metrics = ['accuracy'])
```

```
model.fit(x , y , epochs = 5)
```

```
Epoch 1/5
188/188 [==============================] - 2616s 14s/step - loss: 1.3068 - accuracy: 0.5633
Epoch 2/5
188/188 [==============================] - 2773s 15s/step - loss: 0.4834 - accuracy: 0.8447
Epoch 3/5
188/188 [==============================] - 2651s 14s/step - loss: 0.1843 - accuracy: 0.9488
Epoch 4/5
188/188 [==============================] - 2488s 13s/step - loss: 0.0992 - accuracy: 0.9781
Epoch 5/5
188/188 [==============================] - 2164s 12s/step - loss: 0.0713 - accuracy: 0.9808

<keras.callbacks.History at 0x22f81d619d0>
```

```
model.evaluate(x, y)
```

```
188/188 [==============================] - 362s 2s/step - loss: 0.0311 - accuracy: 0.9918
```

Analysis

On analysing our dataset, we can clearly see that it is an imbalanced dataset as some folders have a greater number of images than others and this difference is significant with black_pants having 871 images and blue_shorts having 299 images. Because of this imbalanced dataset, it leads to an overfitting condition and some images are predicted correctly but on further testing we can see that our model doesn't work well with unknown images from the internet and classifies incorrectly. We will see this outcome when we see our results ahead.

Results

1) It is observed that using the argmax function when we test our model on the known images from the dataset, it returns the correct output and classifies the image correctly according to its apparel class.

```
x[0].shape
```

```
(350, 350, 3)
```

```
test = x[1800]
test = test.reshape(1, 350, 350, 3)
```

```
np.argmax(model.predict(test), axis=-1)
```

```
array([2], dtype=int64)
```

```
test1 = x[0]
test1 = test1.reshape(1, 350, 350, 3)
```

```
np.argmax(model.predict(test1), axis=-1)
```

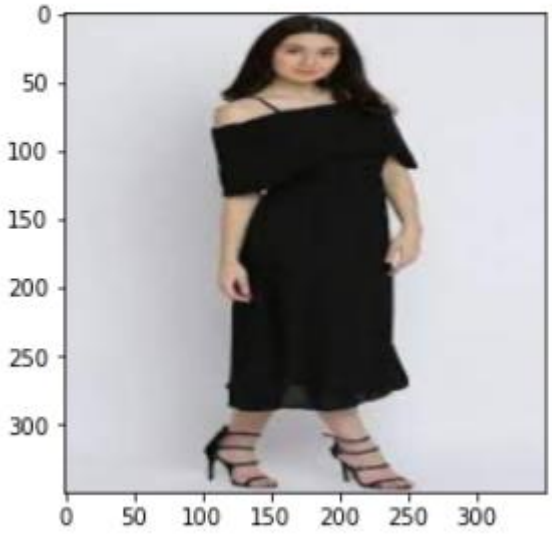
```
array([0], dtype=int64)
```

2) As you can see in the above image, it classifies and returning the highest probability array index. We can conclude that our model is accurately predicting the classes on our training set. Black shirt is being classified as black shirt and black dress is being classifies as black dress.

3) On further testing of our model, we grabbed some images from the internet and read those images again using cv2.imread function. Upon testing whether our model is predicting these test images correctly, we observed that the classes with a greater number of training images are predicted correctly. For example:

```
plt.imshow(resized_img)
```

```
<matplotlib.image.AxesImage at 0x22f81edfa90>
```



Black dress test image

```
testimgg = plt.imread(os.path.join('testimage.jpg'))
resized_img = cv2.resize(testimgg , (350,350))
```

```
testimage = np.array(resized_img)
```

```
testimage.shape
```

```
(350, 350, 3)
```

```
testimage = testimage.reshape(1,350,350, 3)
```

```
np.argmax(model.predict(testimage), axis=-1)
```

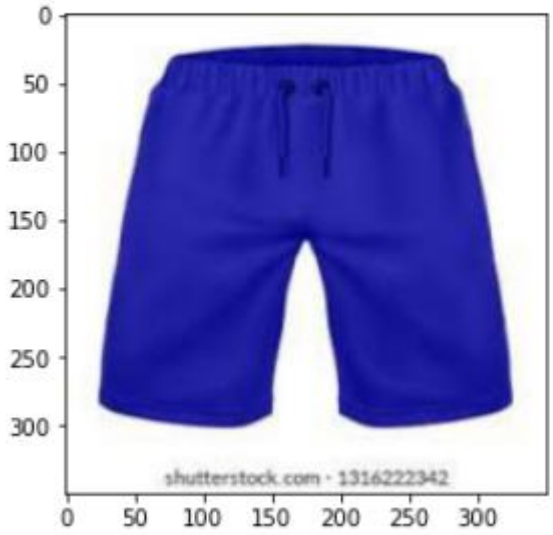
```
array([0], dtype=int64)
```

Black dress test image correct prediction

4) But, testing with test images which should belong to a folder with a smaller number of images, we saw that our model was predicting incorrectly.

```
plt.imshow(resized_img2)
```

```
<matplotlib.image.AxesImage at 0x22f81f33b20>
```



Blue shorts test image

```
testimg = plt.imread(os.path.join('testimage2.jpg'))
resized_img2 = cv2.resize(testimg , (350,350))
```

```
testimage2 = np.array(resized_img2)
```

```
testimage2.shape
```

```
(350, 350, 3)
```

```
testimage2 = testimage2.reshape(1,350,350, 3)
```

```
np.argmax(model.predict(testimage2), axis=-1)
```

```
array([7], dtype=int64)
```

Blue shorts test image incorrect prediction

5) It is observed that our model thinks that this test image of a blue short belongs to the class of blue shirt. Hence, we can conclude that there is a case of overfitting.

References

https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/#:~:text=Max%20pooling%20is%20a%20pooling,of%20the%20previous%20feature%20map.

https://keras.io/api/layers/convolution_layers/convolution2d/

https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5#:~:text=Sequential%20is%20the%20easiest%20way,2%20layers%20are%20Conv2D%20layers.

https://www.tutorialspoint.com/keras/keras_flatten_layers.htm

https://www.tensorflow.org/tutorials/images/classification

https://monkeylearn.com/keyword-extraction/

https://searchengineland.com/guide/what-is-seo