Importing the Libraries

In [1]:

```
from tensorflow.keras.applications.imagenet_utils import preprocess_input
from tensorflow.keras.applications import VGG19, inception_v3, imagenet_utils
from tensorflow.keras.preprocessing.image import img_to_array, load_img, ImageDataGenerat
or

import numpy as np
import os
```

In [3]:

```
model = VGG19()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/
vgg19_weights_tf_dim_ordering_tf_kernels.h5
574717952/574710816 [==============================] - 12s 0us/step
```

In [ ]:

```
model.summary()
```

Lets try to predict the dress using VGG19 model

In [23]:

```
imagesize = (224,224)
image = load_img('/content/pic_12.jpg', target_size= imagesize)
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
image = preprocess_input(image)
pred = model.predict(image)
pred = imagenet_utils.decode_predictions(pred)
Class = pred[0][0][1]
Conf = pred[0][0][2]
print(Class, "With", Conf, "Confidence")
```

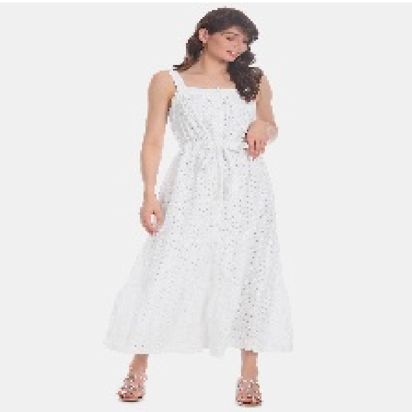
```
miniskirt With 0.8848513 Confidence
```

We can check for the first image our model predict the correct dress type i.e.miniskirt With 0.8848513 Confidence

In [24]:

```
load_img('/content/pic_111.jpg', target_size=(224,224))
```

Out[24]:



In [35]:

```
imagesize = (224,224)
image = load_img('/content/pic_111.jpg', target_size= imagesize)
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
image = preprocess_input(image)
pred = model.predict(image)
pred = imagenet_utils.decode_predictions(pred)
Class = pred[0][0][1]
Conf = pred[0][0][2]
print(Class, "With", Conf, "Confidence")
```

```
gown With 0.18726611 Confidence
```

In [ ]:

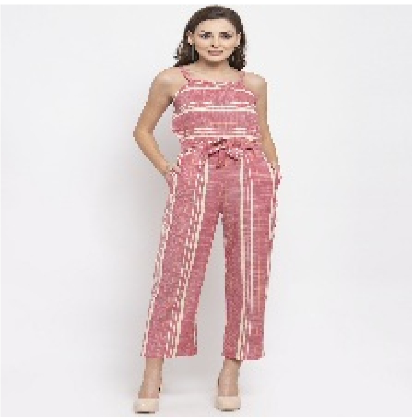
```

```

In [30]:

```
load_img('/content/pic_291.jpg', target_size=(224,224))
```

Out[30]:



In [29]:

```
imagesize = (224,224)
image = load_img('/content/pic_291.jpg', target_size= imagesize)
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
image = preprocess_input(image)
pred = model.predict(image)
pred = imagenet_utils.decode_predictions(pred)
Class = pred[0][0][1]
Conf = pred[0][0][2]
print(Class, "With", Conf, "Confidence")
```

```
stole With 0.16670981 Confidence
```

In [31]:

```

```

Lets try to buillt a own model on the given dataset and try to predict the Material, Pattern and Neckline.

In [2]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Conv2D, Dropout, MaxPooling2D, Flatten, Dense, Batch
Normalization, Input
```

```
import pandas as pd
import cv2
```

In [3]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [5]:

```
cd /content/drive/MyDrive/catalogue\ assignment
```

```
/content/drive/MyDrive/catalogue assignment
```

In [6]:

```
pwd
```

Out[6]:

```
'/content/drive/MyDrive/catalogue assignment'
```

In [7]:

```
df = pd.read_excel('dataset.xlsx')
```

In [8]:

```
df.head()
```

Out[8]:

	Title	Description	Material	Pattern	Neckline	Image_Path	Image_Path.1	Unnamed: 7
0	Peach Poly Crepe jumpsuit	This stylish foil print kurta from janasya is ...	Crepe	Printed	Round Neck	/images/pic_0.jpg	/images/pic_0.jpg	pic_0.jpg
1	Light Brown Bias Yoke Checks Top	This check pattern top by Work Label is crafte...	Cotton	Checks	Round Neck	/images/pic_1.jpg	/images/pic_1.jpg	pic_1.jpg
2	Off White Geometric Straight Cotton Dobby Top ...	Featuring elegant printed details, this off wh...	Viscose	Checks	Round Neck	/images/pic_2.jpg	/images/pic_2.jpg	NaN
3	Blue Me Away Cape Top	Add an extra dose of style to your casual ward...	Polyester	Solid/Plain	V-Neck	/images/pic_3.jpg	/images/pic_3.jpg	NaN
4	Yellow On A High Gown	Yellow polyester georgette maxi dress. Polyest...	Polyester	Solid/Plain	V-Neck	/images/pic_4.jpg	/images/pic_4.jpg	NaN

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Title         500 non-null    object
 1   Description   500 non-null    object
 2   Material      500 non-null    object
 3   Pattern       500 non-null    object
 4   Neckline      500 non-null    object
 5   Image_Path    500 non-null    object
 6   Image_Path.1  500 non-null    object
```

```
 7   Unnamed: 7    2 non-null      object
dtypes: object(8)
memory usage: 31.4+ KB
```

In [10]:

```
df = df.drop(['Image_Path.1', 'Unnamed: 7'], axis=1)
```

In [11]:

```
print('Total Number of Unique value in Material Column:',df.Material.nunique())
print('Total Number of Unique value in Pattern Column:',df.Pattern.nunique())
print('Total Number of Unique value in Neckline Column:', df.Neckline.nunique())
```

```
Total Number of Unique value in Material Column: 30
Total Number of Unique value in Pattern Column: 18
Total Number of Unique value in Neckline Column: 22
```

In [87]:

```
Materials = np.array(sorted(list(df.Material.unique())))
Materials
```

Out[87]:

```
array(['Blended Fabric', 'Chiffon', 'Cotton', 'Crepe', 'Crinkled',
       'Denim', 'Georgette', 'Khadi', 'Knitted', 'Lace', 'Leather',
       'Linen', 'Lyocell', 'Modal', 'Net', 'Nylon', 'Organic', 'Organza',
       'Polyamide', 'Polycotton', 'Polyester', 'Poplin', 'Rayon', 'Satin',
       'Sequin', 'Silk', 'Suede', 'Velvet', 'Viscose', 'Wool'],
      dtype='<U14')
```

In [89]:

```
Patterns = np.array(sorted(list(df.Pattern.unique())))
Patterns
```

Out[89]:

```
array(['Animal Print', 'Checks', 'Detailing', 'Embellished/Sequined',
       'Embroidered', 'Floral', 'Geometric', 'Mirror Work', 'Ombre',
       'Patterned', 'Plaid', 'Pleated', 'Polka Dots', 'Printed',
       'Ruffled', 'Solid/Plain', 'Stripes', 'Tie & Dye'], dtype='<U20')
```

In [90]:

```
Necklines = np.array(sorted(list(df.Neckline.unique())))
Necklines
```

Out[90]:

```
array(['Boat Neck', 'Cold Shoulder', 'Collar Neck', 'Cowl Neck',
       'Crew Neck', 'Halter Neck', 'High Neck', 'Hooded', 'Keyhole Neck',
       'Mandarin Neck', 'Off Shoulder', 'One Shoulder', 'Plunging Neck',
       'Queen Anne', 'Round Neck', 'Ruffled Neck', 'Scoop Neck',
       'Shoulder Straps', 'Square Neck', 'Strapless/Tube', 'Sweetheart',
       'V-Neck'], dtype='<U15')
```

Seperate the dependent and independent data

In [15]:

```
data = list(df['Image_Path'])
label_Material = df["Material"]
label_Pattern = df["Pattern"]
label_Neckline = df["Neckline"]
```

In [16]:

```
train = []
for imagePath in data:
```

```
    image = cv2.imread("."+imagePath)
    image = cv2.resize(image, (224,224))
    image = img_to_array(image)
    train.append(image)
```

In [17]:

```
Data = np.array(train)
label_Material = np.array(label_Material)
```

In [18]:

```
len(label_Material), len(Data)
```

Out[18]:

```
(500, 500)
```

In [194]:

```

```

Train a model to predict the material.

In [19]:

```
## Convert the labels to the machine understandable form
lb = LabelEncoder()
Label_Material = lb.fit_transform(label_Material)
Label_Materials = to_categorical(Label_Material)
```

In [20]:

```
len(Label_Materials[0])
```

Out[20]:

```
30
```

In [21]:

```
X_train, X_test, y_train, y_test = train_test_split(Data, Label_Materials, test_size=0.2
5 )
```

In [22]:

```
imageGen = ImageDataGenerator(rotation_range=30, height_shift_range=0.1, horizontal_flip=
True, vertical_flip=True, fill_mode='nearest', width_shift_range=0.1)
```

In [23]:

```
model = Sequential()
```

In [24]:

```
model.add(Conv2D(20, (5,5), activation='relu', input_shape=(224,224,3)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2)))


model.add(Conv2D(50, (5,5), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2)))

model.add(Conv2D(500, (5,5), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2)))

model.add(Flatten())
```

```python
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))

##Outputlayer
model.add(Dense(30, activation='softmax'))

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

In [25]:

```python
model.fit(imageGen.flow(X_train, y_train, batch_size=32), validation_data=(X_test, y_tes
t), epochs=50)
```

```
Epoch 1/50
12/12 [==============================] - 11s 367ms/step - loss: 146.4503 - accuracy: 0.16
18 - val_loss: 1859.7549 - val_accuracy: 0.2880
Epoch 2/50
12/12 [==============================] - 4s 319ms/step - loss: 171.2586 - accuracy: 0.205
0 - val_loss: 2163.8633 - val_accuracy: 0.0080
Epoch 3/50
12/12 [==============================] - 4s 320ms/step - loss: 104.3687 - accuracy: 0.247
8 - val_loss: 554.7886 - val_accuracy: 0.1440
Epoch 4/50
12/12 [==============================] - 4s 324ms/step - loss: 91.7396 - accuracy: 0.2679
- val_loss: 481.5151 - val_accuracy: 0.0960
Epoch 5/50
12/12 [==============================] - 4s 328ms/step - loss: 90.8535 - accuracy: 0.2189
- val_loss: 128.9599 - val_accuracy: 0.1760
Epoch 6/50
12/12 [==============================] - 4s 325ms/step - loss: 61.9983 - accuracy: 0.2640
- val_loss: 222.3885 - val_accuracy: 0.2720
Epoch 7/50
12/12 [==============================] - 4s 340ms/step - loss: 52.0703 - accuracy: 0.1853
- val_loss: 106.3356 - val_accuracy: 0.1120
Epoch 8/50
12/12 [==============================] - 4s 326ms/step - loss: 34.3199 - accuracy: 0.2215
- val_loss: 86.0017 - val_accuracy: 0.2160
Epoch 9/50
12/12 [==============================] - 4s 327ms/step - loss: 25.0528 - accuracy: 0.2375
- val_loss: 42.2863 - val_accuracy: 0.2080
Epoch 10/50
12/12 [==============================] - 4s 332ms/step - loss: 25.6171 - accuracy: 0.2157
- val_loss: 15.9778 - val_accuracy: 0.2320
Epoch 11/50
12/12 [==============================] - 4s 326ms/step - loss: 18.3058 - accuracy: 0.2055
- val_loss: 15.4092 - val_accuracy: 0.2240
Epoch 12/50
12/12 [==============================] - 4s 325ms/step - loss: 18.4169 - accuracy: 0.2383
- val_loss: 10.9831 - val_accuracy: 0.1760
Epoch 13/50
12/12 [==============================] - 4s 332ms/step - loss: 10.9898 - accuracy: 0.2653
- val_loss: 74.5023 - val_accuracy: 0.0640
Epoch 14/50
12/12 [==============================] - 4s 313ms/step - loss: 11.5964 - accuracy: 0.2502
- val_loss: 68.1587 - val_accuracy: 0.1520
Epoch 15/50
12/12 [==============================] - 4s 314ms/step - loss: 16.4728 - accuracy: 0.2688
- val_loss: 54.1093 - val_accuracy: 0.0880
Epoch 16/50
12/12 [==============================] - 4s 320ms/step - loss: 10.4028 - accuracy: 0.2505
- val_loss: 37.3896 - val_accuracy: 0.2880
Epoch 17/50
12/12 [==============================] - 4s 316ms/step - loss: 8.1423 - accuracy: 0.2872
- val_loss: 32.1946 - val_accuracy: 0.1760
Epoch 18/50
12/12 [==============================] - 4s 316ms/step - loss: 8.6923 - accuracy: 0.2389
- val_loss: 20.2474 - val_accuracy: 0.1280
Epoch 19/50
12/12 [==============================] - 4s 316ms/step - loss: 5.8122 - accuracy: 0.2687
- val_loss: 45.7015 - val_accuracy: 0.1840
Epoch 20/50
12/12 [==============================] - 4s 314ms/step - loss: 5.7319 - accuracy: 0.2278
```

```
- val_loss: 32.2533 - val_accuracy: 0.1600
Epoch 21/50
12/12 [==============================] - 4s 316ms/step - loss: 4.9591 - accuracy: 0.2955
- val_loss: 10.4609 - val_accuracy: 0.2320
Epoch 22/50
12/12 [==============================] - 4s 323ms/step - loss: 5.3722 - accuracy: 0.2808
- val_loss: 11.3460 - val_accuracy: 0.1760
Epoch 23/50
12/12 [==============================] - 4s 323ms/step - loss: 5.7186 - accuracy: 0.2964
- val_loss: 53.4697 - val_accuracy: 0.1600
Epoch 24/50
12/12 [==============================] - 4s 318ms/step - loss: 6.2505 - accuracy: 0.2668
- val_loss: 13.4883 - val_accuracy: 0.2000
Epoch 25/50
12/12 [==============================] - 4s 320ms/step - loss: 5.6291 - accuracy: 0.3025
- val_loss: 11.7721 - val_accuracy: 0.1200
Epoch 26/50
12/12 [==============================] - 4s 321ms/step - loss: 4.9032 - accuracy: 0.2759
- val_loss: 7.1522 - val_accuracy: 0.1120
Epoch 27/50
12/12 [==============================] - 4s 313ms/step - loss: 3.7843 - accuracy: 0.2864
- val_loss: 6.3048 - val_accuracy: 0.2640
Epoch 28/50
12/12 [==============================] - 4s 316ms/step - loss: 5.3339 - accuracy: 0.3026
- val_loss: 26.6835 - val_accuracy: 0.1360
Epoch 29/50
12/12 [==============================] - 4s 317ms/step - loss: 4.2846 - accuracy: 0.2670
- val_loss: 21.6983 - val_accuracy: 0.1280
Epoch 30/50
12/12 [==============================] - 4s 316ms/step - loss: 3.9690 - accuracy: 0.2685
- val_loss: 22.8627 - val_accuracy: 0.1840
Epoch 31/50
12/12 [==============================] - 4s 322ms/step - loss: 4.2349 - accuracy: 0.3016
- val_loss: 27.2164 - val_accuracy: 0.2480
Epoch 32/50
12/12 [==============================] - 4s 321ms/step - loss: 3.2897 - accuracy: 0.3159
- val_loss: 13.5270 - val_accuracy: 0.2240
Epoch 33/50
12/12 [==============================] - 4s 315ms/step - loss: 3.9039 - accuracy: 0.2370
- val_loss: 6.5771 - val_accuracy: 0.2560
Epoch 34/50
12/12 [==============================] - 4s 316ms/step - loss: 3.1721 - accuracy: 0.3262
- val_loss: 10.0269 - val_accuracy: 0.1200
Epoch 35/50
12/12 [==============================] - 4s 318ms/step - loss: 3.5493 - accuracy: 0.3316
- val_loss: 21.5114 - val_accuracy: 0.2720
Epoch 36/50
12/12 [==============================] - 4s 315ms/step - loss: 3.1754 - accuracy: 0.2817
- val_loss: 18.4833 - val_accuracy: 0.3040
Epoch 37/50
12/12 [==============================] - 4s 315ms/step - loss: 3.3454 - accuracy: 0.2738
- val_loss: 11.7960 - val_accuracy: 0.2160
Epoch 38/50
12/12 [==============================] - 4s 313ms/step - loss: 3.2433 - accuracy: 0.3115
- val_loss: 26.6633 - val_accuracy: 0.2240
Epoch 39/50
12/12 [==============================] - 4s 311ms/step - loss: 3.1265 - accuracy: 0.3028
- val_loss: 5.8699 - val_accuracy: 0.2320
Epoch 40/50
12/12 [==============================] - 4s 314ms/step - loss: 3.2214 - accuracy: 0.2967
- val_loss: 5.6354 - val_accuracy: 0.2400
Epoch 41/50
12/12 [==============================] - 4s 318ms/step - loss: 3.3273 - accuracy: 0.3177
- val_loss: 12.0149 - val_accuracy: 0.2240
Epoch 42/50
12/12 [==============================] - 4s 312ms/step - loss: 3.0154 - accuracy: 0.2966
- val_loss: 9.9877 - val_accuracy: 0.2480
Epoch 43/50
12/12 [==============================] - 4s 314ms/step - loss: 3.6234 - accuracy: 0.2734
- val_loss: 9.6197 - val_accuracy: 0.2400
Epoch 44/50
12/12 [==============================] - 4s 317ms/step - loss: 3.0659 - accuracy: 0.2897
```

```
- val_loss: 7.1339 - val_accuracy: 0.3360
Epoch 45/50
12/12 [==============================] - 4s 309ms/step - loss: 2.6666 - accuracy: 0.3192
- val_loss: 8.6196 - val_accuracy: 0.2640
Epoch 46/50
12/12 [==============================] - 4s 311ms/step - loss: 3.2532 - accuracy: 0.2825
- val_loss: 9.9626 - val_accuracy: 0.2480
Epoch 47/50
12/12 [==============================] - 4s 316ms/step - loss: 3.0084 - accuracy: 0.2424
- val_loss: 12.2933 - val_accuracy: 0.3440
Epoch 48/50
12/12 [==============================] - 4s 325ms/step - loss: 2.7542 - accuracy: 0.3067
- val_loss: 6.6502 - val_accuracy: 0.2960
Epoch 49/50
12/12 [==============================] - 4s 309ms/step - loss: 3.2493 - accuracy: 0.3258
- val_loss: 6.8469 - val_accuracy: 0.2480
Epoch 50/50
12/12 [==============================] - 4s 310ms/step - loss: 2.9103 - accuracy: 0.3213
- val_loss: 6.6942 - val_accuracy: 0.2880
```

Out[25]:

```
<tensorflow.python.keras.callbacks.History at 0x7f40d0189dd8>
```

The Accuracy is not good for our model. Beacuse the size of our dataset is very less.

Lets apply a VGG19 Model on our dataset and Try to predict the material from the image with the help of Tranfer Learning.

In [26]:

```
## Convert the images into the array
Train = []
for imagePath in data:
  image = load_img("."+ imagePath, target_size=(224,224))
  image = img_to_array(image)
  image = preprocess_input(image)
  Train.append(image)
```

In [27]:

```
Image_data = np.array(Train)
```

In [28]:

```
generator = ImageDataGenerator(rotation_range=30, height_shift_range=0.1, horizontal_flip
=True, vertical_flip=True, fill_mode='nearest', width_shift_range=0.1)
```

In [29]:

```
X_train, X_test, y_train, y_test = train_test_split(Image_data, Label_Materials, test_si
ze=0.25 )
```

In [30]:

```
### build model using transfer learning with VGG19 dataset

basemodel = VGG19(weights='imagenet', include_top=False, input_tensor=(Input(shape=(224,
224,3))))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/
vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80142336/80134624 [==============================] - 1s 0us/step
```

In [31]:

```
basemodel.summary()
```

```
Model: "vgg19"
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_conv4 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_conv4 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv4 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0         
=================================================================
Total params: 20,024,384
Trainable params: 20,024,384
Non-trainable params: 0
_________________________________________________________________
```

In [32]:

```
## Functional API
headmodel = basemodel.output    ## it will returns the last layer of our basemodel
headmodel = Flatten()(headmodel)
headmodel = Dense(1024, activation='relu')(headmodel)
headmodel = Dropout(0.5)(headmodel)
headmodel = Dense(30, activation='softmax')(headmodel)
```

In [33]:

```
material_model = Model(inputs= basemodel.input, outputs= headmodel)
```

In [34]:

```
## We don't want to train our pretrained basemodel VGG19
for layers in basemodel.layers:
  layers.trainable = False
```

In [ ]:

```
# material_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc
uracy'])
# print('Model is gong to train..........')
# PMaterial = material_ model.fit_generator(generator.flow(X_train, y_train, batch_size=3
2), validation_data=(X_test, y_test), epochs=100)
```

We can train our model for both training and testing data. But we have a very small dataset and many materials have only one pic in our datset that will make our data Underfitting as well as overfitting.

To reduce Underfitting we are using our whole dataset for the training.

In [35]:

```
material_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accu
racy'])
print('Model is gong to train..........')
PMaterial = material_model.fit_generator(generator.flow(Image_data, Label_Materials, batc
h_size=32), epochs=100)
```

```
Model is gong to train..........
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:1844: U
serWarning: `Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
Epoch 1/100
16/16 [==============================] - 6s 302ms/step - loss: 72.7254 - accuracy: 0.1712
Epoch 2/100
16/16 [==============================] - 5s 301ms/step - loss: 27.7748 - accuracy: 0.3273
Epoch 3/100
16/16 [==============================] - 5s 300ms/step - loss: 11.0313 - accuracy: 0.2676
Epoch 4/100
16/16 [==============================] - 5s 298ms/step - loss: 3.9881 - accuracy: 0.2726
Epoch 5/100
16/16 [==============================] - 5s 307ms/step - loss: 3.2635 - accuracy: 0.2886
Epoch 6/100
16/16 [==============================] - 5s 323ms/step - loss: 2.6493 - accuracy: 0.3276
Epoch 7/100
16/16 [==============================] - 5s 315ms/step - loss: 2.4809 - accuracy: 0.3849
Epoch 8/100
16/16 [==============================] - 5s 313ms/step - loss: 2.4116 - accuracy: 0.3935
Epoch 9/100
16/16 [==============================] - 5s 300ms/step - loss: 2.2046 - accuracy: 0.4582
Epoch 10/100
16/16 [==============================] - 5s 313ms/step - loss: 2.3757 - accuracy: 0.3622
Epoch 11/100
16/16 [==============================] - 5s 309ms/step - loss: 2.4413 - accuracy: 0.4191
Epoch 12/100
16/16 [==============================] - 5s 315ms/step - loss: 2.0140 - accuracy: 0.4421
Epoch 13/100
16/16 [==============================] - 5s 316ms/step - loss: 2.2955 - accuracy: 0.4184
Epoch 14/100
16/16 [==============================] - 5s 301ms/step - loss: 2.1892 - accuracy: 0.4411
Epoch 15/100
16/16 [==============================] - 5s 314ms/step - loss: 2.1878 - accuracy: 0.4677
Epoch 16/100
16/16 [==============================] - 5s 306ms/step - loss: 2.0024 - accuracy: 0.4342
Epoch 17/100
16/16 [==============================] - 5s 300ms/step - loss: 1.9215 - accuracy: 0.4392
Epoch 18/100
16/16 [==============================] - 5s 300ms/step - loss: 2.0644 - accuracy: 0.4398
Epoch 19/100
16/16 [==============================] - 5s 309ms/step - loss: 1.9084 - accuracy: 0.4748
Epoch 20/100
16/16 [==============================] - 5s 306ms/step - loss: 1.9791 - accuracy: 0.4629
Epoch 21/100
16/16 [==============================] - 5s 300ms/step - loss: 1.6865 - accuracy: 0.4551
Epoch 22/100
```

```
16/16 [==============================] - 5s 295ms/step - loss: 1.8038 - accuracy: 0.4521
Epoch 23/100
16/16 [==============================] - 5s 291ms/step - loss: 1.9473 - accuracy: 0.4664
Epoch 24/100
16/16 [==============================] - 5s 302ms/step - loss: 1.9338 - accuracy: 0.4700
Epoch 25/100
16/16 [==============================] - 5s 299ms/step - loss: 1.7169 - accuracy: 0.5019
Epoch 26/100
16/16 [==============================] - 5s 310ms/step - loss: 1.6497 - accuracy: 0.5029
Epoch 27/100
16/16 [==============================] - 5s 297ms/step - loss: 1.8241 - accuracy: 0.5189
Epoch 28/100
16/16 [==============================] - 5s 298ms/step - loss: 1.8033 - accuracy: 0.4590
Epoch 29/100
16/16 [==============================] - 5s 297ms/step - loss: 1.6122 - accuracy: 0.5217
Epoch 30/100
16/16 [==============================] - 5s 305ms/step - loss: 1.4825 - accuracy: 0.5191
Epoch 31/100
16/16 [==============================] - 5s 293ms/step - loss: 1.8192 - accuracy: 0.4590
Epoch 32/100
16/16 [==============================] - 5s 308ms/step - loss: 1.7741 - accuracy: 0.5154
Epoch 33/100
16/16 [==============================] - 5s 306ms/step - loss: 1.6666 - accuracy: 0.4971
Epoch 34/100
16/16 [==============================] - 5s 308ms/step - loss: 1.6075 - accuracy: 0.4905
Epoch 35/100
16/16 [==============================] - 5s 311ms/step - loss: 1.7137 - accuracy: 0.5198
Epoch 36/100
16/16 [==============================] - 5s 302ms/step - loss: 1.6180 - accuracy: 0.4944
Epoch 37/100
16/16 [==============================] - 5s 298ms/step - loss: 1.4927 - accuracy: 0.5962
Epoch 38/100
16/16 [==============================] - 5s 305ms/step - loss: 1.6101 - accuracy: 0.5465
Epoch 39/100
16/16 [==============================] - 5s 307ms/step - loss: 1.4202 - accuracy: 0.5717
Epoch 40/100
16/16 [==============================] - 5s 297ms/step - loss: 1.6408 - accuracy: 0.5471
Epoch 41/100
16/16 [==============================] - 5s 303ms/step - loss: 1.6893 - accuracy: 0.4866
Epoch 42/100
16/16 [==============================] - 5s 306ms/step - loss: 1.5196 - accuracy: 0.5695
Epoch 43/100
16/16 [==============================] - 5s 302ms/step - loss: 1.6336 - accuracy: 0.5148
Epoch 44/100
16/16 [==============================] - 5s 298ms/step - loss: 1.7738 - accuracy: 0.4838
Epoch 45/100
16/16 [==============================] - 5s 297ms/step - loss: 1.7213 - accuracy: 0.5540
Epoch 46/100
16/16 [==============================] - 5s 304ms/step - loss: 1.3585 - accuracy: 0.5716
Epoch 47/100
16/16 [==============================] - 5s 295ms/step - loss: 1.6418 - accuracy: 0.5249
Epoch 48/100
16/16 [==============================] - 5s 292ms/step - loss: 1.5456 - accuracy: 0.5262
Epoch 49/100
16/16 [==============================] - 5s 296ms/step - loss: 1.6674 - accuracy: 0.5267
Epoch 50/100
16/16 [==============================] - 5s 292ms/step - loss: 1.6648 - accuracy: 0.5468
Epoch 51/100
16/16 [==============================] - 5s 294ms/step - loss: 1.6135 - accuracy: 0.5484
Epoch 52/100
16/16 [==============================] - 5s 297ms/step - loss: 1.6036 - accuracy: 0.5139
Epoch 53/100
16/16 [==============================] - 5s 297ms/step - loss: 1.7754 - accuracy: 0.5306
Epoch 54/100
16/16 [==============================] - 5s 298ms/step - loss: 1.4440 - accuracy: 0.5752
Epoch 55/100
16/16 [==============================] - 5s 295ms/step - loss: 1.5574 - accuracy: 0.5445
Epoch 56/100
16/16 [==============================] - 5s 297ms/step - loss: 1.2849 - accuracy: 0.5981
Epoch 57/100
16/16 [==============================] - 5s 301ms/step - loss: 1.4817 - accuracy: 0.5714
Epoch 58/100
```

```
16/16 [==============================] - 5s 302ms/step - loss: 1.3916 - accuracy: 0.5973
Epoch 59/100
16/16 [==============================] - 5s 306ms/step - loss: 1.3443 - accuracy: 0.5599
Epoch 60/100
16/16 [==============================] - 5s 303ms/step - loss: 1.5447 - accuracy: 0.5434
Epoch 61/100
16/16 [==============================] - 5s 296ms/step - loss: 1.6136 - accuracy: 0.5714
Epoch 62/100
16/16 [==============================] - 5s 299ms/step - loss: 1.3633 - accuracy: 0.5890
Epoch 63/100
16/16 [==============================] - 5s 295ms/step - loss: 1.4623 - accuracy: 0.5311
Epoch 64/100
16/16 [==============================] - 5s 295ms/step - loss: 1.5037 - accuracy: 0.5917
Epoch 65/100
16/16 [==============================] - 5s 298ms/step - loss: 1.6519 - accuracy: 0.5572
Epoch 66/100
16/16 [==============================] - 5s 301ms/step - loss: 1.5433 - accuracy: 0.5426
Epoch 67/100
16/16 [==============================] - 5s 300ms/step - loss: 1.3222 - accuracy: 0.5778
Epoch 68/100
16/16 [==============================] - 5s 295ms/step - loss: 1.4831 - accuracy: 0.5764
Epoch 69/100
16/16 [==============================] - 5s 296ms/step - loss: 1.3483 - accuracy: 0.5944
Epoch 70/100
16/16 [==============================] - 5s 300ms/step - loss: 1.3526 - accuracy: 0.5660
Epoch 71/100
16/16 [==============================] - 5s 289ms/step - loss: 1.2521 - accuracy: 0.5938
Epoch 72/100
16/16 [==============================] - 5s 297ms/step - loss: 1.4500 - accuracy: 0.5792
Epoch 73/100
16/16 [==============================] - 5s 293ms/step - loss: 1.3388 - accuracy: 0.6185
Epoch 74/100
16/16 [==============================] - 5s 293ms/step - loss: 1.6419 - accuracy: 0.5495
Epoch 75/100
16/16 [==============================] - 5s 291ms/step - loss: 1.3231 - accuracy: 0.6397
Epoch 76/100
16/16 [==============================] - 5s 293ms/step - loss: 1.4573 - accuracy: 0.6091
Epoch 77/100
16/16 [==============================] - 5s 297ms/step - loss: 1.5091 - accuracy: 0.5622
Epoch 78/100
16/16 [==============================] - 5s 295ms/step - loss: 1.3122 - accuracy: 0.6315
Epoch 79/100
16/16 [==============================] - 5s 287ms/step - loss: 1.2464 - accuracy: 0.6216
Epoch 80/100
16/16 [==============================] - 5s 291ms/step - loss: 1.4105 - accuracy: 0.6363
Epoch 81/100
16/16 [==============================] - 5s 292ms/step - loss: 1.2844 - accuracy: 0.6092
Epoch 82/100
16/16 [==============================] - 5s 295ms/step - loss: 1.3762 - accuracy: 0.6030
Epoch 83/100
16/16 [==============================] - 5s 303ms/step - loss: 1.1360 - accuracy: 0.5850
Epoch 84/100
16/16 [==============================] - 5s 304ms/step - loss: 1.2127 - accuracy: 0.6212
Epoch 85/100
16/16 [==============================] - 5s 295ms/step - loss: 1.1646 - accuracy: 0.6224
Epoch 86/100
16/16 [==============================] - 5s 292ms/step - loss: 1.3154 - accuracy: 0.6128
Epoch 87/100
16/16 [==============================] - 5s 290ms/step - loss: 1.3325 - accuracy: 0.6439
Epoch 88/100
16/16 [==============================] - 5s 286ms/step - loss: 1.3247 - accuracy: 0.5996
Epoch 89/100
16/16 [==============================] - 5s 287ms/step - loss: 1.1875 - accuracy: 0.6400
Epoch 90/100
16/16 [==============================] - 5s 292ms/step - loss: 1.2103 - accuracy: 0.6740
Epoch 91/100
16/16 [==============================] - 5s 286ms/step - loss: 1.4393 - accuracy: 0.5812
Epoch 92/100
16/16 [==============================] - 5s 289ms/step - loss: 1.3181 - accuracy: 0.6096
Epoch 93/100
16/16 [==============================] - 5s 285ms/step - loss: 1.2880 - accuracy: 0.6488
Epoch 94/100
```

```
16/16 [==============================] - 5s 286ms/step - loss: 1.1328 - accuracy: 0.6421
Epoch 95/100
16/16 [==============================] - 5s 288ms/step - loss: 1.1522 - accuracy: 0.6366
Epoch 96/100
16/16 [==============================] - 5s 281ms/step - loss: 1.6275 - accuracy: 0.6136
Epoch 97/100
16/16 [==============================] - 5s 287ms/step - loss: 1.4154 - accuracy: 0.6129
Epoch 98/100
16/16 [==============================] - 5s 290ms/step - loss: 1.1448 - accuracy: 0.6584
Epoch 99/100
16/16 [==============================] - 5s 291ms/step - loss: 1.2754 - accuracy: 0.6151
Epoch 100/100
16/16 [==============================] - 5s 288ms/step - loss: 1.1823 - accuracy: 0.6535
```

In [102]:

```
##Lets predict
image = load_img('./images/pic_2.jpg', target_size=(224,224))
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
image = preprocess_input(image)

## lets make some prediction on images
pred = material_model.predict(image)
pred
```

Out[102]:

```
array([[1.2641018e-18, 1.4812258e-07, 1.6611574e-02, 5.9917811e-02,
        5.7133235e-16, 1.2553185e-09, 2.1460095e-05, 5.5108887e-15,
        4.3146418e-05, 5.1892597e-15, 1.0930961e-18, 1.6013593e-09,
        1.1774786e-08, 1.7516619e-09, 9.7918817e-10, 1.5545682e-08,
        3.2731129e-22, 1.0151598e-09, 4.0673514e-15, 1.0790802e-05,
        2.0609492e-01, 8.7787644e-24, 8.4526360e-02, 5.9509244e-07,
        4.0833881e-11, 1.2692946e-13, 1.8096080e-20, 1.5152658e-06,
        6.3277167e-01, 2.5990741e-13]], dtype=float32)
```

In [103]:

```
## Predicted class
pred_mat = np.argmax(pred)
ypred = Materials[pred_mat]

## Actual Class
actual = df[df['Image_Path'] == '/images/pic_2.jpg']
yactual = actual.Material.values

print(f'Our model predict {ypred} pattern.... and the real class is {yactual}')
```

```
Our model predict Viscose pattern.... and the real class is ['Viscose']
```

In [104]:

```
##Lets predict for another image
image = load_img('./images/pic_49.jpg', target_size=(224,224))
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
image = preprocess_input(image)

## lets make some prediction on images
pred = material_model.predict(image)
pred
```

Out[104]:

```
array([[6.5854345e-14, 7.1876508e-07, 1.1447824e-02, 4.2548282e-03,
        6.9239527e-15, 6.0462395e-09, 1.3293510e-05, 8.3617995e-14,
        2.9343297e-05, 3.7057327e-17, 4.9266587e-14, 2.9764356e-06,
        1.4596591e-06, 4.5630929e-07, 4.2368072e-08, 8.7130447e-06,
        5.2827301e-22, 2.9663443e-08, 1.8825368e-08, 9.5331267e-04,
        4.9701074e-01, 3.1822340e-16, 3.3296010e-01, 1.1915033e-07,
        1.8049764e-09, 9.1422833e-12, 8.6095843e-16, 1.4277776e-09,
        1.5331604e-01, 2.5439926e-13]], dtype=float32)
```

In [106]:

```
## Predicted class
pred_mat = np.argmax(pred)
ypred = Materials[pred_mat]

## Actual Class
actual = df[df['Image_Path'] == '/images/pic_49.jpg']
yactual = actual.Material.values

print(f'Our model predict {ypred} pattern.... and the real class is {yactual}')
```

```
Our model predict Polyester pattern.... and the real class is ['Rayon']
```

In [141]:

```
## Save the material model
material_model.save('Material_model.h5')
```

Lets try the same above model to predict the Pattern present in the image.

In [39]:

```
## Convert the labels to the machine understandable form
lb = LabelEncoder()
Label_Pattern = lb.fit_transform(label_Pattern)
Label_Patterns = to_categorical(Label_Pattern)
```

In [40]:

```
len(Label_Patterns[0])
```

Out[40]:

```
18
```

In [41]:

```
generator = ImageDataGenerator(rotation_range=30, height_shift_range=0.1, horizontal_flip
=True, vertical_flip=True, fill_mode='nearest', width_shift_range=0.1)
```

In [265]:

```
# X_train, X_test, y_train, y_test = train_test_split(Image_data, Label_Patterns, test_si
ze=0.25 )
```

In [42]:

```
### build model using transfer learning with VGG19 dataset

basemodel = VGG19(weights='imagenet', include_top=False, input_tensor=(Input(shape=(224,
224,3))))
```

In [43]:

```
headmodel = basemodel.output    ## it will returns the last layer of our basemodel
headmodel = Flatten()(headmodel)
headmodel = Dense(1024, activation='relu')(headmodel)
headmodel = Dropout(0.5)(headmodel)
headmodel = Dense(18, activation='softmax')(headmodel)

## Combine basemodel and headmodel in a single model
pattern_model = Model(inputs= basemodel.input, outputs= headmodel)
```

In [44]:

```
## We don't want to train our pretrained basemodel VGG19
for layers in basemodel.layers:
```

```
    layers.trainable = False
```

In [224]:

```
pattern_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accur
acy'])
print('Model is gong to train..........')
PPatterns = pattern_model.fit_generator(generator.flow(X_train, y_train, batch_size=32),
validation_data=(X_test, y_test), epochs=100)
```

```
Model is gong to train..........
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:1844: U
serWarning: `Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
Epoch 1/100
12/12 [==============================] - 6s 401ms/step - loss: 65.4850 - accuracy: 0.2316
- val_loss: 41.4870 - val_accuracy: 0.4240
Epoch 2/100
12/12 [==============================] - 4s 369ms/step - loss: 32.8938 - accuracy: 0.5375
- val_loss: 30.2185 - val_accuracy: 0.4080
Epoch 3/100
12/12 [==============================] - 5s 376ms/step - loss: 24.5160 - accuracy: 0.5258
- val_loss: 21.2205 - val_accuracy: 0.4720
Epoch 4/100
12/12 [==============================] - 4s 370ms/step - loss: 15.1141 - accuracy: 0.5306
- val_loss: 13.6091 - val_accuracy: 0.5280
Epoch 5/100
12/12 [==============================] - 4s 371ms/step - loss: 7.7149 - accuracy: 0.6493
- val_loss: 10.2982 - val_accuracy: 0.4800
Epoch 6/100
12/12 [==============================] - 5s 378ms/step - loss: 6.6053 - accuracy: 0.5592
- val_loss: 6.8253 - val_accuracy: 0.4880
Epoch 7/100
12/12 [==============================] - 4s 367ms/step - loss: 2.6252 - accuracy: 0.6708
- val_loss: 5.0217 - val_accuracy: 0.4640
Epoch 8/100
12/12 [==============================] - 4s 368ms/step - loss: 2.6519 - accuracy: 0.5554
- val_loss: 4.3650 - val_accuracy: 0.4240
Epoch 9/100
12/12 [==============================] - 5s 371ms/step - loss: 2.7171 - accuracy: 0.6298
- val_loss: 3.6152 - val_accuracy: 0.4640
Epoch 10/100
12/12 [==============================] - 5s 372ms/step - loss: 1.4417 - accuracy: 0.6795
- val_loss: 3.2638 - val_accuracy: 0.5120
Epoch 11/100
12/12 [==============================] - 5s 372ms/step - loss: 1.6021 - accuracy: 0.6944
- val_loss: 3.3055 - val_accuracy: 0.4800
Epoch 12/100
12/12 [==============================] - 4s 367ms/step - loss: 1.6050 - accuracy: 0.7004
- val_loss: 2.9412 - val_accuracy: 0.4560
Epoch 13/100
12/12 [==============================] - 4s 368ms/step - loss: 1.7219 - accuracy: 0.6176
- val_loss: 2.8818 - val_accuracy: 0.4800
Epoch 14/100
12/12 [==============================] - 5s 372ms/step - loss: 1.7182 - accuracy: 0.6085
- val_loss: 2.6447 - val_accuracy: 0.4960
Epoch 15/100
12/12 [==============================] - 4s 370ms/step - loss: 1.2459 - accuracy: 0.7440
- val_loss: 2.8318 - val_accuracy: 0.4560
Epoch 16/100
12/12 [==============================] - 5s 378ms/step - loss: 1.6230 - accuracy: 0.6663
- val_loss: 2.9327 - val_accuracy: 0.4560
Epoch 17/100
12/12 [==============================] - 4s 378ms/step - loss: 1.2259 - accuracy: 0.6925
- val_loss: 2.9689 - val_accuracy: 0.4400
Epoch 18/100
12/12 [==============================] - 4s 371ms/step - loss: 1.4014 - accuracy: 0.6843
- val_loss: 3.1273 - val_accuracy: 0.4640
Epoch 19/100
```

```
12/12 [==============================] - 5s 372ms/step - loss: 1.1562 - accuracy: 0.6989
- val_loss: 2.7257 - val_accuracy: 0.4640
Epoch 20/100
12/12 [==============================] - 4s 370ms/step - loss: 1.3295 - accuracy: 0.7396
- val_loss: 2.6909 - val_accuracy: 0.4720
Epoch 21/100
12/12 [==============================] - 5s 372ms/step - loss: 1.5009 - accuracy: 0.6698
- val_loss: 3.4583 - val_accuracy: 0.4960
Epoch 22/100
12/12 [==============================] - 5s 372ms/step - loss: 1.1335 - accuracy: 0.7472
- val_loss: 2.9045 - val_accuracy: 0.4880
Epoch 23/100
12/12 [==============================] - 5s 371ms/step - loss: 0.9565 - accuracy: 0.7577
- val_loss: 3.2300 - val_accuracy: 0.4960
Epoch 24/100
12/12 [==============================] - 4s 370ms/step - loss: 1.2499 - accuracy: 0.6953
- val_loss: 3.1929 - val_accuracy: 0.5440
Epoch 25/100
12/12 [==============================] - 5s 372ms/step - loss: 1.1046 - accuracy: 0.7512
- val_loss: 3.1512 - val_accuracy: 0.5280
Epoch 26/100
12/12 [==============================] - 5s 378ms/step - loss: 1.1216 - accuracy: 0.7633
- val_loss: 3.2692 - val_accuracy: 0.4560
Epoch 27/100
12/12 [==============================] - 5s 378ms/step - loss: 1.2328 - accuracy: 0.7387
- val_loss: 2.9385 - val_accuracy: 0.4560
Epoch 28/100
12/12 [==============================] - 5s 373ms/step - loss: 1.1070 - accuracy: 0.7059
- val_loss: 3.0080 - val_accuracy: 0.5200
Epoch 29/100
12/12 [==============================] - 4s 369ms/step - loss: 0.9744 - accuracy: 0.7498
- val_loss: 3.3017 - val_accuracy: 0.5200
Epoch 30/100
12/12 [==============================] - 5s 378ms/step - loss: 1.3822 - accuracy: 0.6932
- val_loss: 3.2081 - val_accuracy: 0.4640
Epoch 31/100
12/12 [==============================] - 4s 367ms/step - loss: 0.8859 - accuracy: 0.7409
- val_loss: 3.1540 - val_accuracy: 0.4640
Epoch 32/100
12/12 [==============================] - 4s 368ms/step - loss: 1.1385 - accuracy: 0.7390
- val_loss: 3.0799 - val_accuracy: 0.4960
Epoch 33/100
12/12 [==============================] - 5s 373ms/step - loss: 0.8657 - accuracy: 0.7824
- val_loss: 3.7045 - val_accuracy: 0.4880
Epoch 34/100
12/12 [==============================] - 5s 373ms/step - loss: 1.1751 - accuracy: 0.6950
- val_loss: 3.8692 - val_accuracy: 0.4880
Epoch 35/100
12/12 [==============================] - 5s 374ms/step - loss: 0.9288 - accuracy: 0.7665
- val_loss: 3.7861 - val_accuracy: 0.4880
Epoch 36/100
12/12 [==============================] - 5s 380ms/step - loss: 1.0000 - accuracy: 0.7456
- val_loss: 3.2037 - val_accuracy: 0.4720
Epoch 37/100
12/12 [==============================] - 5s 379ms/step - loss: 0.9585 - accuracy: 0.7888
- val_loss: 3.3596 - val_accuracy: 0.4720
Epoch 38/100
12/12 [==============================] - 4s 368ms/step - loss: 1.1666 - accuracy: 0.7567
- val_loss: 3.4014 - val_accuracy: 0.4720
Epoch 39/100
12/12 [==============================] - 5s 374ms/step - loss: 0.9681 - accuracy: 0.7454
- val_loss: 4.1078 - val_accuracy: 0.5600
Epoch 40/100
12/12 [==============================] - 4s 367ms/step - loss: 0.7317 - accuracy: 0.8014
- val_loss: 3.1009 - val_accuracy: 0.5040
Epoch 41/100
12/12 [==============================] - 5s 371ms/step - loss: 1.0591 - accuracy: 0.8001
- val_loss: 3.4787 - val_accuracy: 0.4800
Epoch 42/100
12/12 [==============================] - 5s 370ms/step - loss: 1.2347 - accuracy: 0.7047
- val_loss: 3.8975 - val_accuracy: 0.4400
Epoch 43/100
12/12 [==============================] - 5s 381ms/step - loss: 1.2799 - accuracy: 0.7540
```

```
12/12 [==============================] - 5s 381ms/step - loss: 1.2799 - accuracy: 0.7540
- val_loss: 3.5674 - val_accuracy: 0.5040
Epoch 44/100
12/12 [==============================] - 5s 388ms/step - loss: 0.6352 - accuracy: 0.8059
- val_loss: 3.9705 - val_accuracy: 0.5280
Epoch 45/100
12/12 [==============================] - 4s 369ms/step - loss: 1.4345 - accuracy: 0.7257
- val_loss: 4.3387 - val_accuracy: 0.5200
Epoch 46/100
12/12 [==============================] - 5s 391ms/step - loss: 1.0097 - accuracy: 0.8051
- val_loss: 3.9534 - val_accuracy: 0.5040
Epoch 47/100
12/12 [==============================] - 5s 375ms/step - loss: 1.1689 - accuracy: 0.7827
- val_loss: 3.4826 - val_accuracy: 0.4720
Epoch 48/100
12/12 [==============================] - 4s 368ms/step - loss: 0.8576 - accuracy: 0.8077
- val_loss: 3.8413 - val_accuracy: 0.5440
Epoch 49/100
12/12 [==============================] - 4s 369ms/step - loss: 0.9061 - accuracy: 0.7731
- val_loss: 3.4049 - val_accuracy: 0.5040
Epoch 50/100
12/12 [==============================] - 5s 378ms/step - loss: 0.7859 - accuracy: 0.7754
- val_loss: 3.4076 - val_accuracy: 0.4880
Epoch 51/100
12/12 [==============================] - 5s 371ms/step - loss: 0.7144 - accuracy: 0.8207
- val_loss: 3.5909 - val_accuracy: 0.5280
Epoch 52/100
12/12 [==============================] - 5s 371ms/step - loss: 0.8910 - accuracy: 0.8178
- val_loss: 3.5482 - val_accuracy: 0.5280
Epoch 53/100
12/12 [==============================] - 4s 369ms/step - loss: 1.0727 - accuracy: 0.7531
- val_loss: 3.8535 - val_accuracy: 0.4880
Epoch 54/100
12/12 [==============================] - 4s 367ms/step - loss: 0.9355 - accuracy: 0.7867
- val_loss: 4.7334 - val_accuracy: 0.5360
Epoch 55/100
12/12 [==============================] - 5s 381ms/step - loss: 0.8610 - accuracy: 0.7696
- val_loss: 5.1109 - val_accuracy: 0.5280
Epoch 56/100
12/12 [==============================] - 4s 369ms/step - loss: 0.8600 - accuracy: 0.8129
- val_loss: 4.1982 - val_accuracy: 0.5280
Epoch 57/100
12/12 [==============================] - 4s 369ms/step - loss: 0.8329 - accuracy: 0.7850
- val_loss: 3.7739 - val_accuracy: 0.5440
Epoch 58/100
12/12 [==============================] - 4s 370ms/step - loss: 0.9878 - accuracy: 0.7806
- val_loss: 3.6728 - val_accuracy: 0.5120
Epoch 59/100
12/12 [==============================] - 5s 376ms/step - loss: 0.9939 - accuracy: 0.7512
- val_loss: 3.3592 - val_accuracy: 0.5520
Epoch 60/100
12/12 [==============================] - 4s 369ms/step - loss: 0.9042 - accuracy: 0.7829
- val_loss: 4.2851 - val_accuracy: 0.5760
Epoch 61/100
12/12 [==============================] - 5s 375ms/step - loss: 0.7127 - accuracy: 0.7978
- val_loss: 4.0065 - val_accuracy: 0.5120
Epoch 62/100
12/12 [==============================] - 4s 377ms/step - loss: 0.5823 - accuracy: 0.8402
- val_loss: 4.3532 - val_accuracy: 0.5280
Epoch 63/100
12/12 [==============================] - 4s 369ms/step - loss: 0.8529 - accuracy: 0.8038
- val_loss: 4.3286 - val_accuracy: 0.5680
Epoch 64/100
12/12 [==============================] - 4s 371ms/step - loss: 0.9322 - accuracy: 0.7833
- val_loss: 3.8418 - val_accuracy: 0.5280
Epoch 65/100
12/12 [==============================] - 4s 370ms/step - loss: 0.7579 - accuracy: 0.8063
- val_loss: 3.6112 - val_accuracy: 0.5360
Epoch 66/100
12/12 [==============================] - 5s 380ms/step - loss: 0.7307 - accuracy: 0.8397
- val_loss: 3.4405 - val_accuracy: 0.5600
Epoch 67/100
12/12 [==============================] - 4s 370ms/step - loss: 0.4521 - accuracy: 0.8757
```

```
12/12 [==============================] - 4s 370ms/step - loss: 0.4521 - accuracy: 0.8757
- val_loss: 3.9466 - val_accuracy: 0.4160
Epoch 68/100
12/12 [==============================] - 4s 365ms/step - loss: 0.8757 - accuracy: 0.7760
- val_loss: 4.3238 - val_accuracy: 0.5360
Epoch 69/100
12/12 [==============================] - 5s 376ms/step - loss: 0.8380 - accuracy: 0.8032
- val_loss: 4.5450 - val_accuracy: 0.5200
Epoch 70/100
12/12 [==============================] - 4s 370ms/step - loss: 0.7123 - accuracy: 0.8172
- val_loss: 4.4296 - val_accuracy: 0.5600
Epoch 71/100
12/12 [==============================] - 4s 367ms/step - loss: 0.5591 - accuracy: 0.8379
- val_loss: 4.4365 - val_accuracy: 0.5600
Epoch 72/100
12/12 [==============================] - 4s 369ms/step - loss: 1.0431 - accuracy: 0.7642
- val_loss: 3.8140 - val_accuracy: 0.5440
Epoch 73/100
12/12 [==============================] - 4s 371ms/step - loss: 0.9733 - accuracy: 0.7582
- val_loss: 3.6561 - val_accuracy: 0.5200
Epoch 74/100
12/12 [==============================] - 5s 372ms/step - loss: 0.8551 - accuracy: 0.8006
- val_loss: 4.9616 - val_accuracy: 0.5440
Epoch 75/100
12/12 [==============================] - 5s 380ms/step - loss: 0.9365 - accuracy: 0.7980
- val_loss: 5.3011 - val_accuracy: 0.4960
Epoch 76/100
12/12 [==============================] - 4s 379ms/step - loss: 0.8306 - accuracy: 0.8089
- val_loss: 5.6771 - val_accuracy: 0.5360
Epoch 77/100
12/12 [==============================] - 4s 369ms/step - loss: 0.9812 - accuracy: 0.8166
- val_loss: 4.7132 - val_accuracy: 0.5680
Epoch 78/100
12/12 [==============================] - 5s 376ms/step - loss: 0.8561 - accuracy: 0.8383
- val_loss: 5.1028 - val_accuracy: 0.4720
Epoch 79/100
12/12 [==============================] - 4s 369ms/step - loss: 1.0817 - accuracy: 0.8148
- val_loss: 4.3834 - val_accuracy: 0.5200
Epoch 80/100
12/12 [==============================] - 4s 366ms/step - loss: 0.9272 - accuracy: 0.8459
- val_loss: 5.0028 - val_accuracy: 0.4560
Epoch 81/100
12/12 [==============================] - 4s 368ms/step - loss: 0.8293 - accuracy: 0.8045
- val_loss: 4.6892 - val_accuracy: 0.5200
Epoch 82/100
12/12 [==============================] - 5s 373ms/step - loss: 0.8589 - accuracy: 0.8139
- val_loss: 4.7470 - val_accuracy: 0.5280
Epoch 83/100
12/12 [==============================] - 5s 379ms/step - loss: 1.0931 - accuracy: 0.7992
- val_loss: 5.2794 - val_accuracy: 0.4800
Epoch 84/100
12/12 [==============================] - 5s 374ms/step - loss: 0.8340 - accuracy: 0.8159
- val_loss: 4.5077 - val_accuracy: 0.5040
Epoch 85/100
12/12 [==============================] - 4s 366ms/step - loss: 0.5354 - accuracy: 0.8564
- val_loss: 5.0359 - val_accuracy: 0.5360
Epoch 86/100
12/12 [==============================] - 4s 377ms/step - loss: 0.8648 - accuracy: 0.8403
- val_loss: 4.9937 - val_accuracy: 0.5200
Epoch 87/100
12/12 [==============================] - 5s 380ms/step - loss: 0.7622 - accuracy: 0.8324
- val_loss: 4.7854 - val_accuracy: 0.4800
Epoch 88/100
12/12 [==============================] - 5s 372ms/step - loss: 0.9742 - accuracy: 0.8407
- val_loss: 4.6778 - val_accuracy: 0.4640
Epoch 89/100
12/12 [==============================] - 4s 368ms/step - loss: 0.7801 - accuracy: 0.8455
- val_loss: 4.3216 - val_accuracy: 0.4960
Epoch 90/100
12/12 [==============================] - 4s 369ms/step - loss: 1.1043 - accuracy: 0.7977
- val_loss: 4.5864 - val_accuracy: 0.5280
Epoch 91/100
12/12 [==============================] - 5s 373ms/step - loss: 0.6711 - accuracy: 0.8310
```

```
12/12 [==============================] - 5s 372ms/step - loss: 0.6711 - accuracy: 0.8319
- val_loss: 4.3449 - val_accuracy: 0.5440
Epoch 92/100
12/12 [==============================] - 5s 375ms/step - loss: 0.7141 - accuracy: 0.8452
- val_loss: 4.1748 - val_accuracy: 0.4880
Epoch 93/100
12/12 [==============================] - 5s 371ms/step - loss: 1.0994 - accuracy: 0.8158
- val_loss: 5.0460 - val_accuracy: 0.4880
Epoch 94/100
12/12 [==============================] - 5s 387ms/step - loss: 0.7830 - accuracy: 0.8315
- val_loss: 5.2482 - val_accuracy: 0.5520
Epoch 95/100
12/12 [==============================] - 5s 373ms/step - loss: 0.8887 - accuracy: 0.8154
- val_loss: 5.5143 - val_accuracy: 0.5360
Epoch 96/100
12/12 [==============================] - 5s 377ms/step - loss: 0.6228 - accuracy: 0.8337
- val_loss: 4.5892 - val_accuracy: 0.5360
Epoch 97/100
12/12 [==============================] - 4s 367ms/step - loss: 0.8322 - accuracy: 0.8293
- val_loss: 3.9727 - val_accuracy: 0.5360
Epoch 98/100
12/12 [==============================] - 4s 370ms/step - loss: 0.6534 - accuracy: 0.8318
- val_loss: 4.0137 - val_accuracy: 0.5440
Epoch 99/100
12/12 [==============================] - 5s 392ms/step - loss: 0.8666 - accuracy: 0.8185
- val_loss: 4.0823 - val_accuracy: 0.5280
Epoch 100/100
12/12 [==============================] - 5s 370ms/step - loss: 0.7019 - accuracy: 0.8350
- val_loss: 4.3290 - val_accuracy: 0.5200
```

In [45]:

```
pattern_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accur
acy'])
print('Model is gong to train.........')
PPattern = pattern_model.fit_generator(generator.flow(Image_data, Label_Patterns, batch_s
ize=32), epochs=100)
```

```
Model is gong to train.........
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:1844: U
serWarning: `Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
Epoch 1/100
16/16 [==============================] - 5s 298ms/step - loss: 79.1379 - accuracy: 0.2431
Epoch 2/100
16/16 [==============================] - 5s 298ms/step - loss: 30.3525 - accuracy: 0.4779
Epoch 3/100
16/16 [==============================] - 5s 299ms/step - loss: 10.9469 - accuracy: 0.5282
Epoch 4/100
16/16 [==============================] - 5s 296ms/step - loss: 4.9651 - accuracy: 0.5576
Epoch 5/100
16/16 [==============================] - 5s 311ms/step - loss: 3.8326 - accuracy: 0.4879
Epoch 6/100
16/16 [==============================] - 5s 300ms/step - loss: 1.8905 - accuracy: 0.5939
Epoch 7/100
16/16 [==============================] - 5s 300ms/step - loss: 2.1221 - accuracy: 0.4924
Epoch 8/100
16/16 [==============================] - 5s 304ms/step - loss: 1.7905 - accuracy: 0.5959
Epoch 9/100
16/16 [==============================] - 5s 292ms/step - loss: 1.7741 - accuracy: 0.5845
Epoch 10/100
16/16 [==============================] - 5s 301ms/step - loss: 1.5759 - accuracy: 0.6059
Epoch 11/100
16/16 [==============================] - 5s 295ms/step - loss: 1.5744 - accuracy: 0.5764
Epoch 12/100
16/16 [==============================] - 5s 295ms/step - loss: 1.7861 - accuracy: 0.5744
Epoch 13/100
16/16 [==============================] - 5s 302ms/step - loss: 1.5464 - accuracy: 0.6085
Epoch 14/100
16/16 [==============================] - 5s 294ms/step - loss: 1.6411 - accuracy: 0.6000
```

```
Epoch 15/100
16/16 [==============================] - 5s 296ms/step - loss: 1.9795 - accuracy: 0.5631
Epoch 16/100
16/16 [==============================] - 5s 295ms/step - loss: 1.6365 - accuracy: 0.5985
Epoch 17/100
16/16 [==============================] - 5s 300ms/step - loss: 1.3570 - accuracy: 0.6409
Epoch 18/100
16/16 [==============================] - 5s 293ms/step - loss: 1.6420 - accuracy: 0.6039
Epoch 19/100
16/16 [==============================] - 5s 295ms/step - loss: 1.3972 - accuracy: 0.6350
Epoch 20/100
16/16 [==============================] - 5s 290ms/step - loss: 1.4717 - accuracy: 0.6331
Epoch 21/100
16/16 [==============================] - 5s 293ms/step - loss: 1.3235 - accuracy: 0.6338
Epoch 22/100
16/16 [==============================] - 5s 297ms/step - loss: 1.3402 - accuracy: 0.6586
Epoch 23/100
16/16 [==============================] - 5s 297ms/step - loss: 1.5061 - accuracy: 0.6545
Epoch 24/100
16/16 [==============================] - 5s 294ms/step - loss: 1.2493 - accuracy: 0.6241
Epoch 25/100
16/16 [==============================] - 5s 293ms/step - loss: 1.3045 - accuracy: 0.6587
Epoch 26/100
16/16 [==============================] - 5s 295ms/step - loss: 1.2707 - accuracy: 0.6739
Epoch 27/100
16/16 [==============================] - 5s 291ms/step - loss: 1.5803 - accuracy: 0.6363
Epoch 28/100
16/16 [==============================] - 5s 286ms/step - loss: 1.5203 - accuracy: 0.5962
Epoch 29/100
16/16 [==============================] - 5s 291ms/step - loss: 1.1455 - accuracy: 0.7084
Epoch 30/100
16/16 [==============================] - 5s 289ms/step - loss: 1.3935 - accuracy: 0.6153
Epoch 31/100
16/16 [==============================] - 5s 289ms/step - loss: 1.2483 - accuracy: 0.6837
Epoch 32/100
16/16 [==============================] - 5s 307ms/step - loss: 1.3514 - accuracy: 0.6791
Epoch 33/100
16/16 [==============================] - 5s 296ms/step - loss: 1.2192 - accuracy: 0.6789
Epoch 34/100
16/16 [==============================] - 5s 284ms/step - loss: 1.2741 - accuracy: 0.6774
Epoch 35/100
16/16 [==============================] - 5s 290ms/step - loss: 1.2272 - accuracy: 0.6889
Epoch 36/100
16/16 [==============================] - 5s 296ms/step - loss: 1.1106 - accuracy: 0.6875
Epoch 37/100
16/16 [==============================] - 5s 291ms/step - loss: 1.2120 - accuracy: 0.6881
Epoch 38/100
16/16 [==============================] - 5s 293ms/step - loss: 1.1841 - accuracy: 0.6768
Epoch 39/100
16/16 [==============================] - 5s 293ms/step - loss: 0.9906 - accuracy: 0.7376
Epoch 40/100
16/16 [==============================] - 5s 286ms/step - loss: 1.2815 - accuracy: 0.7154
Epoch 41/100
16/16 [==============================] - 5s 294ms/step - loss: 1.0040 - accuracy: 0.7056
Epoch 42/100
16/16 [==============================] - 5s 298ms/step - loss: 1.0953 - accuracy: 0.7227
Epoch 43/100
16/16 [==============================] - 5s 295ms/step - loss: 0.9435 - accuracy: 0.6954
Epoch 44/100
16/16 [==============================] - 5s 285ms/step - loss: 1.1422 - accuracy: 0.6935
Epoch 45/100
16/16 [==============================] - 5s 283ms/step - loss: 0.9045 - accuracy: 0.7349
Epoch 46/100
16/16 [==============================] - 5s 289ms/step - loss: 1.0795 - accuracy: 0.7276
Epoch 47/100
16/16 [==============================] - 5s 289ms/step - loss: 1.0026 - accuracy: 0.7062
Epoch 48/100
16/16 [==============================] - 5s 287ms/step - loss: 0.9133 - accuracy: 0.7272
Epoch 49/100
16/16 [==============================] - 5s 288ms/step - loss: 0.9119 - accuracy: 0.7384
Epoch 50/100
16/16 [==============================] - 5s 289ms/step - loss: 1.0611 - accuracy: 0.7039
```

```
Epoch 51/100
16/16 [==============================] - 5s 289ms/step - loss: 1.0580 - accuracy: 0.7407
Epoch 52/100
16/16 [==============================] - 5s 290ms/step - loss: 0.9846 - accuracy: 0.7253
Epoch 53/100
16/16 [==============================] - 5s 288ms/step - loss: 1.1302 - accuracy: 0.7279
Epoch 54/100
16/16 [==============================] - 5s 290ms/step - loss: 1.1906 - accuracy: 0.7217
Epoch 55/100
16/16 [==============================] - 5s 291ms/step - loss: 1.0774 - accuracy: 0.7035
Epoch 56/100
16/16 [==============================] - 5s 299ms/step - loss: 1.1877 - accuracy: 0.7241
Epoch 57/100
16/16 [==============================] - 5s 287ms/step - loss: 0.9470 - accuracy: 0.7288
Epoch 58/100
16/16 [==============================] - 5s 289ms/step - loss: 1.3208 - accuracy: 0.6968
Epoch 59/100
16/16 [==============================] - 5s 285ms/step - loss: 1.2935 - accuracy: 0.7124
Epoch 60/100
16/16 [==============================] - 5s 287ms/step - loss: 1.0121 - accuracy: 0.7349
Epoch 61/100
16/16 [==============================] - 5s 290ms/step - loss: 0.9802 - accuracy: 0.7336
Epoch 62/100
16/16 [==============================] - 5s 290ms/step - loss: 0.9209 - accuracy: 0.7706
Epoch 63/100
16/16 [==============================] - 5s 284ms/step - loss: 0.9699 - accuracy: 0.7110
Epoch 64/100
16/16 [==============================] - 5s 290ms/step - loss: 0.9881 - accuracy: 0.7245
Epoch 65/100
16/16 [==============================] - 5s 285ms/step - loss: 0.9962 - accuracy: 0.7482
Epoch 66/100
16/16 [==============================] - 5s 287ms/step - loss: 1.0238 - accuracy: 0.7289
Epoch 67/100
16/16 [==============================] - 5s 283ms/step - loss: 0.8856 - accuracy: 0.7410
Epoch 68/100
16/16 [==============================] - 5s 285ms/step - loss: 0.9280 - accuracy: 0.7612
Epoch 69/100
16/16 [==============================] - 5s 287ms/step - loss: 1.1911 - accuracy: 0.7251
Epoch 70/100
16/16 [==============================] - 5s 288ms/step - loss: 1.2296 - accuracy: 0.7206
Epoch 71/100
16/16 [==============================] - 5s 288ms/step - loss: 0.9149 - accuracy: 0.7588
Epoch 72/100
16/16 [==============================] - 5s 288ms/step - loss: 0.9864 - accuracy: 0.7068
Epoch 73/100
16/16 [==============================] - 5s 298ms/step - loss: 1.2122 - accuracy: 0.7091
Epoch 74/100
16/16 [==============================] - 5s 295ms/step - loss: 0.9538 - accuracy: 0.7314
Epoch 75/100
16/16 [==============================] - 5s 284ms/step - loss: 0.7935 - accuracy: 0.7585
Epoch 76/100
16/16 [==============================] - 5s 282ms/step - loss: 0.6864 - accuracy: 0.8163
Epoch 77/100
16/16 [==============================] - 5s 287ms/step - loss: 1.1142 - accuracy: 0.7062
Epoch 78/100
16/16 [==============================] - 5s 287ms/step - loss: 0.8882 - accuracy: 0.7604
Epoch 79/100
16/16 [==============================] - 5s 296ms/step - loss: 0.9729 - accuracy: 0.7096
Epoch 80/100
16/16 [==============================] - 5s 284ms/step - loss: 1.0677 - accuracy: 0.7738
Epoch 81/100
16/16 [==============================] - 5s 281ms/step - loss: 1.0577 - accuracy: 0.7618
Epoch 82/100
16/16 [==============================] - 5s 287ms/step - loss: 0.7331 - accuracy: 0.7972
Epoch 83/100
16/16 [==============================] - 5s 297ms/step - loss: 1.0506 - accuracy: 0.7143
Epoch 84/100
16/16 [==============================] - 5s 287ms/step - loss: 0.9536 - accuracy: 0.7587
Epoch 85/100
16/16 [==============================] - 5s 288ms/step - loss: 0.8952 - accuracy: 0.7395
Epoch 86/100
16/16 [==============================] - 5s 291ms/step - loss: 0.9752 - accuracy: 0.7559
```

```
Epoch 87/100
16/16 [==============================] - 5s 289ms/step - loss: 0.7562 - accuracy: 0.7841
Epoch 88/100
16/16 [==============================] - 5s 289ms/step - loss: 0.8755 - accuracy: 0.8031
Epoch 89/100
16/16 [==============================] - 5s 294ms/step - loss: 0.9947 - accuracy: 0.7676
Epoch 90/100
16/16 [==============================] - 5s 289ms/step - loss: 0.8109 - accuracy: 0.7503
Epoch 91/100
16/16 [==============================] - 5s 287ms/step - loss: 0.9137 - accuracy: 0.7333
Epoch 92/100
16/16 [==============================] - 5s 289ms/step - loss: 0.7543 - accuracy: 0.7630
Epoch 93/100
16/16 [==============================] - 5s 287ms/step - loss: 0.6605 - accuracy: 0.8064
Epoch 94/100
16/16 [==============================] - 5s 286ms/step - loss: 1.0369 - accuracy: 0.7402
Epoch 95/100
16/16 [==============================] - 5s 287ms/step - loss: 0.9437 - accuracy: 0.7278
Epoch 96/100
16/16 [==============================] - 5s 294ms/step - loss: 0.7820 - accuracy: 0.7744
Epoch 97/100
16/16 [==============================] - 5s 284ms/step - loss: 0.7163 - accuracy: 0.7861
Epoch 98/100
16/16 [==============================] - 5s 292ms/step - loss: 0.7971 - accuracy: 0.7456
Epoch 99/100
16/16 [==============================] - 5s 285ms/step - loss: 0.8895 - accuracy: 0.7693
Epoch 100/100
16/16 [==============================] - 5s 290ms/step - loss: 0.8829 - accuracy: 0.7823
```

In [119]:

```
## Check the prediction

image = load_img('./images/pic_99.jpg', target_size=(224,224))
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
image = preprocess_input(image)

## lets make some prediction on images
pred = pattern_model.predict(image)
pred
```

Out[119]:

```
array([[1.2150788e-30, 2.7571619e-23, 2.1156658e-17, 1.2301986e-16,
        4.7250459e-19, 1.2336007e-24, 1.3176320e-24, 0.0000000e+00,
        4.9878679e-25, 1.9322758e-16, 0.0000000e+00, 7.2863400e-20,
        7.2781691e-21, 6.6929000e-14, 4.5732005e-13, 1.0000000e+00,
        6.6017899e-22, 0.0000000e+00]], dtype=float32)
```

In [121]:

```
## Predicted class
pred_pat = np.argmax(pred)
ypred = Patterns[pred_pat]

## Actual class
actual = df[df['Image_Path'] == '/images/pic_99.jpg']
yactual = actual['Pattern'].values

print(f'Our model predict {ypred} pattern.... and the real class is {yactual}')
```

```
Our model predict Solid/Plain pattern.... and the real class is ['Solid/Plain']
```

In [131]:

```
## Check the prediction

image = load_img('./images/pic_39.jpg', target_size=(224,224))
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
```

```
image = preprocess_input(image)

## lets make some prediction on images
pred = pattern_model.predict(image)
pred
```

Out[131]:

```
array([[2.4806906e-13, 7.5490025e-06, 2.9408242e-05, 2.3496255e-05,
        1.1917627e-03, 1.7136354e-02, 2.7223192e-07, 5.3202300e-09,
        6.1833393e-11, 1.4149218e-06, 3.3586048e-08, 2.1937602e-07,
        7.2781802e-03, 9.7432762e-01, 2.7041349e-12, 3.7578914e-06,
        1.5903987e-07, 2.4401462e-12]], dtype=float32)
```

In [132]:

```
## Predicted class
pred_pat = np.argmax(pred)
ypred = Patterns[pred_pat]

## Actual class
actual = df[df['Image_Path'] == '/images/pic_39.jpg']
yactual = actual['Pattern'].values

print(f'Our model predict {ypred} pattern.... and the real class is {yactual}')
```

```
Our model predict Printed pattern.... and the real class is ['Printed']
```

In [140]:

```
## Save the model
pattern_model.save('Pattern_model.h5')
```

Lets try the same above model to predict the Neckline present in the image.

In [49]:

```
## Convert the labels to the machine understandable form
lb = LabelEncoder()
Label_Neckline = lb.fit_transform(label_Neckline)
Label_Necklines = to_categorical(Label_Neckline)
```

In [50]:

```
len(Label_Necklines[0])
```

Out[50]:

```
22
```

In [51]:

```
generator = ImageDataGenerator(rotation_range=30, height_shift_range=0.1, horizontal_flip
=True, vertical_flip=True, fill_mode='nearest', width_shift_range=0.1)
```

In [52]:

```
# X_train, X_test, y_train, y_test = train_test_split(Image_data, Label_Necklines, test_s
ize=0.25 )
```

In [53]:

```
### build model using transfer learning with VGG19 dataset

basemodel = VGG19(weights='imagenet', include_top=False, input_tensor=(Input(shape=(224,
224,3))))
```

In [54]:

```
headmodel = basemodel.output     ## it will returns the last layer of our basemodel
```

```python
headmodel = Flatten()(headmodel)
headmodel = Dense(1024, activation='relu')(headmodel)
headmodel = Dropout(0.5)(headmodel)
headmodel = Dense(22, activation='softmax')(headmodel)

## Combine basemodel and headmodel in a single model
neckline_model = Model(inputs= basemodel.input, outputs= headmodel)
```

In [55]:

```python
## We don't want to train our pretrained basemodel VGG19
for layers in basemodel.layers:
  layers.trainable = False
```

In [56]:

```python
neckline_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accu
racy'])
print('Model is gong to train..........')
PNeckline = neckline_model.fit_generator(generator.flow(Image_data, Label_Necklines, batc
h_size=32), epochs=100)
```

```
Model is gong to train..........
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:1844: U
serWarning: `Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
```

```
Epoch 1/100
16/16 [==============================] - 5s 296ms/step - loss: 61.3377 - accuracy: 0.1314
Epoch 2/100
16/16 [==============================] - 5s 301ms/step - loss: 29.3260 - accuracy: 0.2632
Epoch 3/100
16/16 [==============================] - 5s 297ms/step - loss: 10.9440 - accuracy: 0.2483
Epoch 4/100
16/16 [==============================] - 5s 300ms/step - loss: 4.0435 - accuracy: 0.2436
Epoch 5/100
16/16 [==============================] - 5s 309ms/step - loss: 2.9519 - accuracy: 0.2662
Epoch 6/100
16/16 [==============================] - 5s 305ms/step - loss: 3.0388 - accuracy: 0.2254
Epoch 7/100
16/16 [==============================] - 5s 301ms/step - loss: 2.9777 - accuracy: 0.2218
Epoch 8/100
16/16 [==============================] - 5s 302ms/step - loss: 3.0997 - accuracy: 0.2638
Epoch 9/100
16/16 [==============================] - 5s 299ms/step - loss: 2.8074 - accuracy: 0.2776
Epoch 10/100
16/16 [==============================] - 5s 303ms/step - loss: 2.6632 - accuracy: 0.2810
Epoch 11/100
16/16 [==============================] - 5s 297ms/step - loss: 2.7482 - accuracy: 0.3312
Epoch 12/100
16/16 [==============================] - 5s 303ms/step - loss: 2.5653 - accuracy: 0.3073
Epoch 13/100
16/16 [==============================] - 5s 305ms/step - loss: 2.5116 - accuracy: 0.2912
Epoch 14/100
16/16 [==============================] - 5s 305ms/step - loss: 2.5357 - accuracy: 0.2953
Epoch 15/100
16/16 [==============================] - 5s 303ms/step - loss: 2.4201 - accuracy: 0.3514
Epoch 16/100
16/16 [==============================] - 5s 311ms/step - loss: 2.4819 - accuracy: 0.3371
Epoch 17/100
16/16 [==============================] - 5s 320ms/step - loss: 2.5188 - accuracy: 0.3121
Epoch 18/100
16/16 [==============================] - 5s 315ms/step - loss: 2.3676 - accuracy: 0.3303
Epoch 19/100
16/16 [==============================] - 5s 310ms/step - loss: 2.6027 - accuracy: 0.2923
Epoch 20/100
16/16 [==============================] - 5s 302ms/step - loss: 2.3309 - accuracy: 0.3567
Epoch 21/100
16/16 [==============================] - 5s 306ms/step - loss: 2.3098 - accuracy: 0.3437
Epoch 22/100
```

```
16/16 [==============================] - 5s 303ms/step - loss: 2.3873 - accuracy: 0.3384
Epoch 23/100
16/16 [==============================] - 5s 300ms/step - loss: 2.5743 - accuracy: 0.3841
Epoch 24/100
16/16 [==============================] - 5s 303ms/step - loss: 2.2213 - accuracy: 0.3544
Epoch 25/100
16/16 [==============================] - 5s 302ms/step - loss: 2.2030 - accuracy: 0.3670
Epoch 26/100
16/16 [==============================] - 5s 304ms/step - loss: 2.1807 - accuracy: 0.3542
Epoch 27/100
16/16 [==============================] - 5s 302ms/step - loss: 2.2839 - accuracy: 0.3297
Epoch 28/100
16/16 [==============================] - 5s 303ms/step - loss: 2.2263 - accuracy: 0.3858
Epoch 29/100
16/16 [==============================] - 5s 301ms/step - loss: 2.3932 - accuracy: 0.3456
Epoch 30/100
16/16 [==============================] - 5s 304ms/step - loss: 2.2175 - accuracy: 0.3453
Epoch 31/100
16/16 [==============================] - 5s 306ms/step - loss: 2.2840 - accuracy: 0.3783
Epoch 32/100
16/16 [==============================] - 5s 315ms/step - loss: 2.1406 - accuracy: 0.4090
Epoch 33/100
16/16 [==============================] - 5s 310ms/step - loss: 2.0844 - accuracy: 0.4118
Epoch 34/100
16/16 [==============================] - 5s 306ms/step - loss: 2.2348 - accuracy: 0.3927
Epoch 35/100
16/16 [==============================] - 5s 303ms/step - loss: 2.1814 - accuracy: 0.4170
Epoch 36/100
16/16 [==============================] - 5s 303ms/step - loss: 1.9831 - accuracy: 0.3693
Epoch 37/100
16/16 [==============================] - 5s 304ms/step - loss: 2.2346 - accuracy: 0.3731
Epoch 38/100
16/16 [==============================] - 5s 310ms/step - loss: 1.9529 - accuracy: 0.4252
Epoch 39/100
16/16 [==============================] - 5s 301ms/step - loss: 2.0630 - accuracy: 0.3730
Epoch 40/100
16/16 [==============================] - 5s 301ms/step - loss: 1.9705 - accuracy: 0.4157
Epoch 41/100
16/16 [==============================] - 5s 297ms/step - loss: 2.3780 - accuracy: 0.3623
Epoch 42/100
16/16 [==============================] - 5s 305ms/step - loss: 2.1133 - accuracy: 0.3744
Epoch 43/100
16/16 [==============================] - 5s 295ms/step - loss: 1.9756 - accuracy: 0.3962
Epoch 44/100
16/16 [==============================] - 5s 296ms/step - loss: 2.0457 - accuracy: 0.3901
Epoch 45/100
16/16 [==============================] - 5s 295ms/step - loss: 2.0160 - accuracy: 0.4450
Epoch 46/100
16/16 [==============================] - 5s 297ms/step - loss: 1.9295 - accuracy: 0.3893
Epoch 47/100
16/16 [==============================] - 5s 298ms/step - loss: 2.0619 - accuracy: 0.3877
Epoch 48/100
16/16 [==============================] - 5s 297ms/step - loss: 2.1282 - accuracy: 0.4210
Epoch 49/100
16/16 [==============================] - 5s 300ms/step - loss: 1.9189 - accuracy: 0.4086
Epoch 50/100
16/16 [==============================] - 5s 298ms/step - loss: 1.9091 - accuracy: 0.4208
Epoch 51/100
16/16 [==============================] - 5s 308ms/step - loss: 1.8878 - accuracy: 0.4164
Epoch 52/100
16/16 [==============================] - 5s 297ms/step - loss: 2.0754 - accuracy: 0.3878
Epoch 53/100
16/16 [==============================] - 5s 295ms/step - loss: 2.0824 - accuracy: 0.3920
Epoch 54/100
16/16 [==============================] - 5s 299ms/step - loss: 2.0879 - accuracy: 0.3684
Epoch 55/100
16/16 [==============================] - 5s 301ms/step - loss: 1.8999 - accuracy: 0.4356
Epoch 56/100
16/16 [==============================] - 5s 299ms/step - loss: 1.8413 - accuracy: 0.3783
Epoch 57/100
16/16 [==============================] - 5s 300ms/step - loss: 1.9015 - accuracy: 0.4663
Epoch 58/100
```

```
16/16 [==============================] - 5s 300ms/step - loss: 2.0482 - accuracy: 0.3915
Epoch 59/100
16/16 [==============================] - 5s 294ms/step - loss: 1.8992 - accuracy: 0.4331
Epoch 60/100
16/16 [==============================] - 5s 301ms/step - loss: 1.8570 - accuracy: 0.4419
Epoch 61/100
16/16 [==============================] - 5s 294ms/step - loss: 2.0883 - accuracy: 0.3720
Epoch 62/100
16/16 [==============================] - 5s 297ms/step - loss: 1.8988 - accuracy: 0.4118
Epoch 63/100
16/16 [==============================] - 5s 299ms/step - loss: 2.0056 - accuracy: 0.4137
Epoch 64/100
16/16 [==============================] - 5s 296ms/step - loss: 1.8983 - accuracy: 0.4434
Epoch 65/100
16/16 [==============================] - 5s 295ms/step - loss: 1.9954 - accuracy: 0.4314
Epoch 66/100
16/16 [==============================] - 5s 295ms/step - loss: 1.7865 - accuracy: 0.4534
Epoch 67/100
16/16 [==============================] - 5s 301ms/step - loss: 1.8086 - accuracy: 0.4473
Epoch 68/100
16/16 [==============================] - 5s 301ms/step - loss: 1.8348 - accuracy: 0.4454
Epoch 69/100
16/16 [==============================] - 5s 295ms/step - loss: 1.7685 - accuracy: 0.4594
Epoch 70/100
16/16 [==============================] - 5s 299ms/step - loss: 1.7646 - accuracy: 0.4970
Epoch 71/100
16/16 [==============================] - 5s 301ms/step - loss: 2.0501 - accuracy: 0.4183
Epoch 72/100
16/16 [==============================] - 5s 296ms/step - loss: 1.9938 - accuracy: 0.3976
Epoch 73/100
16/16 [==============================] - 5s 290ms/step - loss: 1.7058 - accuracy: 0.4541
Epoch 74/100
16/16 [==============================] - 5s 295ms/step - loss: 1.8786 - accuracy: 0.4562
Epoch 75/100
16/16 [==============================] - 5s 298ms/step - loss: 1.8523 - accuracy: 0.4238
Epoch 76/100
16/16 [==============================] - 5s 297ms/step - loss: 1.8105 - accuracy: 0.4619
Epoch 77/100
16/16 [==============================] - 5s 295ms/step - loss: 1.8691 - accuracy: 0.4482
Epoch 78/100
16/16 [==============================] - 5s 294ms/step - loss: 1.9525 - accuracy: 0.4214
Epoch 79/100
16/16 [==============================] - 5s 301ms/step - loss: 1.9710 - accuracy: 0.4368
Epoch 80/100
16/16 [==============================] - 5s 302ms/step - loss: 1.7961 - accuracy: 0.4764
Epoch 81/100
16/16 [==============================] - 5s 301ms/step - loss: 1.8816 - accuracy: 0.4550
Epoch 82/100
16/16 [==============================] - 5s 292ms/step - loss: 1.8077 - accuracy: 0.4269
Epoch 83/100
16/16 [==============================] - 5s 304ms/step - loss: 1.8586 - accuracy: 0.4822
Epoch 84/100
16/16 [==============================] - 5s 291ms/step - loss: 2.1314 - accuracy: 0.4188
Epoch 85/100
16/16 [==============================] - 5s 291ms/step - loss: 1.8259 - accuracy: 0.4136
Epoch 86/100
16/16 [==============================] - 5s 291ms/step - loss: 1.5769 - accuracy: 0.5159
Epoch 87/100
16/16 [==============================] - 5s 293ms/step - loss: 1.8722 - accuracy: 0.4580
Epoch 88/100
16/16 [==============================] - 5s 289ms/step - loss: 2.0913 - accuracy: 0.4484
Epoch 89/100
16/16 [==============================] - 5s 287ms/step - loss: 1.5975 - accuracy: 0.4999
Epoch 90/100
16/16 [==============================] - 5s 288ms/step - loss: 1.8820 - accuracy: 0.4624
Epoch 91/100
16/16 [==============================] - 5s 289ms/step - loss: 1.6974 - accuracy: 0.4802
Epoch 92/100
16/16 [==============================] - 5s 292ms/step - loss: 1.7181 - accuracy: 0.4816
Epoch 93/100
16/16 [==============================] - 5s 291ms/step - loss: 1.7151 - accuracy: 0.4947
Epoch 94/100
```

```
16/16 [==============================] - 5s 292ms/step - loss: 1.7906 - accuracy: 0.4979
Epoch 95/100
16/16 [==============================] - 5s 292ms/step - loss: 1.8889 - accuracy: 0.4741
Epoch 96/100
16/16 [==============================] - 5s 291ms/step - loss: 1.7664 - accuracy: 0.4660
Epoch 97/100
16/16 [==============================] - 5s 290ms/step - loss: 1.7425 - accuracy: 0.4669
Epoch 98/100
16/16 [==============================] - 5s 289ms/step - loss: 1.7760 - accuracy: 0.4715
Epoch 99/100
16/16 [==============================] - 5s 289ms/step - loss: 1.6330 - accuracy: 0.5097
Epoch 100/100
16/16 [==============================] - 5s 289ms/step - loss: 1.7944 - accuracy: 0.4488
```

In [59]:

```
## Check the prediction

image = load_img('./images/pic_99.jpg', target_size=(224,224))
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
image = preprocess_input(image)

## lets make some prediction on images
pred = neckline_model.predict(image)
pred
```

Out[59]:

```
array([[0.07641989, 0.00867034, 0.08442438, 0.01410546, 0.01769494,
        0.03777191, 0.07707424, 0.01926658, 0.04071048, 0.04149117,
        0.05283599, 0.0656744 , 0.02754457, 0.01252226, 0.1066403 ,
        0.01376635, 0.00873786, 0.06234403, 0.06047474, 0.01703551,
        0.02849907, 0.12629554]], dtype=float32)
```

In [134]:

```
## Predicted class
pred_pat = np.argmax(pred)
ypred = Necklines[pred_pat]

## Actual class
actual = df[df['Image_Path'] == '/images/pic_99.jpg']
yactual = actual['Neckline'].values

print(f'Our model predict {ypred} pattern.... and the real class is {yactual}')
```

```
Our model predict Queen Anne pattern.... and the real class is ['V-Neck']
```

In [135]:

```
## Check the prediction for one more image

image = load_img('./images/pic_349.jpg', target_size=(224,224))
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
image = preprocess_input(image)

## lets make some prediction on images
pred = neckline_model.predict(image)
pred
```

Out[135]:

```
array([[1.1244915e-07, 2.1879015e-17, 2.3026510e-05, 2.5042751e-15,
        2.9573699e-09, 4.7244946e-05, 4.9283535e-08, 1.7620569e-09,
        2.8199355e-07, 2.3770165e-03, 1.8548423e-08, 1.6264644e-07,
        3.2114576e-12, 1.5757820e-12, 4.0524764e-04, 4.8900817e-09,
        4.8291246e-15, 3.4079239e-05, 3.6496171e-08, 3.5404565e-15,
        5.7252145e-09, 9.9711275e-01]], dtype=float32)
```

In [136]:

```
## Predicted class
pred_pat = np.argmax(pred)
ypred = Necklines[pred_pat]

## Actual class
actual = df[df['Image_Path'] == '/images/pic_349.jpg']
yactual = actual['Neckline'].values

print(f'Our model predict {ypred} pattern.... and the real class is {yactual}')
```

```
Our model predict V-Neck pattern.... and the real class is ['V-Neck']
```

In [139]:

```
## Save the model
neckline_model.save('Neckline_model.h5')
```

In [ ]:

```
```

Try to predict the above categories for external image

In [170]:

```
# !wget https://m.media-amazon.com/images/I/614qPMvTCRL._UL1500_.jpg -O newimg.jpg
# !wget https://i.pinimg.com/originals/4b/16/31/4b1631e6efc54a7f1fe44982fff89e87.jpg -O n
ewimg2.jpg
```

```
--2020-12-22 14:48:43--  https://i.pinimg.com/originals/4b/16/31/4b1631e6efc54a7f1fe44982
fff89e87.jpg
Resolving i.pinimg.com (i.pinimg.com)... 104.18.14.176, 104.18.15.176, 2a04:4e42:a::84
Connecting to i.pinimg.com (i.pinimg.com)|104.18.14.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 76632 (75K) [image/jpeg]
Saving to: 'newimg2.jpg'

newimg2.jpg         100%[===================>]  74.84K  --.-KB/s    in 0.008s

2020-12-22 14:48:43 (9.54 MB/s) - 'newimg2.jpg' saved [76632/76632]
```

In [162]:

```
image = load_img('newimg.jpg', target_size=(224,224))
image
```

Out[162]:

In [165]:

```
## convert the image in machine readable format
image = load_img('newimg.jpg', target_size=(224,224))
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
image = preprocess_input(image)
```

In [168]:

```
## lets make some prediction on images
pred_mat = material_model.predict(image)
pred_pat = pattern_model.predict(image)
pred_nec = neckline_model.predict(image)


## Predicted class for material
pred_Mat = np.argmax(pred_mat)
pred_Mat = Materials[pred_Mat]
## predicted class for Pattern
pred_Pat = np.argmax(pred_pat)
pred_Pat = Patterns[pred_Pat]
## Predicted class for Neckline
pred_Nec = np.argmax(pred_nec)
pred_Nec = Necklines[pred_Nec]
```

In [169]:

```
print(f"Therefore, the Material, Pattern and Necline for the above downloaded image is: {
pred_Mat} {pred_Pat} {pred_Nec}")
```

```
Therefore, the Material, Pattern and Necline for the above downloaded image is: Cotton So
lid/Plain Hooded
```

In [ ]:

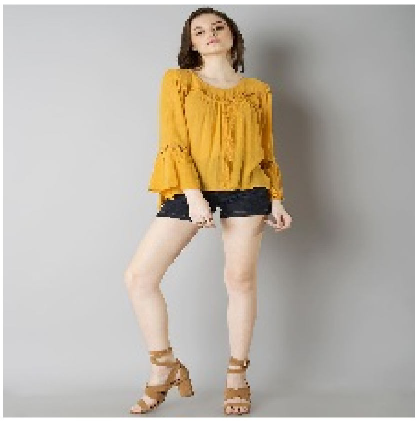
```

```

In [171]:

```
image = load_img('newimg2.jpg', target_size=(224,224))
image
```

Out[171]:



In [172]:

```
## convert the image in machine readable format
image = load_img('newimg2.jpg', target_size=(224,224))
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
image = preprocess_input(image)
```

In [173]:

```
## lets make some prediction on images
pred_mat = material_model.predict(image)
pred_pat = pattern_model.predict(image)
pred_nec = neckline_model.predict(image)


## Predicted class for material
pred_Mat = np.argmax(pred_mat)
pred_Mat = Materials[pred_Mat]
## predicted class for Pattern
```

```python
pred_Pat = np.argmax(pred_pat)
pred_Pat = Patterns[pred_Pat]
## Predicted class for Neckline
pred_Nec = np.argmax(pred_nec)
pred_Nec = Necklines[pred_Nec]
```

In [174]:

```python
print(f"Therefore, the Material, Pattern and Necline for the above downloaded image is: {
pred_Mat} {pred_Pat} {pred_Nec}")
```

```
Therefore, the Material, Pattern and Necline for the above downloaded image is: Polyester
Solid/Plain Shoulder Straps
```

In [ ]:

```python

```