FLIP ROBO

IMAGE SCRAPING AND CLASSIFICATION

SUBMITTED BY

TUSHAR KUMAR PATEL

2021

ACKNOWLEDGEMENT

I would like to express my special thanks to all my mentors of DataTrained and to SME (Sapna Verma) under whom I am an intern at Fliprobo. They all have taught me Machine learning because of that knowledge they have provided has helped me to be able to complete this project. In this whole process some popular websites like google, geeksforgeeks, tutorialspoint etc. was also very helpful for completing the project. For creating the dataset Amazon was the website from where all the data has been scraped using the automation tool selenium. At last github was the location where the final project was uploaded for the submission.

INTRODUCTION

• Business problem framing

Classification is a systematic arrangement in groups and categories based on its features. Image classification came into existence for decreasing the gap between the computer vision and human vision by training the computer with the data. The image classification is achieved by differentiating the image into the prescribed category based on the content of the vision. We use Deep Learning to accomplish the task of image Classification.

In this project we have to classify whether is image is of a Jeans, a trouser or a saree. We could we such model either for automatic segmentation of items using IOT and techniques for Industry.

• Conceptual Background of the Domain Problem

Deep learning (DL) is a sub field to the machine learning, capable of learning through its own method of computing. A deep learning model is introduced to persistently break down information with a homogeneous structure like how a human would make determinations. To accomplish this, deep learning utilizes a layered structure of several algorithms expressed as an artificial neural system (ANN). The architecture of an ANN is simulated with the help of the biological neural network of the human brain. This makes the deep learning most capable than the standard machine learning models.

In deep neural networks every node decides its basic inputs by itself and sends it to the next tier on behalf of the previous tier. We train the data in the networks by giving an input image and conveying the network about its output. Neural networks are expressed in terms of number of layers involved for producing the inputs and outputs and the depth of the neural network.

• Motivation for the Problem Undertaken

The project was the first provided to me by FlipRobo as a part of the internship program. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. Every problem begins with ideas that are further developed and inspired to address a variety of situations and circumstances. Learning the theoretical background for data science or machine learning are often a frightening experience, because it involves multiple fields of mathematics and an extended list of online resources. By proper practical research and practice I can become better in this field. These suggestions are derived from my mentors/SME’s and my own experience in the beginner projects.

Analytical Problem Framing

- **Data Sources and their formats**

  The data was scraped using the automation tool named Selenium and beautifulSoup with the help of programing language Python. Which was saved into .jpeg format for the further analysis for the classification of the images. After loading the training dataset into Jupyter Notebook.
- Data is collected through web scraping of an E-commerce website (Amazon). Three category of clothes are scrapped namely Sarees (women), Trouser (men) and Jeans (men).

- **Data Pre-processing Done**
  The scrapped data is splitted into train and text and as input and output using folder split.

- **Hardware and Software Requirements and Tools Used**
  - Hardware: 8GB RAM, 64-bit, 9th gen i7 processor.
  - Software: MS-Excel, Jupyter Notebook, python 3.6.

- **Libraries used:-**

  Pandas, Numpy, Tensorflow, Keras, warnings etc.

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods).**
  I have scrapped the data from Amazon and then cleaning the data to be further used for training the model.
  Then splitting the data into train and test set to model evaluation.

- **Testing of Identified Approaches (Algorithms)**

Inception model is used and Imagenet as weights, optimizer RMSprop, convolution 2D filters, loss as categorical_crossentropy and accuracy as metrics.

- **Run and Evaluated selected models**

In [8]:
```
# Training Data Generator( Data Augmentation on Training Images)

Train_generator_augmented = ImageDataGenerator(
    rescale=1.0 / 255, zoom_range=0.2, rotation_range=30, horizontal_flip=True
)
Train_generator = Train_generator_augmented.flow_from_directory(
    Train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode="categorical",
)

# Validation Data Generator
Data_gen = ImageDataGenerator(rescale=1.0 / 255)
validation_generator = Data_gen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode="categorical",
)
```

```
Found 833 images belonging to 3 classes.
Found 75 images belonging to 3 classes.
```

In [9]:
```
# Creating the model
model = Sequential()

# First convolution layer
model.add(Conv2D(32, (3, 3), input_shape=input_shape))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Second convolution layer
model.add(Conv2D(32, (3, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Third convolution layer
model.add(Conv2D(64, (3, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Fourth convolution layer
model.add(Conv2D(64, (3, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))


model.add(Flatten())
model.add(Dense(128))
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(3))
model.add(Activation("softmax"))

print(model.summary())

model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 574, 574, 32)      896
```

```
activation (Activation)     (None, 574, 574, 32)      0

max_pooling2d (MaxPooling2D  (None, 287, 287, 32)     0
)

dropout (Dropout)           (None, 287, 287, 32)      0

conv2d_1 (Conv2D)           (None, 285, 285, 32)      9248

activation_1 (Activation)   (None, 285, 285, 32)      0

max_pooling2d_1 (MaxPooling  (None, 142, 142, 32)     0
2D)

dropout_1 (Dropout)         (None, 142, 142, 32)      0

conv2d_2 (Conv2D)           (None, 140, 140, 64)      18496

activation_2 (Activation)   (None, 140, 140, 64)      0

max_pooling2d_2 (MaxPooling  (None, 70, 70, 64)       0
2D)

dropout_2 (Dropout)         (None, 70, 70, 64)        0

conv2d_3 (Conv2D)           (None, 68, 68, 64)        36928

activation_3 (Activation)   (None, 68, 68, 64)        0

max_pooling2d_3 (MaxPooling  (None, 34, 34, 64)       0
2D)

dropout_3 (Dropout)         (None, 34, 34, 64)        0

flatten (Flatten)           (None, 73984)             0

dense (Dense)               (None, 128)               9470080


activation_4 (Activation)   (None, 128)               0

dropout_4 (Dropout)         (None, 128)               0

dense_1 (Dense)             (None, 3)                 387

activation_5 (Activation)   (None, 3)                 0

=================================================================
Total params: 9,536,035
Trainable params: 9,536,035
Non-trainable params: 0

_________________________________________________________________
None
```

- **Key Metrics for success in solving problem under consideration**

In [11]:
```
# Fitting the Training Data
history = model.fit_generator(
    Train_generator,
    epochs=epoch,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size,
    steps_per_epoch=nb_train_samples // batch_size,
    callbacks=[ES, MC],
)
```

```
Epoch 1/150
22/22 [==============================] - ETA: 0s - loss: 2.0390 - accuracy: 0.3254
Epoch 00001: val_accuracy improved from -inf to 0.25000, saving model to best.h5
22/22 [==============================] - 65s 3s/step - loss: 2.0390 - accuracy: 0.3254 - val_loss: 1.0994 - val_accuracy: 0.250
0
Epoch 2/150
22/22 [==============================] - ETA: 0s - loss: 1.0936 - accuracy: 0.4260
Epoch 00002: val_accuracy improved from 0.25000 to 0.37500, saving model to best.h5
22/22 [==============================] - 64s 3s/step - loss: 1.0936 - accuracy: 0.4260 - val_loss: 1.0974 - val_accuracy: 0.375
0
Epoch 3/150
22/22 [==============================] - ETA: 0s - loss: 1.1040 - accuracy: 0.3295
Epoch 00003: val_accuracy improved from 0.37500 to 0.56250, saving model to best.h5
22/22 [==============================] - 66s 3s/step - loss: 1.1040 - accuracy: 0.3295 - val_loss: 1.0895 - val_accuracy: 0.562
5
Epoch 4/150
22/22 [==============================] - ETA: 0s - loss: 1.0977 - accuracy: 0.3432
Epoch 00004: val_accuracy did not improve from 0.56250
22/22 [==============================] - 67s 3s/step - loss: 1.0977 - accuracy: 0.3432 - val_loss: 1.0978 - val_accuracy: 0.312
5
Epoch 5/150
22/22 [==============================] - ETA: 0s - loss: 1.0864 - accuracy: 0.3523
Epoch 00005: val_accuracy did not improve from 0.56250
22/22 [==============================] - 65s 3s/step - loss: 1.0864 - accuracy: 0.3523 - val_loss: 1.0767 - val_accuracy: 0.250
0
Epoch 6/150
22/22 [==============================] - ETA: 0s - loss: 1.1151 - accuracy: 0.5503
Epoch 00006: val_accuracy did not improve from 0.56250
22/22 [==============================] - 64s 3s/step - loss: 1.1151 - accuracy: 0.5503 - val_loss: 1.0902 - val_accuracy: 0.250
0
Epoch 7/150
22/22 [==============================] - ETA: 0s - loss: 1.0962 - accuracy: 0.3523
Epoch 00007: val_accuracy did not improve from 0.56250
22/22 [==============================] - 68s 3s/step - loss: 1.0962 - accuracy: 0.3523 - val_loss: 1.0830 - val_accuracy: 0.500
0
```

```
Epoch 00115: val_accuracy did not improve from 1.00000
22/22 [==============================] - 72s 3s/step - loss: 0.4086 - accuracy: 0.8182 - val_loss: 0.4149 - val_accuracy: 0.812
5
Epoch 116/150
22/22 [==============================] - ETA: 0s - loss: 0.3023 - accuracy: 0.8580
Epoch 00116: val_accuracy did not improve from 1.00000
22/22 [==============================] - 80s 4s/step - loss: 0.3023 - accuracy: 0.8580 - val_loss: 0.6023 - val_accuracy: 0.875
0
Epoch 117/150
22/22 [==============================] - ETA: 0s - loss: 0.2959 - accuracy: 0.8693
Epoch 00117: val_accuracy did not improve from 1.00000
22/22 [==============================] - 72s 3s/step - loss: 0.2959 - accuracy: 0.8693 - val_loss: 0.8728 - val_accuracy: 0.750
0
Epoch 118/150
22/22 [==============================] - ETA: 0s - loss: 0.3690 - accuracy: 0.8295
Epoch 00118: val_accuracy did not improve from 1.00000
22/22 [==============================] - 70s 3s/step - loss: 0.3690 - accuracy: 0.8295 - val_loss: 0.3534 - val_accuracy: 0.875
0
Epoch 119/150
22/22 [==============================] - ETA: 0s - loss: 0.4092 - accuracy: 0.8047
Epoch 00119: val_accuracy did not improve from 1.00000
22/22 [==============================] - 66s 3s/step - loss: 0.4092 - accuracy: 0.8047 - val_loss: 0.5250 - val_accuracy: 0.812
5
Epoch 120/150
22/22 [==============================] - ETA: 0s - loss: 0.3196 - accuracy: 0.8125
Epoch 00120: val_accuracy did not improve from 1.00000
22/22 [==============================] - 71s 3s/step - loss: 0.3196 - accuracy: 0.8125 - val_loss: 0.3325 - val_accuracy: 0.937
5
Epoch 121/150
22/22 [==============================] - ETA: 0s - loss: 0.3145 - accuracy: 0.8636
Epoch 00121: val_accuracy did not improve from 1.00000
22/22 [==============================] - 69s 3s/step - loss: 0.3145 - accuracy: 0.8636 - val_loss: 0.3268 - val_accuracy: 0.875
0
Epoch 122/150
22/22 [==============================] - ETA: 0s - loss: 0.3122 - accuracy: 0.8580
Epoch 00122: val_accuracy did not improve from 1.00000
22/22 [==============================] - 68s 3s/step - loss: 0.3122 - accuracy: 0.8580 - val_loss: 0.2594 - val_accuracy: 0.875
0
Epoch 00122: early stopping
```

In [20]:
```
# Model Evaluation
evl = model.evaluate(validation_generator, steps=1)
print("Test Loss", evl[0])
print("Test Accuracy", evl[1])
```

```
1/1 [==============================] - 1s 583ms/step - loss: 0.3177 - accuracy: 0.8750
Test Loss 0.3177236020565033
Test Accuracy 0.875
```

- Key Metrices used was the Accuracy Score and the prediction is also good. From the above we can see that the model gives a good accuracy score as more than 87.5%.

- ## Interpretation of the Results

➢ Given imagenet as weight for the inception model in layers.

CONCLUSION

- ## Learning Outcomes of the Study in respect of Data Science

a. By using selenium scrapping of images from Amazon is easy.

b. By using optimizers, layers and filters observed that there is difference in scoring.

- ## Limitations of this work and Scope for Future Work

One could always provide more data for training the model in order to get better results. We could use to model for apparel segmentation at Supermarkets and shopping stores.