FLIP ROBO

Image Scraping and Classification

Submitted by:

Ashish Modi

Acknowledgement

I would like to express my gratitude to my guide Shubham Yadav (SME, Flip Robo) for his constant guidance, continuous encouragement and unconditional help towards the development of the project. It was he who helped me whenever I got stuck somewhere in between. The project would have not been completed without his support and confidence he showed towards me.

Lastly, I would l like to thank all those who helped me directly or indirectly toward the successful completion of the project.

Ashish Modi

Introduction

Business Problem Framing

The image classification is a classical problem of image processing, computer vision and machine learning fields. Images are one of the major sources of data in the field of data science and AI. This field is making appropriate use of information that can be gathered through images by examining its features and details. We are trying to give you an exposure of how an end-to-end project is developed in this field.

The idea behind this project is to build a deep learning-based Image Classification model on images that will be scraped from ecommerce portal. This is done to make the model more and more robust.

This task is divided into two phases: Data Collection and Model Building.

We need to scrape images from e-commerce portal, Amazon.com. The clothing categories used for scraping will be:

- Sarees (women)
- Trousers (men)
- Jeans (men)

We need to scrape images of these 3 categories and build our data from it. That data will be provided as an input to your deep learning problem. We need to scrape minimum 200 images of each category. There is no maximum limit to the data collection. We are free to apply image augmentation techniques to increase the size of your data but make sure the quality of data is not compromised.

Conceptual Background of the Domain Problem

Image classification has become a major challenge in machine vision and has a long history with it. The challenge includes a broad intra-class range of images caused by colour, size, environmental conditions and shape. It is required big data of labelled training images and to prepare this big data, it consumes a lot of time and cost as for the training purpose only. In this project, deep neural network, based on TensorFlow is used with Python as the programming language for image classification. Hundreds of images are used as the input data in this project. The accuracy of each percentage of 'train' session will be studied and compared.

Review of Literature

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 explore the study of image classification using deep learning. The conventional methods used for image classifying is part and piece of the field of artificial intelligence (AI) formally called as machine learning. The machine learning consists of feature extraction module that extracts the important features such as edges, textures etc and a classification module that classify based on the features extracted. The main limitation of machine learning is, while separating, it can only extract certain set of features on images and unable to extract differentiating features from the training set of data. This disadvantage is rectified by using the deep learning. 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.

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. The main motivation is to build a prototype of image classification which can used to classify Saree, Jeans and Trouser so that it can be identify what is what.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

We would perform one type of supervised learning algorithms: Classification. While it seems more reasonable to perform Classification since we have 3 class to predict. Here, we will only perform classification. Since there only 1 feature in the dataset.
First, we label the images in our dataset according to their categories as 0 for jeans, 1 for sarees and 2 for trousers.
The images in our dataset are of shape 320, 137, 3 we resize it to the shape 180, 180, 3.
Then we scale images by dividing them with 255 before feeding it to our neural network.

Data Sources and their formats

We have 1 input field i.e. image and 3 output classes.
Image field is type image.
3 other classifications are in 0,1 and 2 forms.

Data Pre-processing Done

In the pre-processing part we resize the images present in our dataset from 320, 137, 3 to 240,240, 3.
Then we scale the images of both the training and test set before feeding it to the neural network.

Hardware and Software Requirements and Tools Used

The system requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view. System requirements are all of the requirements at the system level that describe the functions which the system as a whole should fulfil to satisfy the stakeholder needs and requirements, and is expressed in an appropriate combination of textual statements, views, and non-functional requirements; the latter expressing the levels of safety, security, reliability, etc., that will be necessary.

Hardware requirements: -

1. Processor — core i5 and above
2. RAM — 8 GB or above
3. SSD — 250GB or above

Software requirements: -

Anaconda

Libraries: -

from tensorflow.keras.models import Model

`Model` groups layers into an object with training and inference features.

from tensorflow.keras.applications.vgg16 import VGG16

Instantiates the VGG16 model.

from tensorflow.keras.applications.vgg16 import preprocess_input

Preprocesses a tensor or Numpy array encoding a batch of images.

from tensorflow.keras.preprocessing import image

Public API for tf.keras.preprocessing.image namespace.

from tensorflow.keras.preprocessing.image import ImageDataGenerator

Generate batches of tensor image data with real-time data augmentation.

from tensorflow.keras.models import Sequential

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

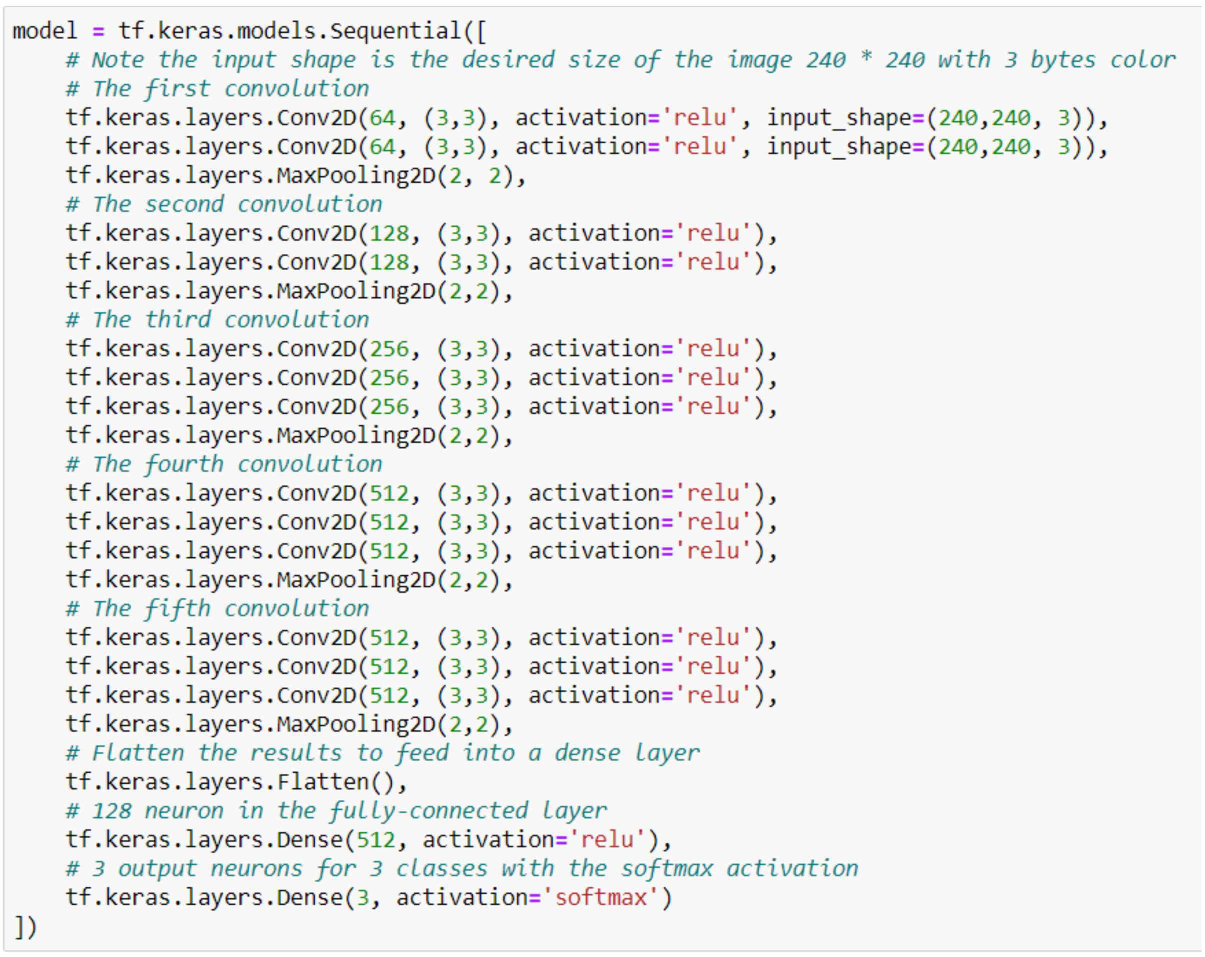
Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

The algorithms we used for the training and testing is convolutional neural network.

We found out that our model was overfitting so with the help of data augmentation techniques we increased the size of our dataset and fed it to the convolutional neural network and the desired accuracy was achieved in our test dataset.

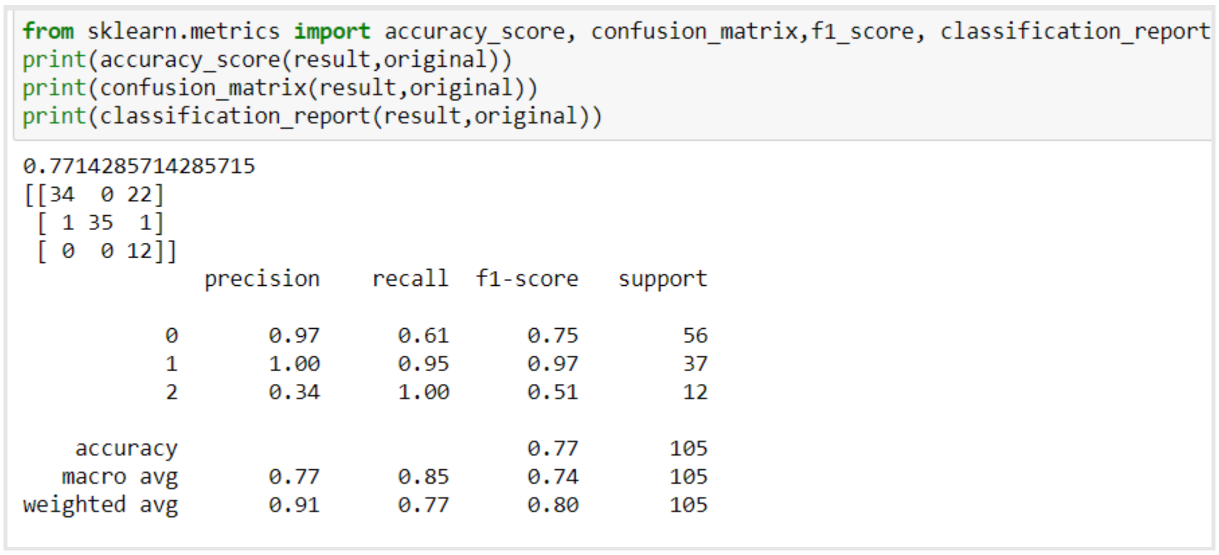
Run and Evaluate selected models



```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 240 * 240 with 3 bytes color
    # The first convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(240,240, 3)),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(240,240, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(256, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(256, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(256, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(512, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(512, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(512, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fifth convolution
    tf.keras.layers.Conv2D(512, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(512, (3,3), activation='relu'),
    tf.keras.layers.Conv2D(512, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a dense layer
    tf.keras.layers.Flatten(),
    # 128 neuron in the fully-connected layer
    tf.keras.layers.Dense(512, activation='relu'),
    # 3 output neurons for 3 classes with the softmax activation
    tf.keras.layers.Dense(3, activation='softmax')
])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 238, 238, 64)      1792
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 236, 236, 64)      36928
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 118, 118, 64)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 116, 116, 128)     73856
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 114, 114, 128)     147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 57, 57, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 55, 55, 256)       295168
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 53, 53, 256)       590080
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 51, 51, 256)       590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 25, 25, 256)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 23, 23, 512)       1180160
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 21, 21, 512)       2359808
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 19, 19, 512)       2359808
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 9, 9, 512)         0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 7, 7, 512)         2359808
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 5, 5, 512)         2359808
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 3, 3, 512)         2359808
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 1, 1, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 512)               262656
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 1539
=================================================================
Total params: 14,978,883
Trainable params: 14,978,883
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 99/100
20/20 [==============================] - 257s 13s/step - loss: 0.5033 -
acc: 0.7710
Epoch 100/100
20/20 [==============================] - 255s 13s/step - loss: 0.5003 -
acc: 0.7677
```



```
from sklearn.metrics import accuracy_score, confusion_matrix,f1_score, classification_report
print(accuracy_score(result,original))
print(confusion_matrix(result,original))
print(classification_report(result,original))
```

```
0.7714285714285715
[[34  0 22]
 [ 1 35  1]
 [ 0  0 12]]
              precision    recall  f1-score   support

           0       0.97      0.61      0.75        56
           1       1.00      0.95      0.97        37
           2       0.34      1.00      0.51        12

    accuracy                           0.77       105
   macro avg       0.77      0.85      0.74       105
weighted avg       0.91      0.77      0.80       105
```

Key Metrics for success in solving problem under consideration

Precision: can be seen as a measure of quality, **higher precision** means that an algorithm returns more relevant results than irrelevant ones.

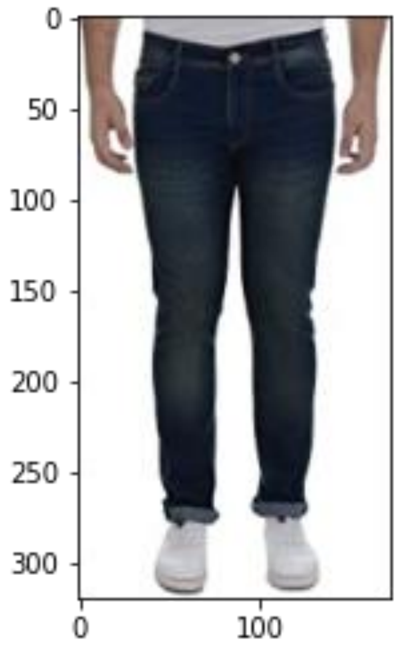
Recall is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

Accuracy score is used when the True Positives and True negatives are more important. **Accuracy** can be used when the class distribution is similar.

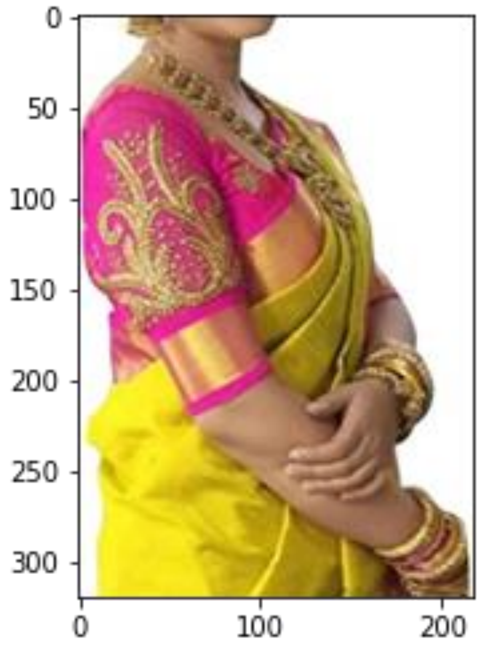
F1-score is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.

Visualizations

Jeans



Saree



Trouser

Interpretation of the Results

On our 100th epoch we got 76.77% accuracy. How ever we can see the f1-score 77%.

Conclusion

Key Findings and Conclusions of the Study

Deep convolution neural networks are used to identify scaling, translation, and other forms of distortion-invariant images. In order to avoid explicit feature extraction, the convolutional network uses feature detection layer to learn from training data implicitly, and because of the weight sharing mechanism, neurons on the same feature mapping surface have the same weight. Weight sharing can greatly reduce the complexity of the network structure. Especially, the multi-dimensional input vector image WDIN can effectively avoid the complexity of data reconstruction in the process of feature extraction and image classification. Deep convolution neural network has incomparable advantages in image feature representation and classification. However, many researchers still regard the deep convolutional neural network as a black box feature extraction model. To explore the connection between each layer of the deep convolutional neural network and the visual nervous system of the human brain, and how to make the deep neural network incremental, as human beings do, to compensate for learning, and to increase understanding of the details of the target object, further research is needed.

Learning Outcomes of the Study in respect of Data Science

The power of visualization is helpful for the understanding of data into the graphical representation its help me to understand that what data is trying to say, Data cleaning is one of the most important steps, tried to make comments shorter and all the necessary keywords in it.

Limitations of this work and Scope for Future Work

Computation time is so high that it took almost 5-6 hours on new i5 10th gen machine with 8Gb Ram. And if we talk about Google collabs, it took almost 7 hours and completed 27 epochs only because of computation high increases while fetching images from google drive. While we couldn't reach out goal of 100% accuracy in image classification, we did end up creating a system that can with enough time and data get very close to that goal. As with any project there is room for improvement here. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project.