AMAZON IMAGES CLASSIFICATION PROJECT

Submitted by:

Nimit Shah

ACKNOWLEDGMENT

The success and final outcome of this project required a lot of guidance and assistance from Sapna Verma ma'am and I am Extremely fortunate to have got this all along the completion of my project work.I am also grateful to Fliprobo Company for assigning this project to me.
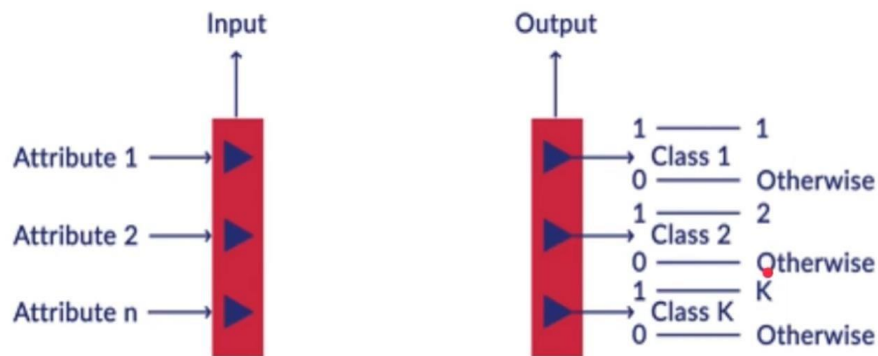
Various references were used like:

- ➢ ImageNet
- ➢ Anlytics Vidhya
- ➢ Medium
- ➢ Data trained Reference materials
- ➢ Github

INTRODUCTION

Business Problem Framing

Image classification of Amazon images Sarees, Trousers and Jeans.

Input data is from the amazon website

The predicted images are from Flipkart website

Conceptual Background of the Domain Problem

Traditional machine learning methods (such as multilayer perception machines, support vector machines, etc.) mostly use shallow structures to deal with a limited number of samples and computing units. When the target objects have rich meanings, the performance and generalization ability of complex classification problems are obviously insufficient. The convolution neural network (CNN) developed in recent years has been widely used in the field of image processing because it is good at dealing with image classification and recognition problems and has brought great improvement in the accuracy of many machine learning tasks. It has become a powerful and universal deep learning model.

Convolutional neural network (CNN) is a multilayer neural network, and it is also the most classical and common deep learning framework. CNNs have broken the mold and ascended the throne to become the state-of-the-art computer vision technique. Among the different types of neural networks (others include recurrent neural networks (RNN), long short term memory (LSTM), artificial neural networks (ANN), etc.), CNNs are easily the most popular.

These convolutional neural network models are ubiquitous in the image data space. They work phenomenally well on computer vision tasks like image classification, object detection, image recognition,

Review of Literature

Image classification being an area of wide interest in the current Data Science Community, various papers have been published in the regard using various datasets. Some of the papers which are widely open and used by the community are:

Trend	Dataset	Best Model	Paper Title	Paper	Code	Compare
	ImageNet	🏆 CoAtNet-7	CoAtNet: Marrying Convolution and Attention for All Data Sizes	📄	◯	See all
	CIFAR-10	🏆 ViT-H/14	An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale	📄	◯	See all
	CIFAR-100	🏆 EffNet-L2 (SAM)	Sharpness-Aware Minimization for Efficiently Improving Generalization	📄	◯	See all
	STL-10	🏆 Wide-ResNet-101 (Spinal FC)	SpinalNet: Deep Neural Network with Gradual Input	📄	◯	See all
	SVHN	🏆 WRN28-10 (SAM)	Sharpness-Aware Minimization for Efficiently Improving Generalization	📄	◯	See all
	MNIST	🏆 Homogeneous ensemble with Simple CNN	An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition	📄	◯	See all
	ImageNet ReaL	🏆 Meta Pseudo Labels (EfficientNet-B6-Wide)	Meta Pseudo Labels	📄	◯	See all
	Flowers-102	🏆 CCT-14/7x2	Escaping the Big Data Paradigm with Compact Transformers	📄	◯	See all

Motivation for the Problem Undertaken

Images are large in number, but understanding if the image is for the use of the task is significant. Various times during building large scale projects we would need to know if the image is of a certain kind or not. Hence Image classification becomes significant.

This particular project deals with segmentation of apparels which can be used in the shopping stores in India, like Bigbazaar which can improve the economy of MSME dealing with apparels. This segmentation can improve the ease of sorting clothes in regions as well as in taking statistics for the particular store with images

Market Analysis (Pie)

- Men (20 to 49 Years)
- Women (20 to 49 Years)
- Boys (5 to 19 Years)
- Girls (5 to 19 Years)
- Infants & Toddlers (0 to 4 Years)
- Internet
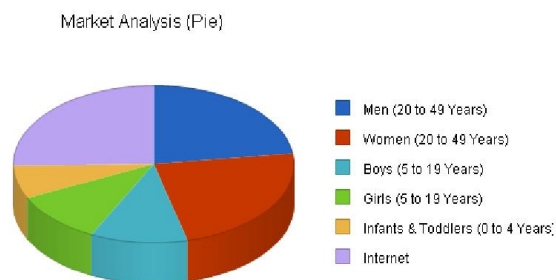
Image classification in a broader spectrum can be used anywhere and anytime in all IOT devices. The particular image can be caught by the camera and model could check if it is the image. If at all it is the image, the IOT device could take the necessary action. For example, an image classification model can detect for an IOT car vehicle if there is a pothole in the road.

Analytical Problem Framing

<u>Mathematical/ Analytical Modeling of the Problem</u>

Vision is arguably our most important sensory system, judging from both the ubiquity of visual forms of communication, and the large proportion of the human brain devoted to visual processing. Nevertheless, it has proven difficult to establish a good mathematical definition (in the form of a statistical model) for visual images.

sssssssss
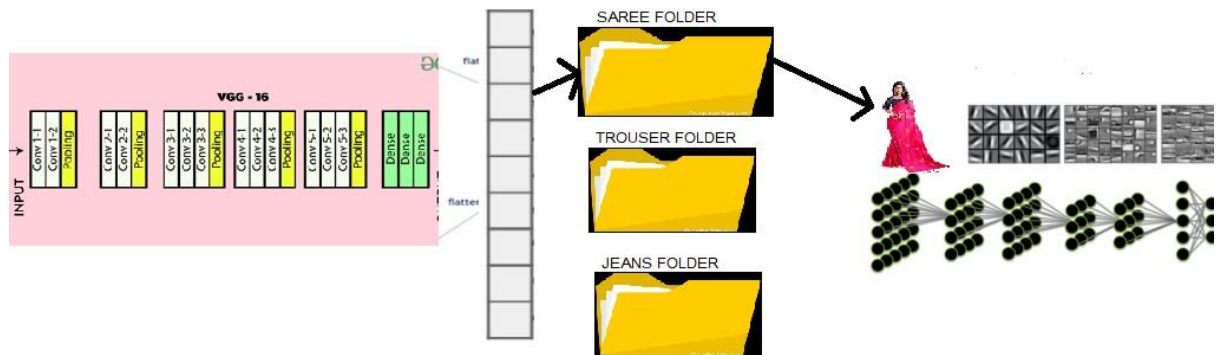
1.Statistical prior models used in this project are the **VGG16,VGG19 and Resnet50**. These are used for transfer learning in the rest of the modelling done. The best accuracy was seen in VGG16 hence it was chosen for final modelling.
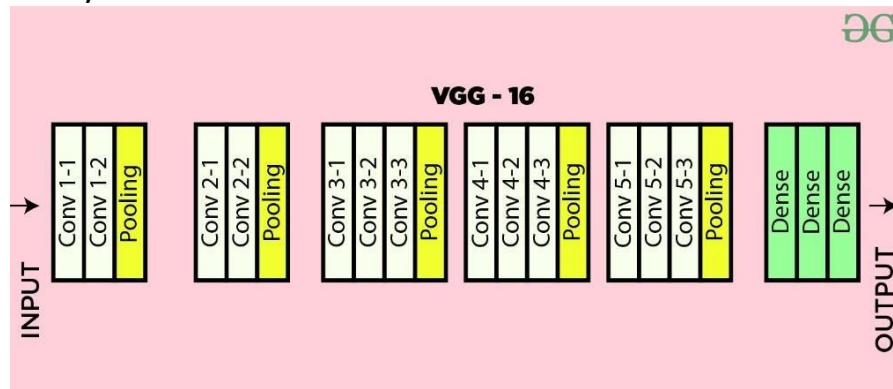
2.Then the input data with the transfer model vgg16 is flattened using mathematical approach.

Flattening is **converting the data into a 1-dimensional array** for inputting it to the next layer.We **flatten the output of the convolutional layers** to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer
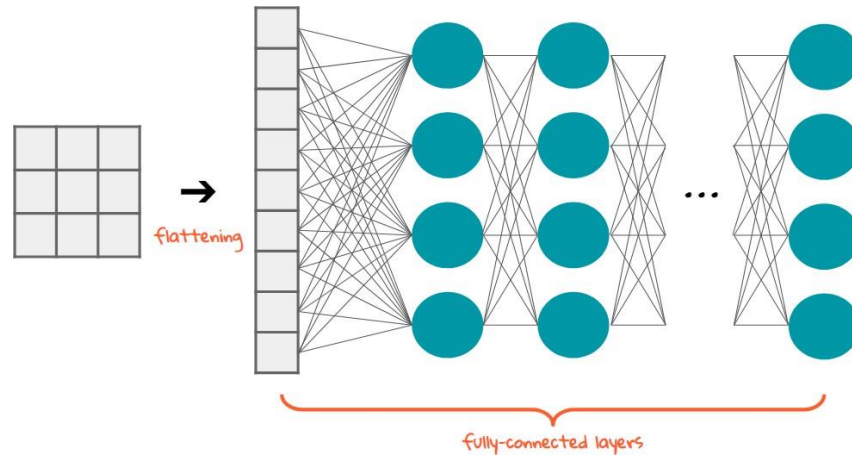
3.Appending folders as **Dense Layer with activation function softmax**. Softmax is used as in multiclass classification best activation function is softmax
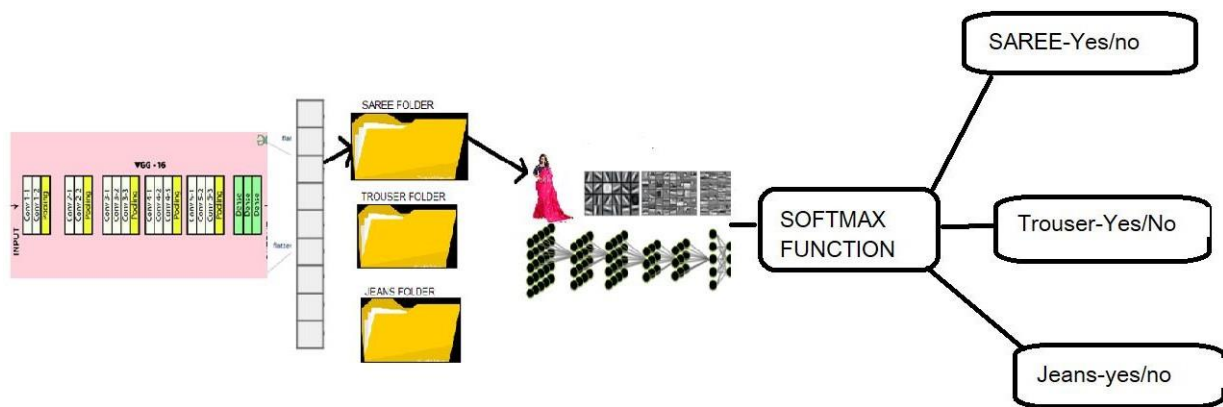
Model structure
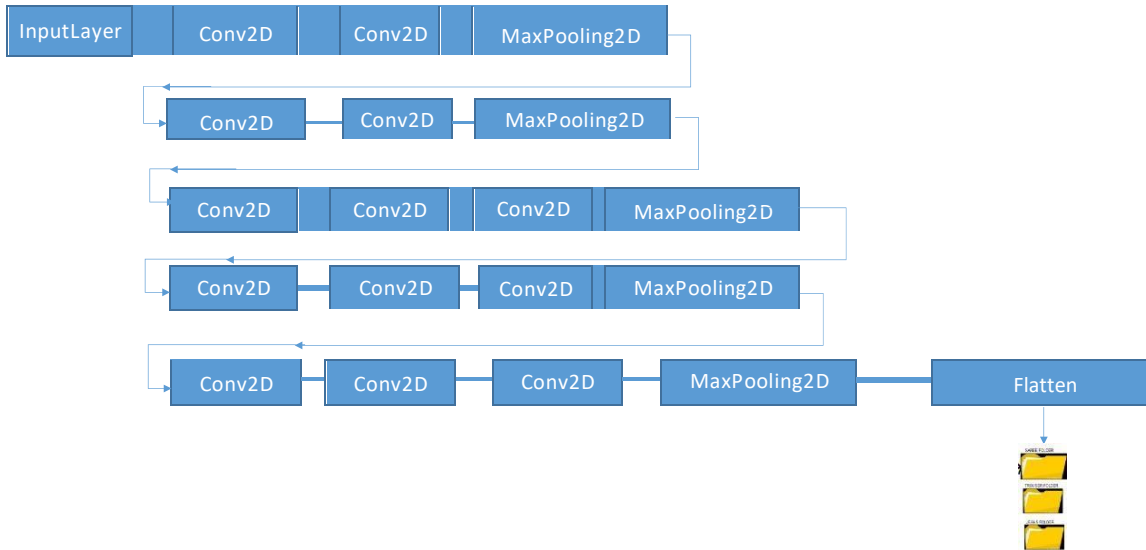
 Mathematical Optimisation done on the data after transfer learning is using Adam Optimiser. An optimization algorithm is a **procedure which is executed iteratively by comparing various solutions** till an optimum or a satisfactory solution is found, that is the design objective could be simply to **minimize the cost of production** or to **maximize the efficiency of production**
The Optimiser used in the model is Adam Optimiser
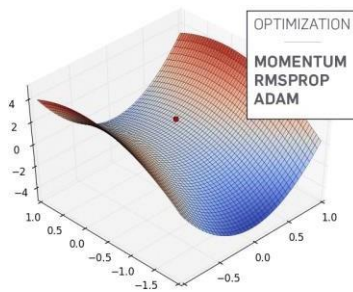
Algorithm 1: Adam Optimizer

Initialize $\theta_0, m_0 \leftarrow 0, v_0 \leftarrow 0, t \leftarrow 0$

While θ_t not converged

$\quad t \leftarrow t + 1$

$\quad g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

$\quad m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)g_t$

$\quad v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$

Bias Correction

$\quad \widehat{m_t} \leftarrow \frac{m_t}{1 - \beta_1^t}, \widehat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t}$

Update

$\quad \theta_t \leftarrow \prod_{\mathcal{F}, \sqrt{\widehat{v_t}}} \left(\theta_{t-1} - \frac{\alpha \widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon}\right)$

Data Sources and their formats

To undertake the experiments have Mined Data from Amazon and **flipkart.com** of the images of Sarees, Trousers and Jeans.

The images mined is balanced in nature
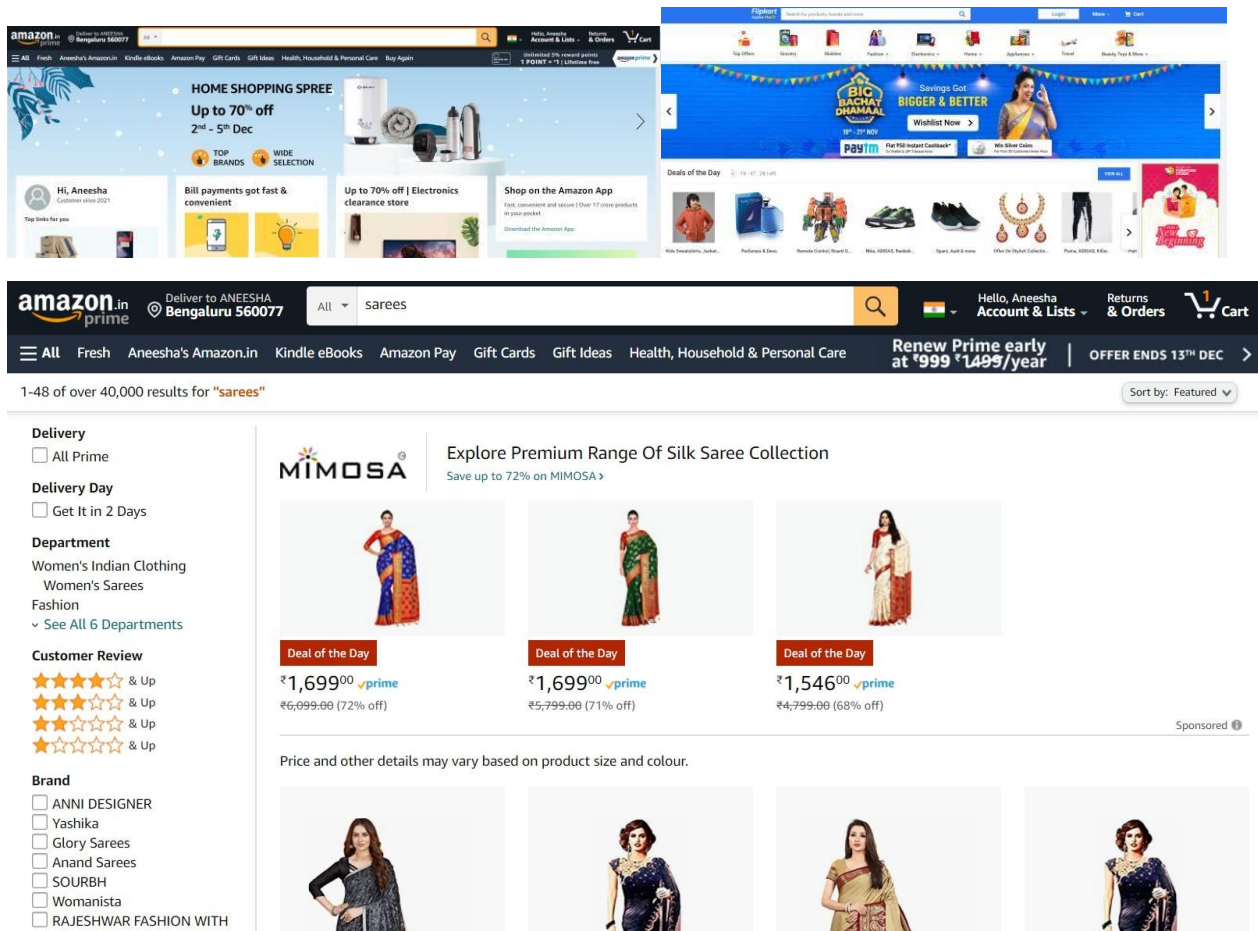
Scrapping was done through:

♦ Jupyter notebook
♦ Selenium

```
: import pandas as pd
  import selenium
  from selenium import webdriver
  import time
  from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException
  from selenium.webdriver.common.by import By
  from selenium.webdriver.common.keys import Keys
  import datetime
  from selenium.webdriver.common.action_chains import ActionChains
  from selenium.webdriver.common.keys import Keys
  import requests

  driver=webdriver.Chrome(r"D:\chromedriver.exe")
```

SAREES

```
url_saree='https://www.amazon.in/s?k=Sarees+%28women%29&ref=nb_sb_noss_2'
driver.get(url_saree)
time.sleep(2)
```

```
#scrapping images

a=0

for i in range(0,6):
        images = driver.find_elements_by_xpath('//img[@class="s-image"]')

        img_urls = []
        img_data = []
        for image in images:
            source= image.get_attribute('src')
            if source is not None:
                if(source[0:4] == 'http'):
                    img_urls.append(source)


        b=len(img_urls)
        for i in range(b):
            if i >= 1000:
                break
            print("Downloading {0} of {1} images" .format(i, 100))
            response= requests.get(img_urls[i])
            file = open(r"D:\folders\fliprobo\assignment 13\images\train_images_amazon\sarees\saree"+str(a)+str(i)+".jpg",
            file.write(response.content)
            time.sleep(2)

        try:
            button=driver.find_element_by_xpath("//*[contains(text(), 'Next')]")
            driver.execute_script("arguments[0].click();", button)
            a+=1
        except NoSuchElementException:
            pass
            time.sleep(2)
```

Folders created with the images:

📁 Jeans	26-11-2021 18:25	File folder
📁 Sarees	26-11-2021 18:12	File folder
📁 Trousers	26-11-2021 18:21	File folder

The data scrapped was stored into a folder called train_images and test_images

Hardware and Software Requirements and Tools Used

<u>Hardware technology being used</u>.

- RAM : 8 GB
- CPU :Intel® Core™ i7-10510U CPU @ 1.80GHz
- GPU: NVIDA-Cuda , performed in tf2.4 enviroment

<u>Software technology being used</u>.

- Programming language : Python
- Distribution : Anaconda Navigator
- Browser based language shell : Jupyter Notebook
- Libraries/Packages specifically being used.: Pandas , NumPy, matplotlib, keras,tensorflow

Model/s Development and Evaluation
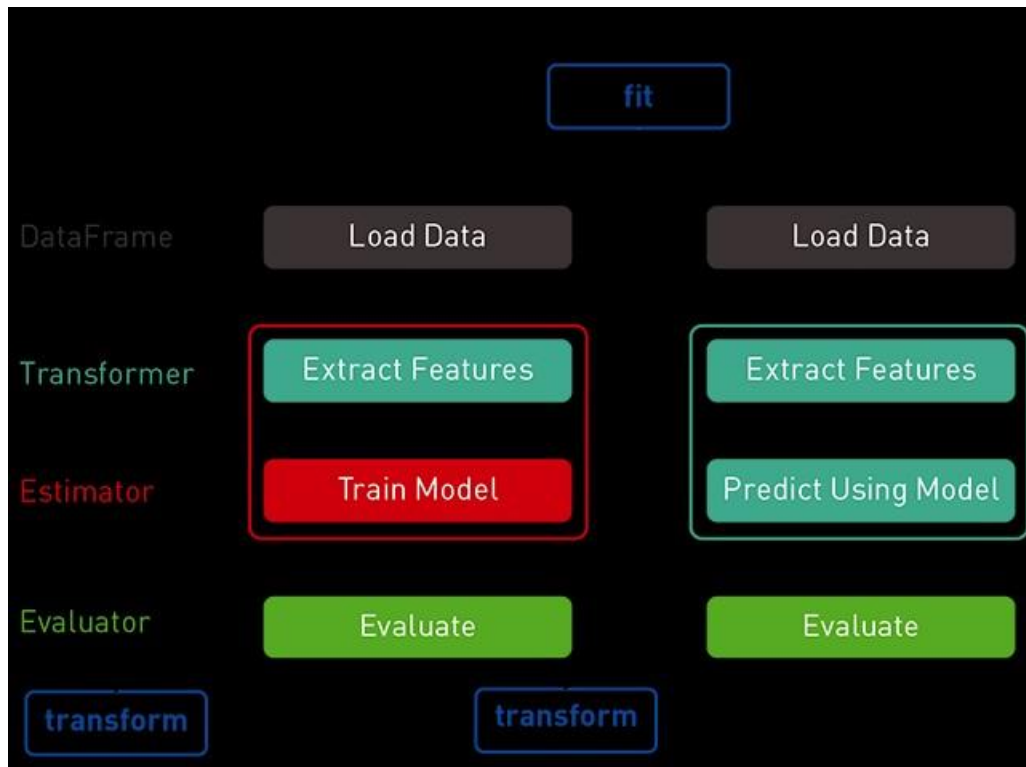
The problem at Hand is a MULTICLASS CLASSIFICATION problem

MODEL BUILDING

1.loading the models with no last layer

```
#Import the  library and add preprocessing layer to the front of transfer models
# Here we will be using imagenet weights

vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
vgg19 = VGG19(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
resnet = ResNet50(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

1. The weights used are of imagenet
2. The include_top=False, means the last layer is getting removed

```
# don't train existing layers- weights will change otherwise
for layer in vgg16.layers:
    layer.trainable = False

for layer in vgg19.layers:
    layer.trainable = False

for layer in resnet.layers:
    layer.trainable = False
```

2.Flattening the output

```
# our layers - you can add more if you want
x1 = Flatten()(vgg16.output)

x2 = Flatten()(vgg19.output)

x3 = Flatten()(resnet.output)
```

3.Checking our folder

```
# useful for getting number of output classes
folders = glob('images/train_images_amazon/*')
folders
```

```
['images/train_images_amazon\\Jeans',
 'images/train_images_amazon\\sarees',
 'images/train_images_amazon\\Trousers']
```

```
len(folders)
```

3

4.Appending folders as Dense Layer with activation function softmax

The folders will get appended as last layer with the flattened x

```
prediction1 = Dense(len(folders), activation='softmax')(x1)

prediction2 = Dense(len(folders), activation='softmax')(x2)

prediction3 = Dense(len(folders), activation='softmax')(x3)
```

```
print(prediction1)
print(prediction2)
print(prediction3)
```

```
KerasTensor(type_spec=TensorSpec(shape=(None, 3), dtype=tf.float32, name=None), name='dense/Softmax:0', description="created by
layer 'dense'")
KerasTensor(type_spec=TensorSpec(shape=(None, 3), dtype=tf.float32, name=None), name='dense_1/Softmax:0', description="created
by layer 'dense_1'")
KerasTensor(type_spec=TensorSpec(shape=(None, 3), dtype=tf.float32, name=None), name='dense_2/Softmax:0', description="created
by layer 'dense_2'")
```

5.Creating final model

```
# create a model object
model1 = Model(inputs=vgg16.input, outputs=prediction1)

model2 = Model(inputs=vgg19.input, outputs=prediction2)

model3 = Model(inputs=resnet.input, outputs=prediction3)
```

```
# view the structure of the model
print("--------model vgg16-----------------------------------------------------")
print(model1.summary())

print("--------model vgg19-----------------------------------------------------")
print(model2.summary())

print("--------model resnet----------------------------------------------------")
print(model3.summary())
```

```
# tell the model what cost and optimization method to use
model1.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)

model2.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)

model3.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

Data Augmentation

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator

#data augmentation
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

# Applying data augmentation
#making sure inputs are of same size

training_set = train_datagen.flow_from_directory('images/train_images_amazon',
                                                 target_size = (224, 224),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

```
Found 1063 images belonging to 3 classes.
```

```python
#data augmentation
test_datagen = ImageDataGenerator(rescale = 1./255)

test_set = test_datagen.flow_from_directory('images/test_images_flipkart',
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
Found 160 images belonging to 3 classes.
```

Applying model on data

```python
# fit the model
vgg16_model_fit = model1.fit_generator(
    training_set,
    validation_data=test_set,
    epochs=25,
    steps_per_epoch=len(training_set),
    validation_steps=len(test_set)
)
```

```
<ipython-input-15-9c8775406a34>:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Pl
ease use `Model.fit`, which supports generators.
  vgg16_model_fit = model1.fit_generator(
```

```
Epoch 1/25
34/34 [==============================] - 107s 3s/step - loss: 0.6247 - accuracy: 0.7178 - val_loss: 8.8683 - val_accuracy: 0.20
62
Epoch 2/25
34/34 [==============================] - 108s 3s/step - loss: 0.2648 - accuracy: 0.8833 - val_loss: 7.8828 - val_accuracy: 0.22
50
Epoch 3/25
34/34 [==============================] - 105s 3s/step - loss: 0.2059 - accuracy: 0.9097 - val_loss: 8.7312 - val_accuracy: 0.22
50
Epoch 4/25
34/34 [==============================] - 108s 3s/step - loss: 0.1954 - accuracy: 0.9210 - val_loss: 9.1597 - val_accuracy: 0.18
75
Epoch 5/25
34/34 [==============================] - 106s 3s/step - loss: 0.1517 - accuracy: 0.9407 - val_loss: 8.8249 - val_accuracy: 0.21
25
Epoch 6/25
34/34 [==============================] - 107s 3s/step - loss: 0.1148 - accuracy: 0.9661 - val_loss: 9.6407 - val_accuracy: 0.09
38
Epoch 7/25
34/34 [==============================] - 108s 3s/step - loss: 0.1279 - accuracy: 0.9558 - val_loss: 9.4853 - val_accuracy: 0.20
62
Epoch 8/25
34/34 [==============================] - 104s 3s/step - loss: 0.1253 - accuracy: 0.9548 - val_loss: 9.6380 - val_accuracy: 0.20
00
Epoch 9/25
34/34 [==============================] - 105s 3s/step - loss: 0.1160 - accuracy: 0.9605 - val_loss: 9.6670 - val_accuracy: 0.20
00
Epoch 10/25
34/34 [==============================] - 106s 3s/step - loss: 0.0816 - accuracy: 0.9718 - val_loss: 9.7572 - val_accuracy: 0.22
50
```

EVALUATION OF SELECTED MODELS

```python
losses_vgg16=pd.DataFrame(model1.history.history)
print("VGG16----------------------------------------------\n",losses_vgg16)
```

```
VGG16----------------------------------------------
        loss  accuracy   val_loss  val_accuracy
0    0.624749  0.717780   8.868276       0.20625
1    0.264838  0.883349   7.882825       0.22500
2    0.205907  0.909690   8.731160       0.22500
3    0.195442  0.920978   9.159671       0.18750
4    0.151661  0.940734   8.824918       0.21250
5    0.114818  0.966134   9.640717       0.09375
6    0.127930  0.955786   9.485296       0.20625
7    0.125320  0.954845   9.637982       0.20000
8    0.115979  0.960489   9.666961       0.20000
9    0.081588  0.971778   9.757243       0.22500
10   0.081679  0.968015   9.725302       0.19375
11   0.073832  0.971778  10.104113       0.20000
12   0.063672  0.980245  10.413096       0.21250
13   0.079410  0.969896  10.435382       0.20625
14   0.068529  0.981185  10.163783       0.20000
15   0.062313  0.976482  10.537808       0.23750
16   0.059408  0.979304  10.613699       0.20625
17   0.066524  0.980245  10.614845       0.21875
18   0.074910  0.969896  11.208082       0.21875
19   0.066881  0.975541  11.166627       0.20625
20   0.053155  0.985889  11.263213       0.19375
21   0.048707  0.984008  11.098978       0.08125
22   0.047899  0.984948  11.384108       0.19375
23   0.032589  0.994356  11.233387       0.18750
24   0.034166  0.993415  11.702196       0.18750
```

Higher epochs were not considered to avoid the vanishing gradient Problem
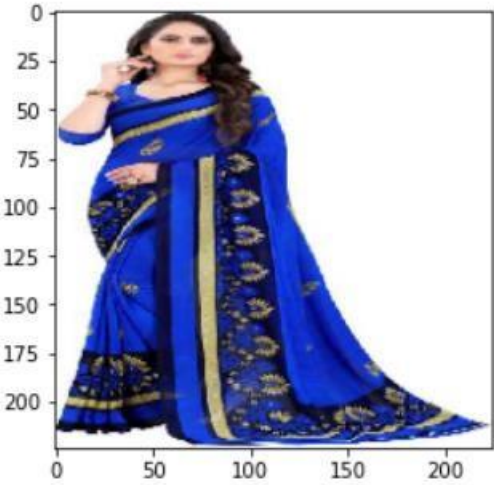
vgg16_model

vgg16_model
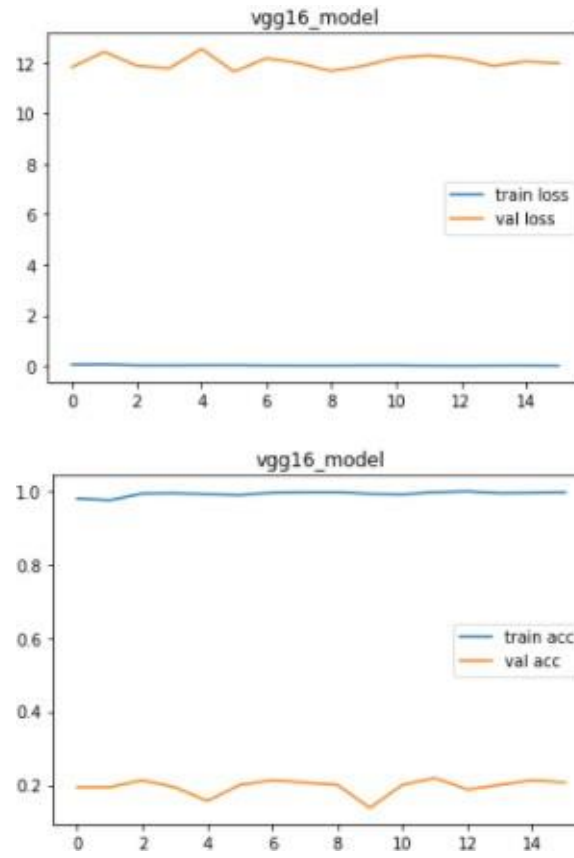
The loss and accuracy is seen to be optimum for the prediction, hence the models learning rate, batch size and learning rate seems to be optimum
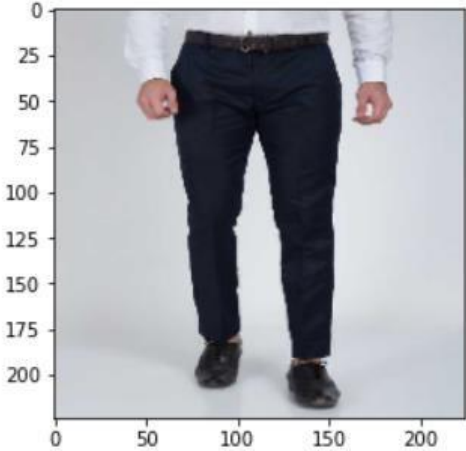
PREDICTION TESTING OF THE SELECTED MODEL

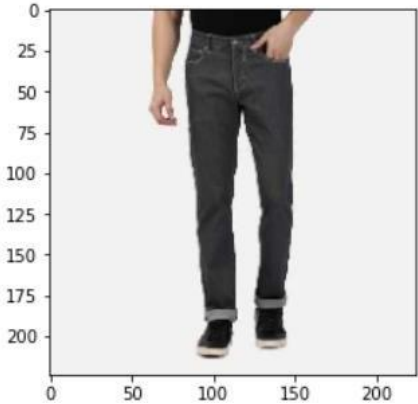|	names	labels
0	Jeans	0
1	Trousers	1
2	sarees	2

sarees

Trousers

Jeans

CONCLUSION

Key findings and conclusion of the study

The model predicts with an accuracy of 99% of images of sarees, Trousers and Jeans

Learning Outcomes of study with respect to Data Science

1. The model built shows that the larger the data, higher is the accuracy. Hence more train data the better accuracy we would get

2. Transfer models improves our resources compared to the model built by us from scratch with respect to projects as such.

3. Adam optimizer and Softmax proved well in respect to image classification of sarees, trousers and Jeans

Limitation of work and Scope for future improvement

If larger number of data was scrapped, better output could have been obtained

THANKYOU

****