1.Import Nessasary Libraries

1.Import Libraries

```
from keras import datasets
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical
from keras.layers import Dense,Flatten,MaxPooling2D,Conv2D,Dropout
from keras.models import Sequential,Model
import warnings
warnings.filterwarnings("ignore")   #Import Nessasary all the Library
```

2.Import Datasets

```
(x_train, y_train), (x_test, y_test)=tf.keras.datasets.fashion_mnist.load_data() # Load Data
```

```
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-
    32768/29515 [=================================] - 0s 0us/step
    40960/29515 [=========================================] - 0s 0us/step
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-
    26427392/26421880 [==============================] - 0s 0us/step
    26435584/26421880 [==============================] - 0s 0us/step
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-l
    16384/5148 [===========================================================================
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-i
    4423680/4422102 [==============================] - 0s 0us/step
    4431872/4422102 [==============================] - 0s 0us/step
```

3.Data Preparing

```
x_train=np.pad(x_train,((0,0),(2,2),(2,2)))
x_test=np.pad(x_test,((0,0),(2,2),(2,2)))  #Add Padd to get dimension of 32*32
```

```
x_train=np.reshape(x_train,newshape=(60000,32,32,1))
x_test=np.reshape(x_test,newshape=(10000,32,32,1))  # Reshape datasets and add input channel
```

```
x_train=np.pad(x_train,((0,0),(0,0),(0,0),(1,1)))
x_test=np.pad(x_test,((0,0),(0,0),(0,0),(1,1)))  #We need add Channel of 3 because image is d
```

```
x_train.shape,x_test.shape   #check input datashape
```

```
    ((60000, 32, 32, 3), (10000, 32, 32, 3))
```

we have 10 diffrent output so we use to_catagory to sepate them in diffrent catagory

```
y_train=to_categorical(y_train,10)
y_test=to_categorical(y_test,10)
```

1.Vgg16

1.Import Library

```
from keras.applications.vgg16 import VGG16
```

2.Import Dataset

```
vgg_model=VGG16(include_top=False , weights='imagenet', input_tensor=None, input_shape=(32,32
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg1
58892288/58889256 [==============================] - 1s 0us/step
58900480/58889256 [==============================] - 1s 0us/step
```

```
for layer in vgg_model.layers:
    layer.trainable=False   #We are making all pre trainable layers to Flase
```

```
c=vgg_model.get_config()
c['layers'][0]['config']['batch_input_shape']=(None,32,32,3)
resnet_model=Model.from_config(c)                   #Reshape input Datasets
```

```
x=Flatten()(vgg_model.output)
prediction=Dense(10,activation='softmax')(x)    #update all weight and bias for model and add
```

```
model=Model(inputs=vgg_model.input,outputs=prediction)
```

3. Compile Model

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

4.Model Training

```
model.fit(x_train,y_train,verbose=10,epochs=10,validation_data=(x_test,y_test))   #Fit Model
```

```
Epoch 1/10
Epoch 2/10
Epoch 3/10
Epoch 4/10
Epoch 5/10
Epoch 6/10
Epoch 7/10
Epoch 8/10
Epoch 9/10
Epoch 10/10
<keras.callbacks.History at 0x7f44601f0bd0>
```

5. Model Evalution

```
score=model.evaluate(x_test,y_test)
print('accuracy:%2.f%%'%(score[1]*100))   #Check accuracy
```

```
313/313 [==============================] - 6s 19ms/step - loss: 0.5354 - accuracy: 0.82
accuracy:83%
```

2.Vgg19

1.Import Library

```
from keras.applications.vgg19 import VGG19   #import Library
```

2.Import Dataset

```
vgg19_model=tf.keras.applications.VGG19( include_top=False,
                                         weights="imagenet",
                                         input_shape=(32,32,3),
                                        classes=1000)            #Import pretrain applicatio
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg1
80142336/80134624 [==============================] - 1s 0us/step
80150528/80134624 [==============================] - 1s 0us/step
```

```
for layer in vgg19_model.layers:
  layer.trainable=False           # False all the previous Layers
```

```
x=Flatten()(vgg19_model.output)
prediction=Dense(10,activation="softmax")(x)     #Add nessasary layers
```

```
model=Model(inputs=vgg19_model.input,outputs=prediction)
```

4Model Compilation

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

5.Model Training

```
model.fit(x_train,y_train,verbose=10,epochs=1,validation_data=(x_test,y_test))
```

```
<keras.callbacks.History at 0x7f43e8cd9f90>
```

6.Model Evaluation

```
score=model.evaluate(x_test,y_test)
print("accuracy:%2.f%%"%(score[1]*100))
```

```
313/313 [==============================] - 7s 23ms/step - loss: 0.6028 - accuracy: 0.81
accuracy:81%
```

3.ResNet50

1.Import Librarie

```
from keras.applications.resnet import ResNet50   #Import Library
```

2.Import Dataset

```
resnet_model=tf.keras.applications.ResNet50(include_top=False,
                                            weights="imagenet",
                                            input_tensor=None,
                                            input_shape=(32,32,3),
                                            pooling=None,
                                            classes=1000,)     #Pretrain model with Flase all
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resne
94773248/94765736 [==============================] - 1s 0us/step
94781440/94765736 [==============================] - 1s 0us/step
```

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

```
x=Flatten()(resnet_model.output)
prediction=Dense(10,activation='softmax')(x)
```

```
model=Model(inputs=resnet_model.input,outputs=prediction)
```

3.Compile the Model

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

4.Model Training

```
model.fit(x_train,y_train,verbose=1,epochs=10,validation_data=(x_test,y_test))  #fit the mode
```

```
Epoch 1/10
1875/1875 [==============================] - 63s 32ms/step - loss: 0.5950 - accuracy: 0
Epoch 2/10
1875/1875 [==============================] - 55s 29ms/step - loss: 0.4749 - accuracy: 0
Epoch 3/10
1875/1875 [==============================] - 55s 30ms/step - loss: 0.4388 - accuracy: 0
Epoch 4/10
1875/1875 [==============================] - 55s 30ms/step - loss: 0.4177 - accuracy: 0
Epoch 5/10
1875/1875 [==============================] - 58s 31ms/step - loss: 0.4089 - accuracy: 0
Epoch 6/10
1875/1875 [==============================] - 55s 29ms/step - loss: 0.3954 - accuracy: 0
Epoch 7/10
1875/1875 [==============================] - 55s 29ms/step - loss: 0.3839 - accuracy: 0
Epoch 8/10
1875/1875 [==============================] - 55s 29ms/step - loss: 0.3859 - accuracy: 0
Epoch 9/10
1875/1875 [==============================] - 55s 29ms/step - loss: 0.3744 - accuracy: 0
Epoch 10/10
1875/1875 [==============================] - 55s 29ms/step - loss: 0.3670 - accuracy: 0
<keras.callbacks.History at 0x7f43e89fc610>
```

5.Model Evalution

```
score=model.evaluate(x_test,y_test)
print("accuracy : % 2.f %%"%(score[1]*100))
```

```
313/313 [==============================] - 8s 27ms/step - loss: 0.5301 - accuracy: 0.848
accuracy :  85 %
```

4.Resnet50V2

1.Import libraries

```
from keras.applications.resnet_v2 import ResNet50V2
```

2.Import Dataset

```
resnetv2_model=tf.keras.applications.ResNet50V2(include_top=False,
                                                weights="imagenet",
                                                    input_shape=(32,32,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resne
94674944/94668760 [==============================] - 1s 0us/step
94683136/94668760 [==============================] - 1s 0us/step
```

```
for layer in resnetv2_model.layers:
    layer.trainable=False
```

```
x=Flatten()(resnetv2_model.output)
prediction=Dense(10,activation='relu')(x)
```

```
model=Model(inputs=resnetv2_model.input,outputs=prediction)
```

3.Model Compiling

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

4.Model Training

```
model.fit(x_train,y_train,verbose=1,epochs=10,validation_data=(x_test,y_test))  #Fit the Mode
```

```
Epoch 1/10
1875/1875 [==============================] - 52s 28ms/step - loss: nan - accuracy: 0.417
Epoch 2/10
1875/1875 [==============================] - 52s 28ms/step - loss: nan - accuracy: 0.418
Epoch 3/10
1875/1875 [==============================] - 52s 28ms/step - loss: nan - accuracy: 0.421
Epoch 4/10
1875/1875 [==============================] - 52s 28ms/step - loss: nan - accuracy: 0.426
Epoch 5/10
1875/1875 [==============================] - 52s 28ms/step - loss: nan - accuracy: 0.419
Epoch 6/10
1875/1875 [==============================] - 52s 28ms/step - loss: nan - accuracy: 0.422
Epoch 7/10
1875/1875 [==============================] - 52s 28ms/step - loss: nan - accuracy: 0.422
Epoch 8/10
1875/1875 [==============================] - 52s 28ms/step - loss: nan - accuracy: 0.421
Epoch 9/10
1875/1875 [==============================] - 52s 28ms/step - loss: nan - accuracy: 0.413
Epoch 10/10
```

```
1875/1875 [==============================] - 52s 28ms/step - loss: nan - accuracy: 0.42
<keras.callbacks.History at 0x7f4378978110>
```

5.Model Evalution

```
score=model.evaluate(x_test,y_test)
print("accuracy·%2.f%%·"%(score[1]*100))··#check·accuracy·We·are·Getting·from·Model
```

```
313/313 [==============================] - 8s 26ms/step - loss: nan - accuracy: 0.4204
accuracy 42%
```

5.CNN architecture

1.Import Dataset

```
(x_train, y_train), (x_test, y_test)=tf.keras.datasets.fashion_mnist.load_data() # Load Data
```

2. Check shape of train and test data

```
x_train.shape,y_train.shape,x_test.shape,y_test.shape   #check shape of input and output data
```

```
((60000, 28, 28), (60000,), (10000, 28, 28), (10000,))
```

```
x_train.std(),x_test.std()
```

```
(90.02118235130519, 89.87325907809718)
```

3.Normlize Input Datasets

```
x_train=x_train/255
x_test=x_test/255   # Normlize input Data
```

```
x_train=np.pad(x_train,pad_width=((0,0),(2,2),(2,2)))
x_test=np.pad(x_test,pad_width=((0,0),(2,2),(2,2)))   #Add Pad to all dimention
```

```
x_train.shape,x_test.shape   #check shape of input and output
```

```
((60000, 32, 32), (10000, 32, 32))
```

4.Reshape input Datasets

```
x_train_reshape=np.reshape(x_train,newshape=(60000,32,32,1))
x_test_reshape=np.reshape(x_test,newshape=(10000,32,32,1))   #reshape input data
```

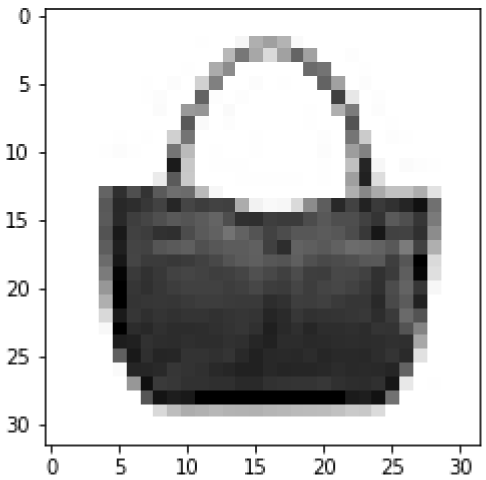
5.Add catagorical to output to convert output in catagory

```
y_train_encoder=to_categorical(y_train)
y_test_encoder=to_categorical(y_test)
```

6.Data Testing

```
plt.imshow(x_train[100],cmap="Greys")    #Test the data
```

```
<matplotlib.image.AxesImage at 0x7f437891ff10>
```



```
y_train[100]
```

```
8
```

7.Model Building

```
model=Sequential()
model.add(Conv2D(input_shape=(32,32,1),strides=1,filters=16, kernel_size=(3,3), padding='vali
model.add(MaxPooling2D(pool_size=(2,2),strides=2))
model.add(Conv2D(strides=1,filters=32, kernel_size=(3,3), padding='valid',kernel_initializer=
model.add(MaxPooling2D(pool_size=(2,2),strides=2))
model.add(Conv2D(strides=1,filters=32, kernel_size=(3,3), padding='valid',kernel_initializer=

model.add(Flatten())
model.add(Dense(100,activation='relu'))
model.add(Dense(40,activation='relu'))
model.add(Dense(10,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 16)        160

 max_pooling2d_3 (MaxPooling  (None, 15, 15, 16)       0
 2D)

 conv2d_1 (Conv2D)           (None, 13, 13, 32)        4640

 max_pooling2d_4 (MaxPooling  (None, 6, 6, 32)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 32)          9248

 flatten_4 (Flatten)         (None, 512)               0

 dense_4 (Dense)             (None, 100)               51300

 dense_5 (Dense)             (None, 40)                4040

 dense_6 (Dense)             (None, 10)                410

=================================================================
Total params: 69,798
Trainable params: 69,798
```

```
Non-trainable params: 0
_________________________________________________________________
```

8.Model Compiling

```
model.compile(optimizer='adam',metrics="categorical_accuracy",loss='categorical_crossentropy'
```

9.model Training

```
model_training=model.fit(x=x_train_reshape, y=y_train_encoder,batch_size=32,epochs=10,verbose
```

```
Epoch 1/10
1875/1875 [==============================] - 12s 6ms/step - loss: 0.5751 - categorical_a
Epoch 2/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3522 - categorical_a
Epoch 3/10
1875/1875 [==============================] - 12s 6ms/step - loss: 0.2945 - categorical_a
Epoch 4/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2628 - categorical_a
Epoch 5/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2401 - categorical_a
Epoch 6/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2222 - categorical_a
Epoch 7/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2063 - categorical_a
Epoch 8/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1928 - categorical_a
Epoch 9/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1823 - categorical_a
Epoch 10/10
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1702 - categorical_a
```

10.Model evalution

```
model.evaluate(x_test_reshape,y_test_encoder)  # Check Accuracy of the model
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.2660 - categorical_accu
[0.2660037875175476, 0.909500002861023]
```

11.Model Testing

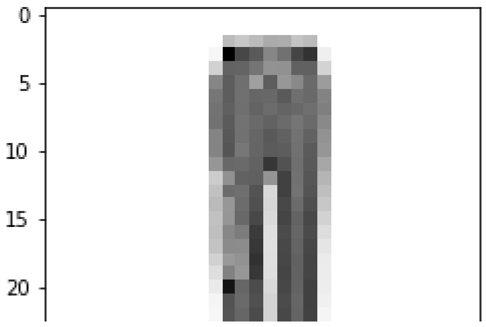
```
y_test_pred=model.predict(x_test)
```

```
class_name=['T-shirt/top',"Trouser","Pullover","Dress","Coat","Sandal",'Shirt','Sneaker','Bag
```

```
#We are taking example 100th data of x_train 100 image is showing bag
```

Example 1

```
plt.imshow(x_train[1000],cmap='Greys')   #Randomly taking 1000th observation of train dataset
plt.show()
```

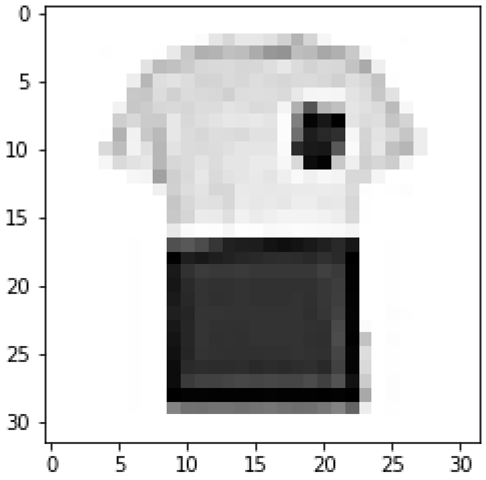


```
class_name[y_train[1000]]     #Prediction
```

```
'Trouser'
```

Example 2

```
plt.imshow(x_test[120],cmap="Greys")
plt.show()
```



```
class_name[np.argmax(y_test_pred[120])]   #argmax trigger preidcted neuron
```

```
'T-shirt/top'
```

```
#  Model      Accuracy
# 1.vggg16      83%
# 2.Vgg19       81%
# 3.Resnet50    85%
# 4.Resnet50v2  42%
# 5.Cnn Model   90%
```

✓ 0s completed at 10:44 PM