ABSTARCT for the Project Fashion MNIST Data Classifier

In this project, we have built a fashion apparel recognition using the Convolutional Neural Network (CNN) model. To train the CNN model, we have used the Fashion MNIST dataset. After successful training, the CNN model can predict the name of the class given apparel item belongs to. This is a multiclass classification problem in which there are 10 apparel classes the items will be classified.

The fashion training set consists of 70,000 images divided into 60,000 training and 10,000 testing samples. Dataset sample consists of 28x28 grayscale images, associated with a label from 10 classes.

So the end goal is to train and test the model using Convolution neural network

OBJECTIVE:-

One of the classic problem that has been used in the Machine Learning world for quite sometime is the MNIST problem. The objective is to identify the digit based on image. But MNIST is not very great problem because we come up with great accuracy even if we are looking at few pixels in the image. So, another common example problem against which we test algorithms is Fashion-MNIST.

This work is part of my experiments with Fashion-MNIST dataset using various Machine Learning algorithms/models. The objective is to identify (predict) different fashion products from the given images using various best possible Machine Learning Models (Algorithms) and compare their results (performance measures/scores) to arrive at the best ML model. I have also experimented with 'dimensionality reduction' technique for this problem.

METHODOLOGY:-

The **Fashion MNIST** dataset was developed as a response to the wide use of the **MNIST dataset**, that has been effectively "**solved**" given the use of modern convolutional neural networks.
Fashion-MNIST was proposed to be a replacement for MNIST, and although it has not been solved, it is possible to routinely achieve error rates of 10% or less. Like MNIST, it can be a useful starting point for developing and practicing a methodology for solving image classification using convolutional neural networks.

Instead of reviewing the literature on well-performing models on the dataset, we can develop a new model from scratch.
The dataset already has a well-defined train and test dataset that we can use.
In order to estimate the performance of a model for a given training run, we can further split the training set into a train and validation dataset. Performance on the train and validation dataset over each run can then be plotted to provide learning curves and insight into how well a model is learning the problem.
The Keras API supports this by specifying the "**validation_data"** argument to the "**model.fit()"** function when training the model, that will, in turn, return an object that describes model performance for the chosen loss and metrics on each training epoch.

CODE:-

Fashion MNIST Data Classification Project

Step 1) Import Libraries

```
import matplotlib.pyplot as plt
import numpy as np
import keras
import seaborn as sns
import tensorflow as tf
```

Step 2) Load data

```
(X_train, y_train), (X_test, y_test)=tf.keras.datasets.fashion_m
nist.load_data()


# Print the shape of data
X_train.shape,y_train.shape, "************" , X_test.shape,y_tes
t.shape

X_train[0]


y_train[0]


class_labels = ["T-
shirt/top","Trouser","Pullover","Dress","Coat","Sandal","Shirt",
"Sneaker","Bag",'Ankel boot']

# show image



plt.imshow(X_train[0],cmap='Greys')

plt.figure(figsize=(16,16))

j=1
for  i in np.random.randint(0,1000,25):
  plt.subplot(5,5,j);j+=1
  plt.imshow(X_train[i],cmap='Greys')
  plt.axis('off')
  plt.title('{} / {}'.format(class_labels[y_train[i]],y_train[i]
))

X_train.ndim


X_train = np.expand_dims(X_train,-1)

X_train.ndim

X_test=np.expand_dims(X_test,-1)
```

Feature Scalling

```
X_train = X_train/255
X_test = X_test/255
```

Split dataset

```
from sklearn.model_selection import  train_test_split
X_train,X_Validation,y_train,y_Validation=train_test_split(X_tra
in,y_train,test_size=0.2,random_state=2020)
```

```
X_train.shape,X_Validation.shape,y_train.shape,y_Validation.shap
e
```

Step 3) Building the CNN model

```
model=keras.models.Sequential([
                         keras.layers.Conv2D(filters=32,kernel_s
ize=3,strides=(1,1),padding='valid',activation='relu',input_shap
e=[28,28,1]),
                         keras.layers.MaxPooling2D(pool_size=(2,
2)),
                         keras.layers.Flatten(),
                         keras.layers.Dense(units=128,activation
='relu'),
                         keras.layers.Dense(units=10,activation=
'softmax')
])
```

```
model.summary()
```

Compile the Model

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

Train the model

```
model.fit(X_train,y_train,epochs=10,batch_size=512,verbose=1,val
idation_data=(X_Validation,y_Validation))
```

Test and Evaluate Neural Network Model

```
y_pred = model.predict(X_test)
y_pred.round(2)

y_test

model.evaluate(X_test, y_test)


plt.figure(figsize=(16,16))

j=1
for i in np.random.randint(0, 1000,25):
  plt.subplot(5,5, j); j+=1
  plt.imshow(X_test[i].reshape(28,28), cmap = 'Greys')
  plt.title('Actual = {} / {} \nPredicted = {} / {}'.format(clas
s_labels[y_test[i]], y_test[i], class_labels[np.argmax(y_pred[i]
)],np.argmax(y_pred[i])))
  plt.axis('off')
```

```
plt.figure(figsize=(16,30))

j=1
for i in np.random.randint(0, 1000,60):
  plt.subplot(10,6, j); j+=1
  plt.imshow(X_test[i].reshape(28,28), cmap = 'Greys')
  plt.title('Actual = {} / {} \nPredicted = {} / {}'.format(clas
s_labels[y_test[i]], y_test[i], class_labels[np.argmax(y_pred[i]
)],np.argmax(y_pred[i])))
  plt.axis('off')
```

Confusion Matrix And Classification Report

```
from sklearn.metrics import confusion_matrix
plt.figure(figsize=(16,9))
y_pred_labels = [ np.argmax(label) for label in y_pred ]
cm = confusion_matrix(y_test, y_pred_labels)

sns.heatmap(cm, annot=True, fmt='d',xticklabels=class_labels, yt
icklabels=class_labels)

from sklearn.metrics import classification_report
cr= classification_report(y_test, y_pred_labels, target_names=cl
ass_labels)
print(cr)
```

Classification Report

```
from sklearn.metrics import classification_report
cr = classification_report(y_test, [np.argmax(i) for i in y_pred
],target_names=class_labels,)
```

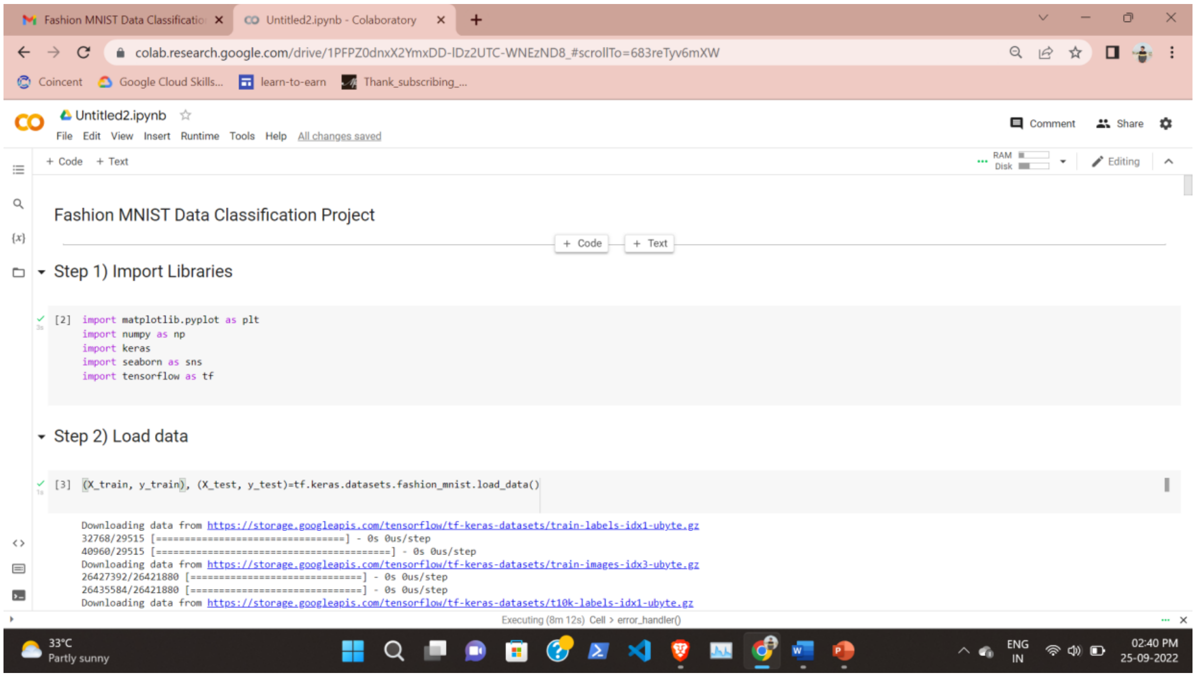
Save Model

```
model.save("Fashion_cnn_model.h5")
model = keras.models.load_model("Fashion_cnn_model.h5")
```

Build 2 complex CNN

```
#Building CNN model
cnn_model2 = keras.models.Sequential([
                         keras.layers.Conv2D(filters=32, kernel_size=3, st
rides=(1,1), padding='valid',activation= 'relu', input_shape=[28,28,1]),
                         keras.layers.MaxPooling2D(pool_size=(2,2)),
                         keras.layers.Conv2D(filters=64, kernel_size=3, st
rides=(2,2), padding='same', activation='relu'),
                         keras.layers.MaxPooling2D(pool_size=(2,2)),
                         keras.layers.Flatten(),
                         keras.layers.Dense(units=128, activation='relu'),
                         keras.layers.Dropout(0.25),
                         keras.layers.Dense(units=256, activation='relu'),
                         keras.layers.Dropout(0.25),
                         keras.layers.Dense(units=128, activation='relu'),
                         keras.layers.Dense(units=10, activation='softmax'
)
                         ])

# complie the model
cnn_model2.compile(optimizer='adam', loss= 'sparse_categorical_crossentrop
y', metrics=['accuracy'])

#Train the Model
cnn_model2.fit(X_train, y_train, epochs=20, batch_size=512, verbose=1, val
idation_data=(X_Validation, y_Validation))

cnn_model2.save("Fashion_cnn_model.h5")

"""######## very complex model"""

#Building CNN model
cnn_model3 = keras.models.Sequential([
                         keras.layers.Conv2D(filters=64, kernel_size=3, st
rides=(1,1), padding='valid',activation= 'relu', input_shape=[28,28,1]),
                         keras.layers.MaxPooling2D(pool_size=(2,2)),
                         keras.layers.Conv2D(filters=128, kernel_size=3, s
trides=(2,2), padding='same', activation='relu'),
                         keras.layers.MaxPooling2D(pool_size=(2,2)),
                         keras.layers.Conv2D(filters=64, kernel_size=3, st
rides=(2,2), padding='same', activation='relu'),
                         keras.layers.MaxPooling2D(pool_size=(2,2)),
                         keras.layers.Flatten(),
                         keras.layers.Dense(units=128, activation='relu'),
```

```
                              keras.layers.Dropout(0.25),
                              keras.layers.Dense(units=256, activation='relu'),
                              keras.layers.Dropout(0.5),
                              keras.layers.Dense(units=256, activation='relu'),
                              keras.layers.Dropout(0.25),

                              keras.layers.Dense(units=128, activation='relu'),
                              keras.layers.Dropout(0.10),

                              keras.layers.Dense(units=10, activation='softmax'
)
                              ])

# complie the model
cnn_model3.compile(optimizer='adam', loss= 'sparse_categorical_crossentrop
y', metrics=['accuracy'])

#Train the Model
cnn_model3.fit(X_train, y_train, epochs=50, batch_size=512, verbose=1, val
idation_data=(X_Validation, y_Validation))

cnn_model3.save("Fashion_cnn_model.h5")

cnn_model3.evaluate(X_test, y_test)
```



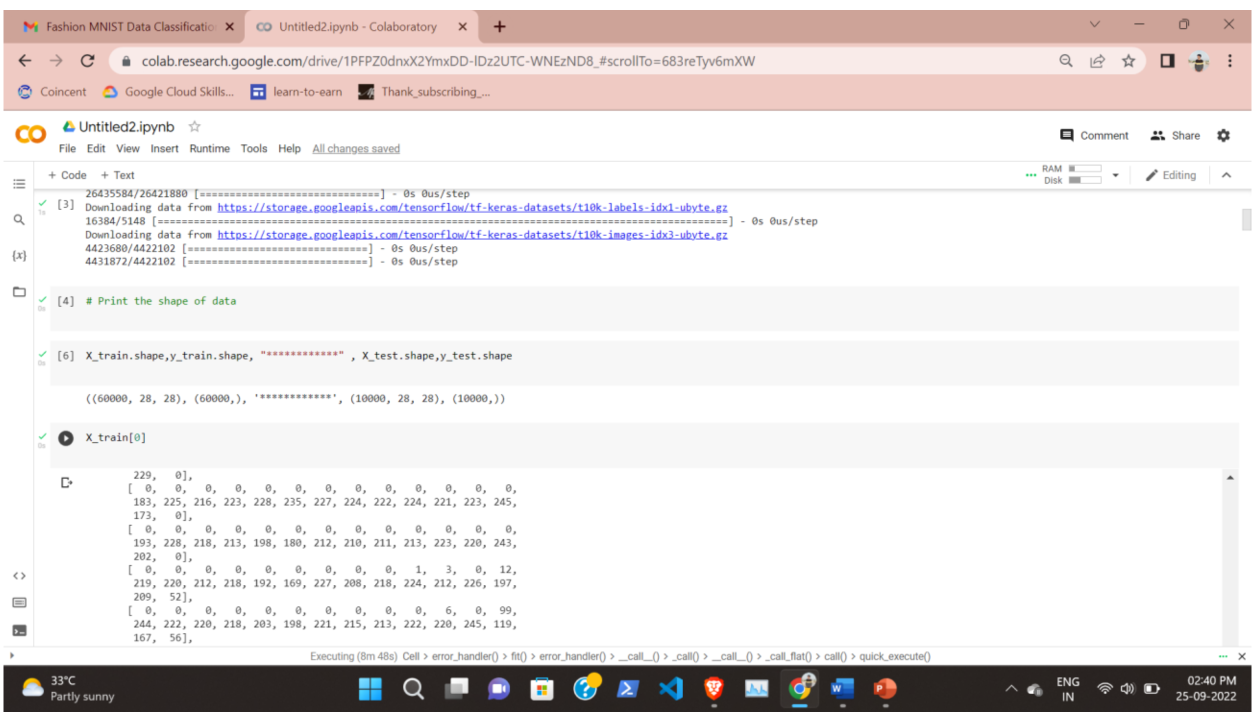
Fashion MNIST Data Classification Project

Step 1) Import Libraries

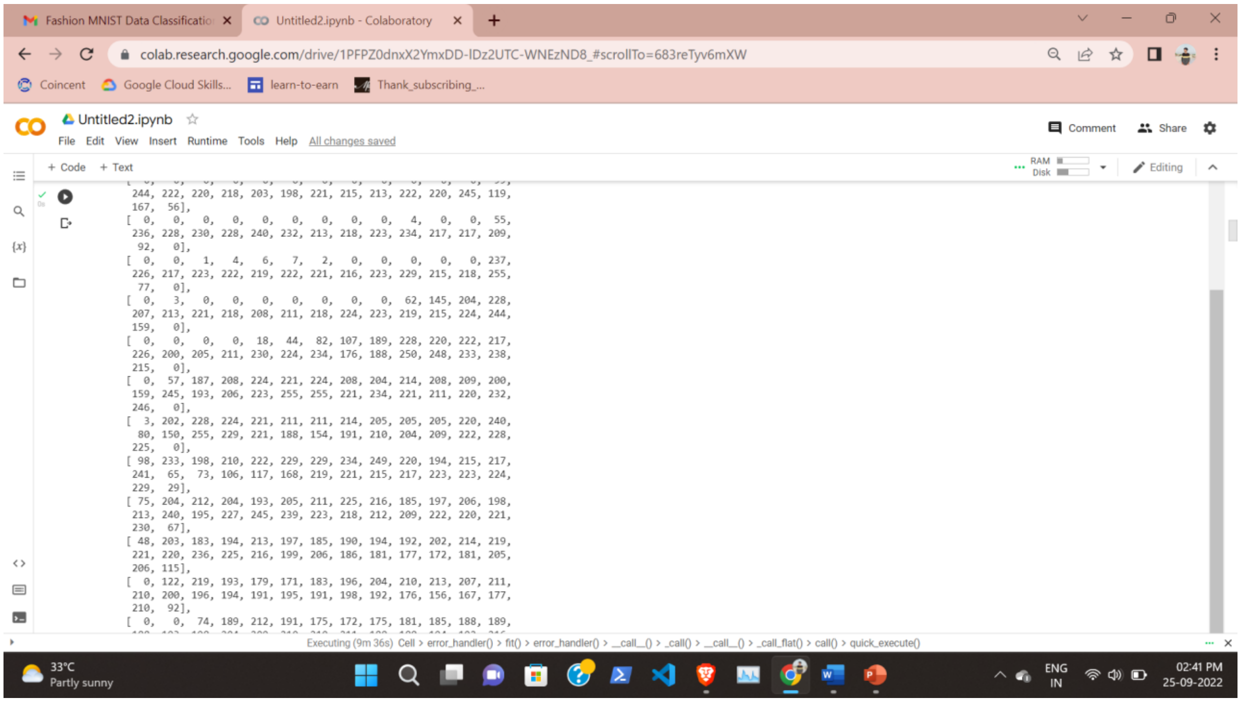
```
[2] import matplotlib.pyplot as plt
    import numpy as np
    import keras
    import seaborn as sns
    import tensorflow as tf
```

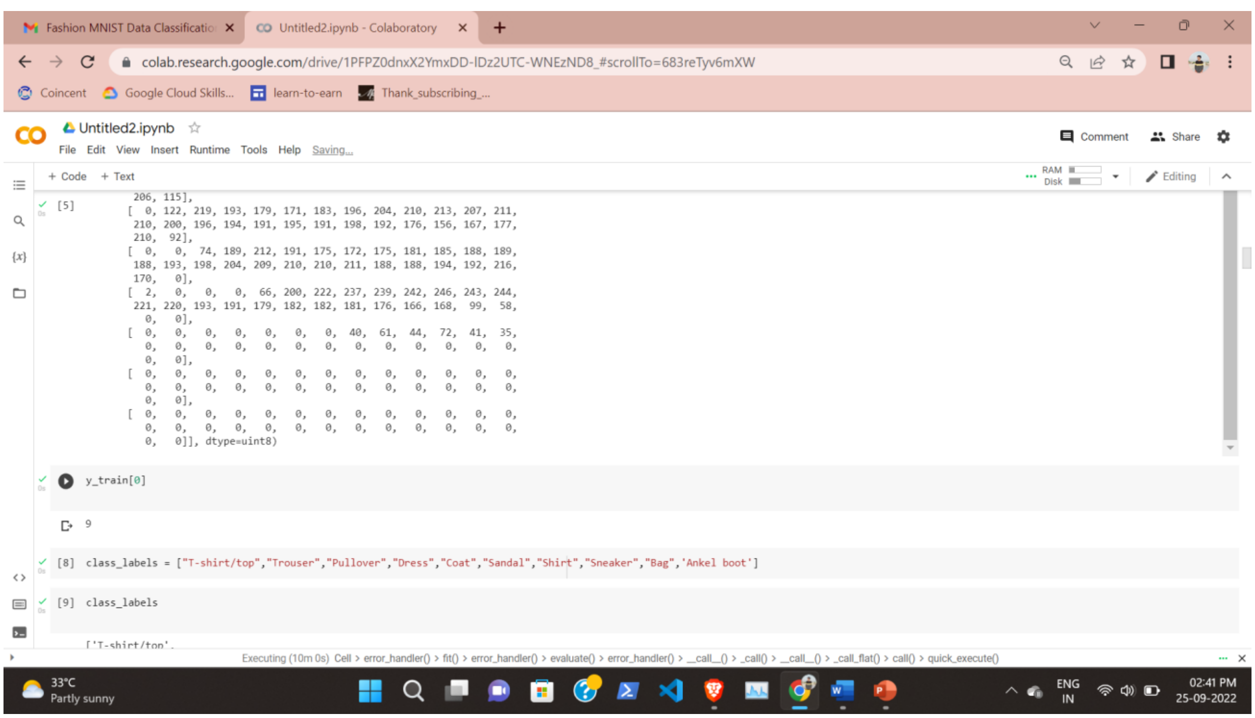
Step 2) Load data

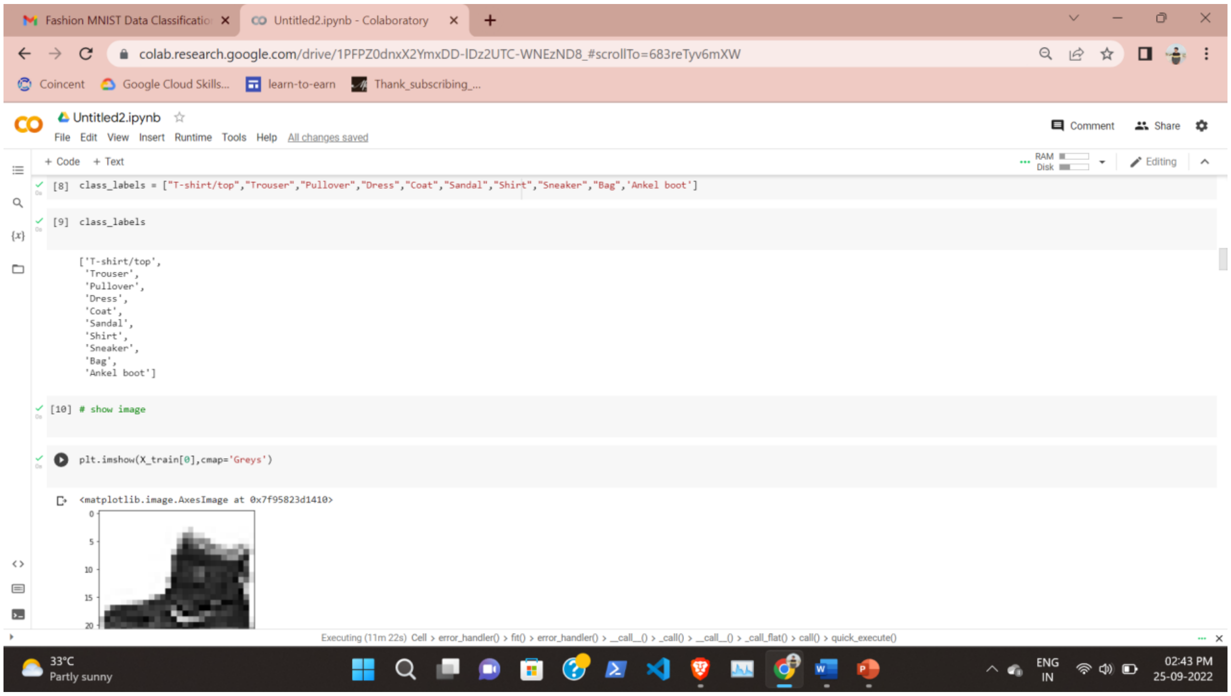
```
[3] (X_train, y_train), (X_test, y_test)=tf.keras.datasets.fashion_mnist.load_data()

    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
    32768/29515 [==============================] - 0s 0us/step
    40960/29515 [=======================================] - 0s 0us/step
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
    26427392/26421880 [==============================] - 0s 0us/step
    26435584/26421880 [==============================] - 0s 0us/step
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
```

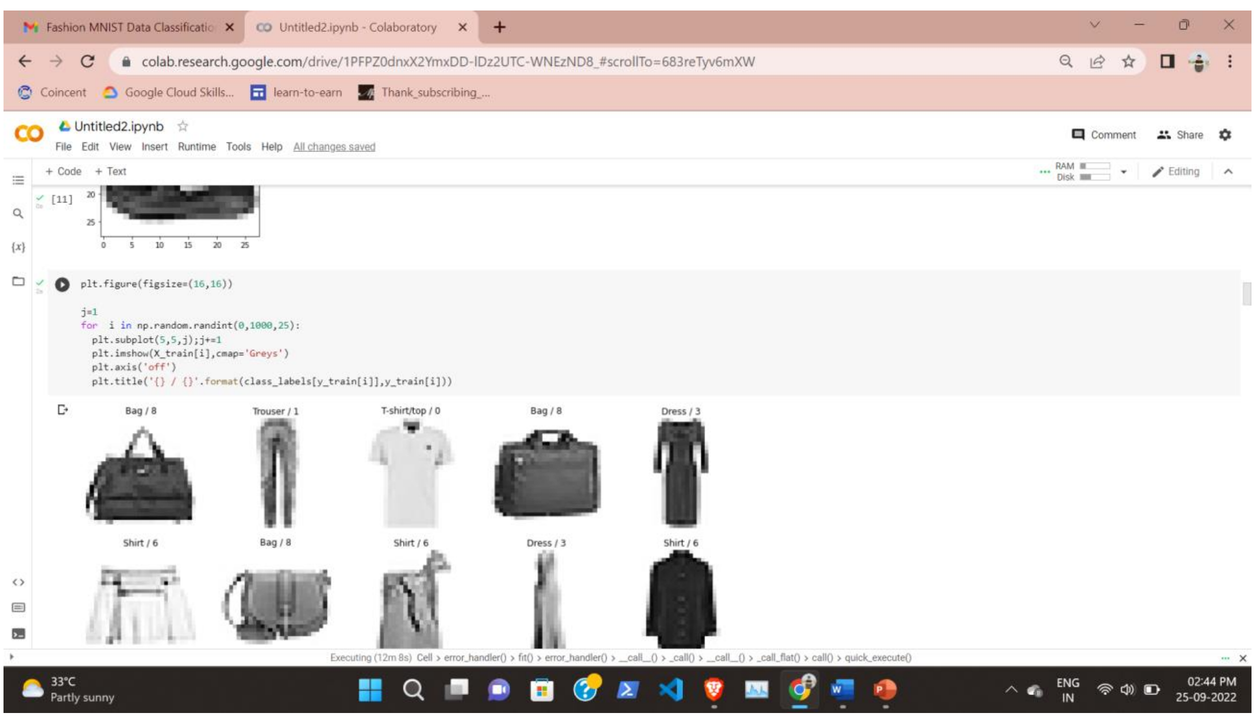


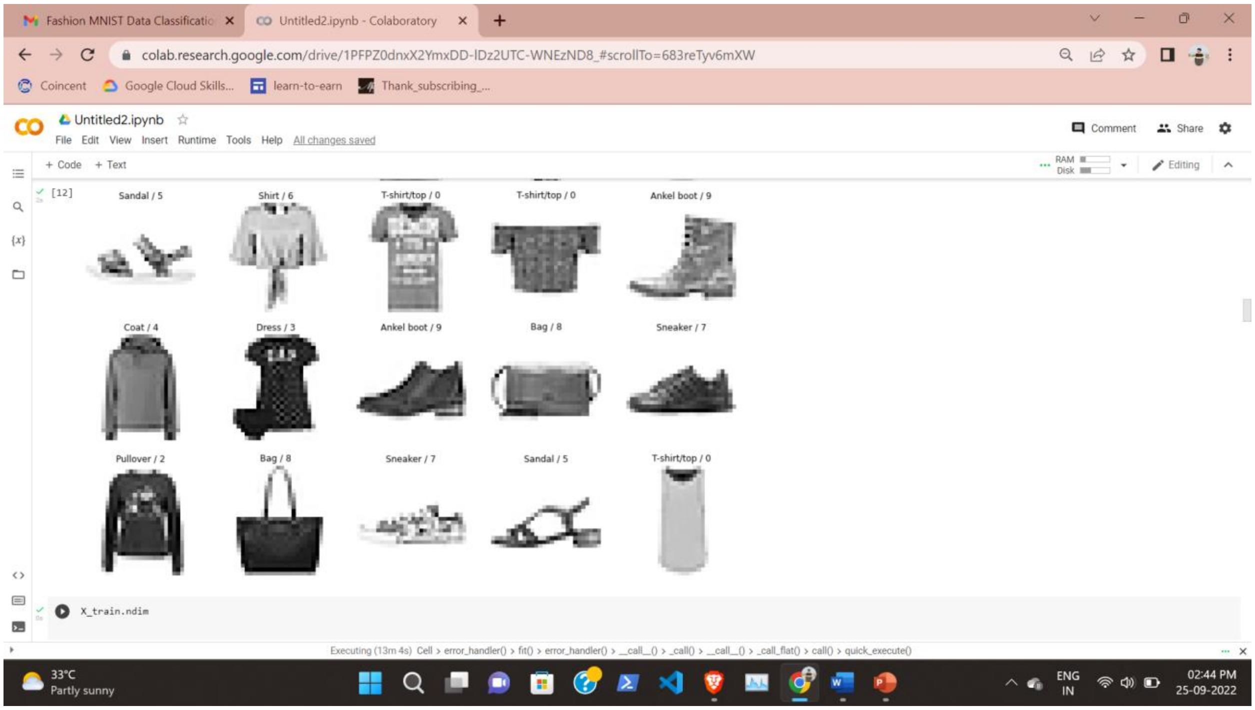
```
    26435584/26421880 [==============================] - 0s 0us/step
[3] Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
    16384/5148 [==========================================================================================] - 0s 0us/step
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
    4423680/4422102 [==============================] - 0s 0us/step
    4431872/4422102 [==============================] - 0s 0us/step

[4] # Print the shape of data

[6] X_train.shape,y_train.shape, "************" , X_test.shape,y_test.shape

    ((60000, 28, 28), (60000,), '************', (10000, 28, 28), (10000,))

    X_train[0]

         229,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         183, 225, 216, 223, 228, 235, 227, 224, 222, 224, 221, 223, 245,
         173,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         193, 228, 218, 213, 198, 180, 212, 210, 211, 213, 223, 220, 243,
         202,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   1,   3,   0,  12,
         219, 220, 212, 218, 192, 169, 227, 208, 218, 224, 212, 226, 197,
         209,  52],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   6,   0,  99,
         244, 222, 220, 218, 203, 198, 221, 215, 213, 222, 220, 245, 119,
         167,  56],
```

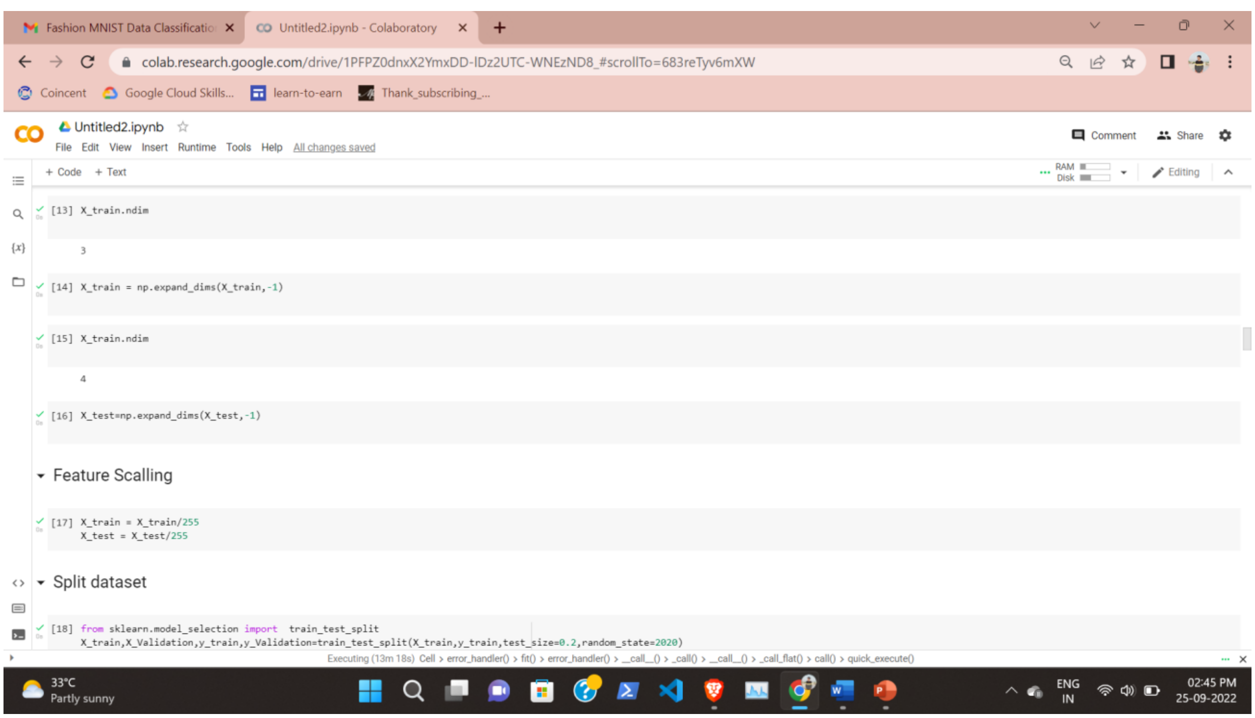

Fashion MNIST Data Classificatio
Untitled2.ipynb - Colaboratory
Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
244, 222, 220, 218, 203, 198, 221, 215, 213, 222, 220, 245, 119,
167, 56],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 55,
236, 228, 230, 228, 240, 232, 213, 218, 223, 234, 217, 217, 209,
92, 0],
[0, 0, 1, 4, 6, 7, 2, 0, 0, 0, 0, 0, 237,
226, 217, 223, 222, 219, 222, 221, 216, 223, 229, 215, 218, 255,
77, 0],
[0, 3, 0, 0, 0, 0, 0, 0, 0, 62, 145, 204, 228,
207, 213, 221, 218, 208, 211, 218, 224, 223, 219, 215, 224, 244,
159, 0],
[0, 0, 0, 0, 18, 44, 82, 107, 189, 228, 220, 222, 217,
226, 200, 205, 211, 230, 224, 234, 176, 188, 250, 248, 233, 238,
215, 0],
[0, 57, 187, 208, 224, 221, 224, 208, 204, 214, 208, 209, 200,
159, 245, 193, 206, 223, 255, 255, 221, 234, 221, 211, 220, 232,
246, 0],
[3, 202, 228, 224, 221, 211, 211, 214, 205, 205, 205, 220, 240,
80, 150, 255, 229, 221, 188, 154, 191, 210, 204, 209, 222, 228,
225, 0],
[98, 233, 198, 210, 222, 229, 229, 234, 249, 220, 194, 215, 217,
241, 65, 73, 106, 117, 168, 219, 221, 215, 217, 223, 223, 224,
229, 29],
[75, 204, 212, 204, 193, 205, 211, 225, 216, 185, 197, 206, 198,
213, 240, 195, 227, 245, 239, 223, 218, 212, 209, 222, 220, 221,
230, 67],
[48, 203, 183, 194, 213, 197, 185, 190, 194, 192, 202, 214, 219,
221, 220, 236, 225, 216, 199, 206, 186, 181, 177, 172, 181, 205,
206, 115],
[0, 122, 219, 193, 179, 171, 183, 196, 204, 210, 213, 207, 211,
210, 200, 196, 194, 191, 195, 191, 198, 192, 176, 156, 167, 177,
210, 92],
[0, 0, 74, 189, 212, 191, 175, 172, 175, 181, 185, 188, 189,
Executing (9m 36s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()

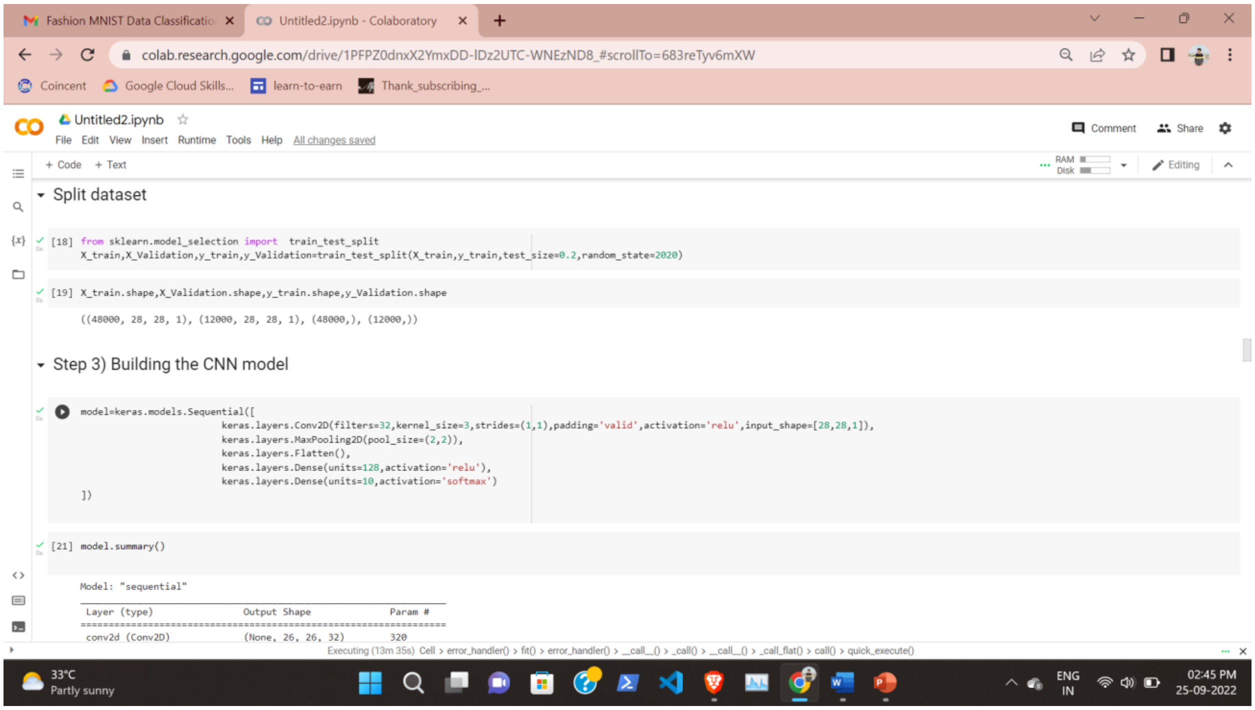

Fashion MNIST Data Classificatio
Untitled2.ipynb - Colaboratory
Untitled2.ipynb
File Edit View Insert Runtime Tools Help Saving...
+ Code + Text
206, 115],
[5]
[0, 122, 219, 193, 179, 171, 183, 196, 204, 210, 213, 207, 211,
210, 200, 196, 194, 191, 195, 191, 198, 192, 176, 156, 167, 177,
210, 92],
[0, 0, 74, 189, 212, 191, 175, 172, 175, 181, 185, 188, 189,
188, 193, 198, 204, 209, 210, 210, 211, 188, 188, 194, 192, 216,
170, 0],
[2, 0, 0, 0, 66, 200, 222, 237, 239, 242, 246, 243, 244,
221, 220, 193, 191, 179, 182, 182, 181, 176, 166, 168, 99, 58,
0, 0],
[0, 0, 0, 0, 0, 0, 0, 40, 61, 44, 72, 41, 35,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0]], dtype=uint8)
y_train[0]
9
[8] class_labels = ["T-shirt/top","Trouser","Pullover","Dress","Coat","Sandal","Shirt","Sneaker","Bag",'Ankel boot']
[9] class_labels
Executing (10m 0s) Cell > error_handler() > fit() > error_handler() > evaluate() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()

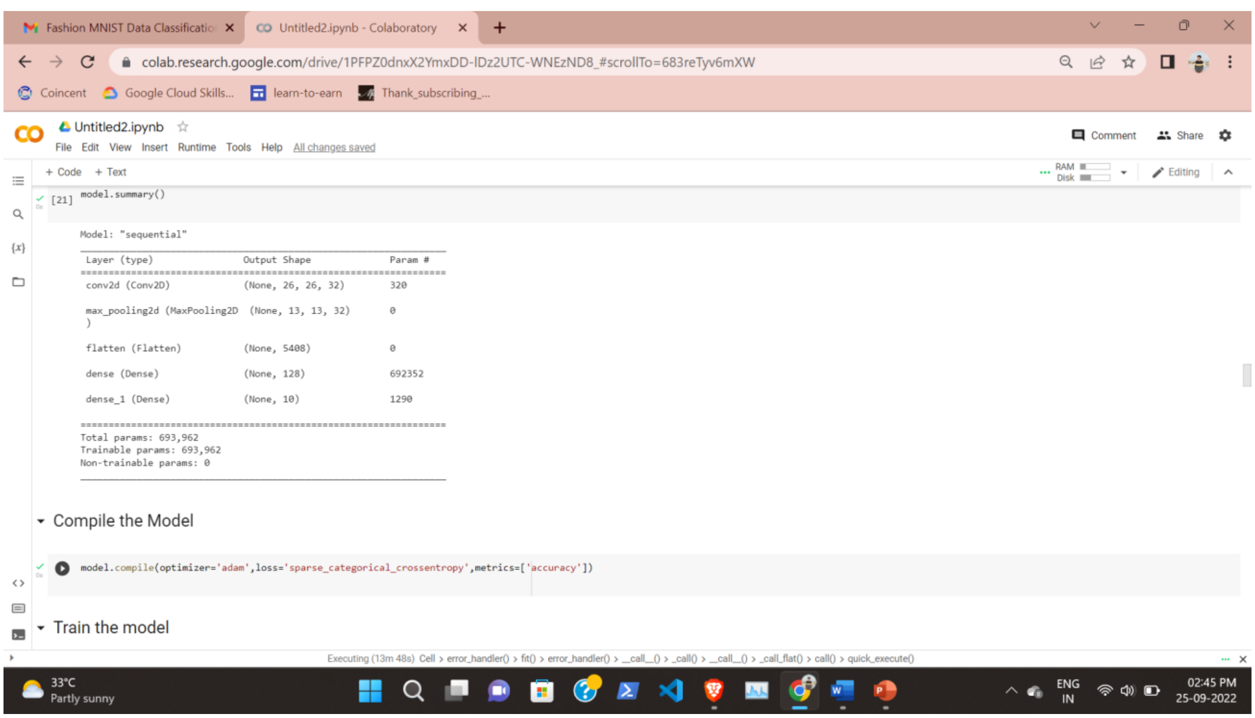

Fashion MNIST Data Classificatio
Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1PFPZ0dnxX2YmxDD-lDz2UTC-WNEzND8_#scrollTo=683reTyv6mXW
Coincent
Google Cloud Skills...
learn-to-earn
Thank_subscribing_...
Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[8] class_labels = ["T-shirt/top","Trouser","Pullover","Dress","Coat","Sandal","Shirt","Sneaker","Bag",'Ankel boot']
[9] class_labels
['T-shirt/top',
'Trouser',
'Pullover',
'Dress',
'Coat',
'Sandal',
'Shirt',
'Sneaker',
'Bag',
'Ankel boot']
[10] # show image
plt.imshow(X_train[0],cmap='Greys')
<matplotlib.image.AxesImage at 0x7f95823d1410>
Executing (11m 22s) Cell › error_handler() › fit() › error_handler() › __call__() › _call() › __call__() › _call_flat() › call() › quick_execute()
33°C
Partly sunny
ENG
IN
02:43 PM
25-09-2022

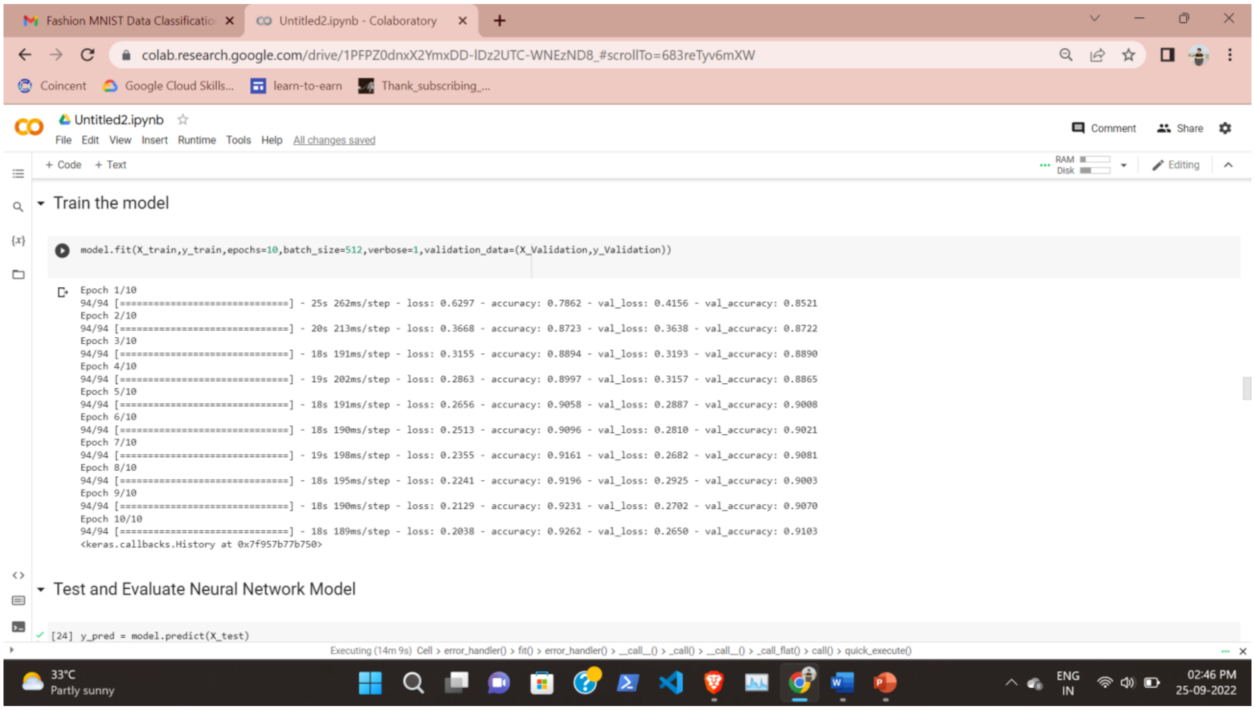

Fashion MNIST Data Classificatio
Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1PFPZ0dnxX2YmxDD-lDz2UTC-WNEzND8_#scrollTo=683reTyv6mXW
Coincent
Google Cloud Skills...
learn-to-earn
Thank_subscribing_...
Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[11]
plt.figure(figsize=(16,16))
j=1
for i in np.random.randint(0,1000,25):
plt.subplot(5,5,j);j+=1
plt.imshow(X_train[i],cmap='Greys')
plt.axis('off')
plt.title('{} / {}'.format(class_labels[y_train[i]],y_train[i]))
Bag / 8
Trouser / 1
T-shirt/top / 0
Bag / 8
Dress / 3
Shirt / 6
Bag / 8
Shirt / 6
Dress / 3
Shirt / 6
Executing (12m 8s) Cell › error_handler() › fit() › error_handler() › __call__() › _call() › __call__() › _call_flat() › call() › quick_execute()
33°C
Partly sunny
ENG
IN
02:44 PM
25-09-2022

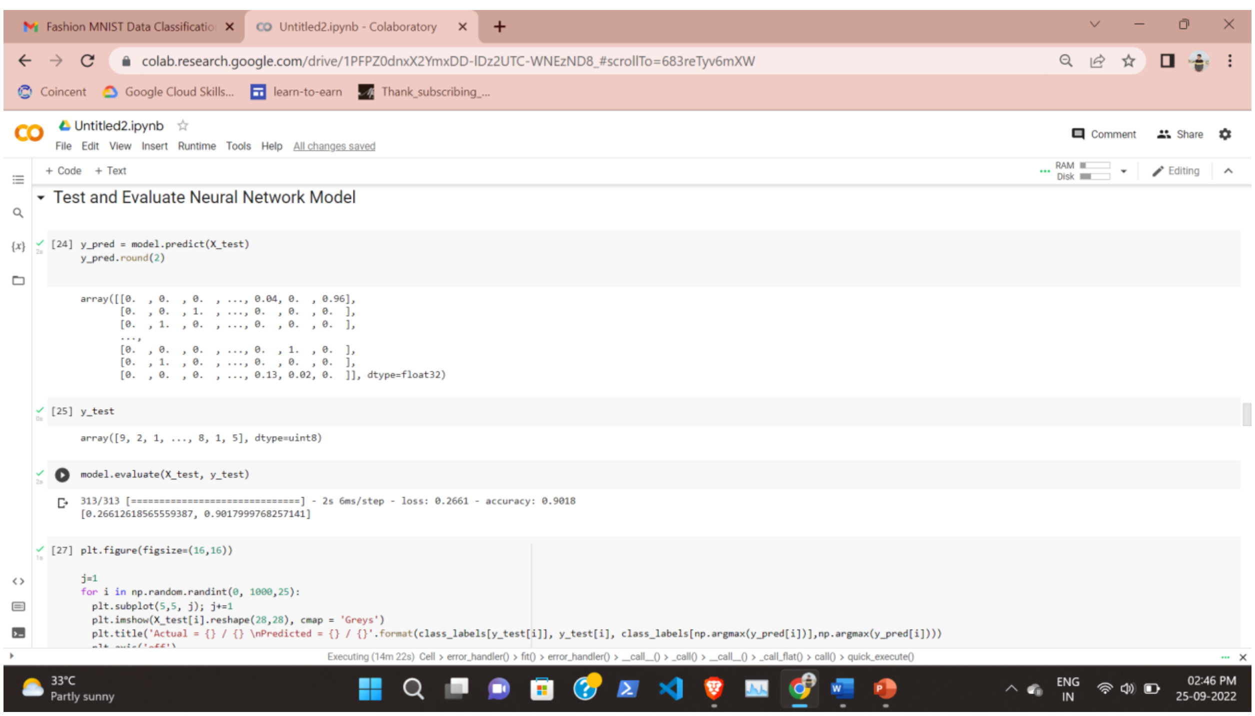

Fashion MNIST Data Classificatio
Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1PFPZ0dnxX2YmxDD-lDz2UTC-WNEzND8_#scrollTo=683reTyv6mXW
Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
[12]
Sandal / 5
Shirt / 6
T-shirt/top / 0
T-shirt/top / 0
Ankel boot / 9
Coat / 4
Dress / 3
Ankel boot / 9
Bag / 8
Sneaker / 7
Pullover / 2
Bag / 8
Sneaker / 7
Sandal / 5
T-shirt/top / 0
X_train.ndim
33°C
Partly sunny
02:44 PM
25-09-2022

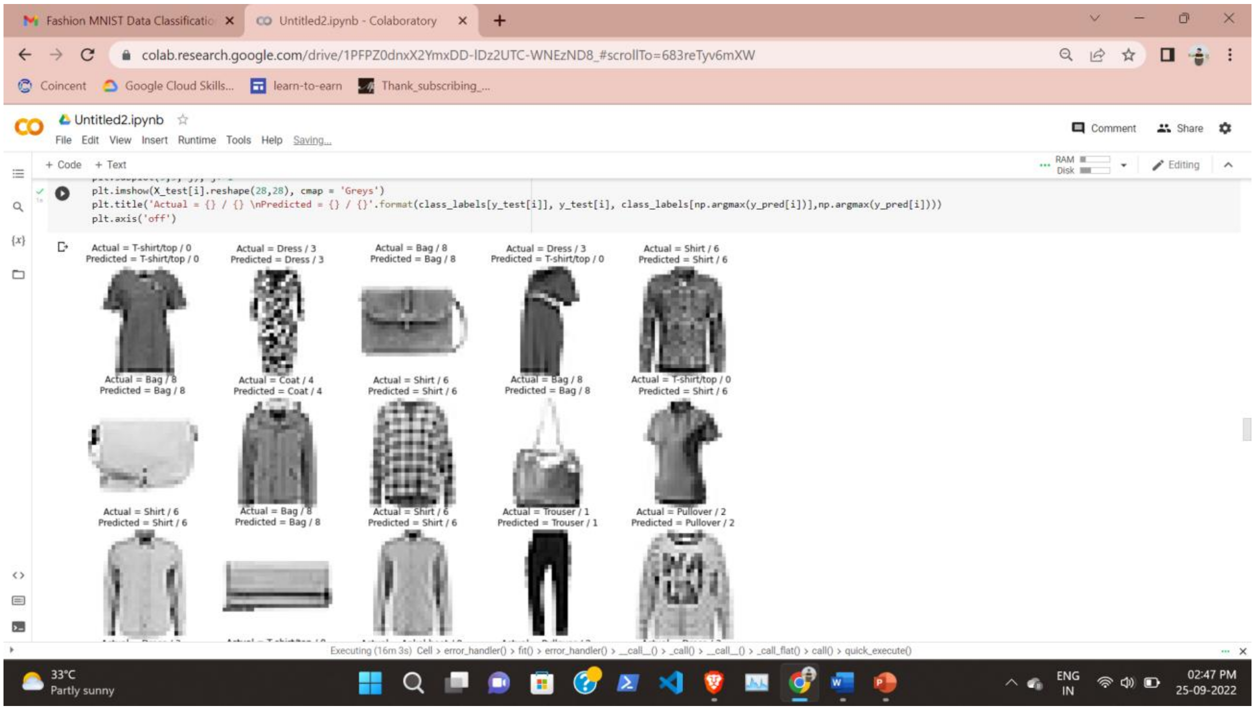

Fashion MNIST Data Classificatio
Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1PFPZ0dnxX2YmxDD-lDz2UTC-WNEzND8_#scrollTo=683reTyv6mXW
Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
[13] X_train.ndim
3
[14] X_train = np.expand_dims(X_train,-1)
[15] X_train.ndim
4
[16] X_test=np.expand_dims(X_test,-1)
Feature Scalling
[17] X_train = X_train/255
X_test = X_test/255
Split dataset
[18] from sklearn.model_selection import train_test_split
X_train,X_Validation,y_train,y_Validation=train_test_split(X_train,y_train,test_size=0.2,random_state=2020)
Executing (13m 18s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()
33°C
Partly sunny
02:45 PM
25-09-2022

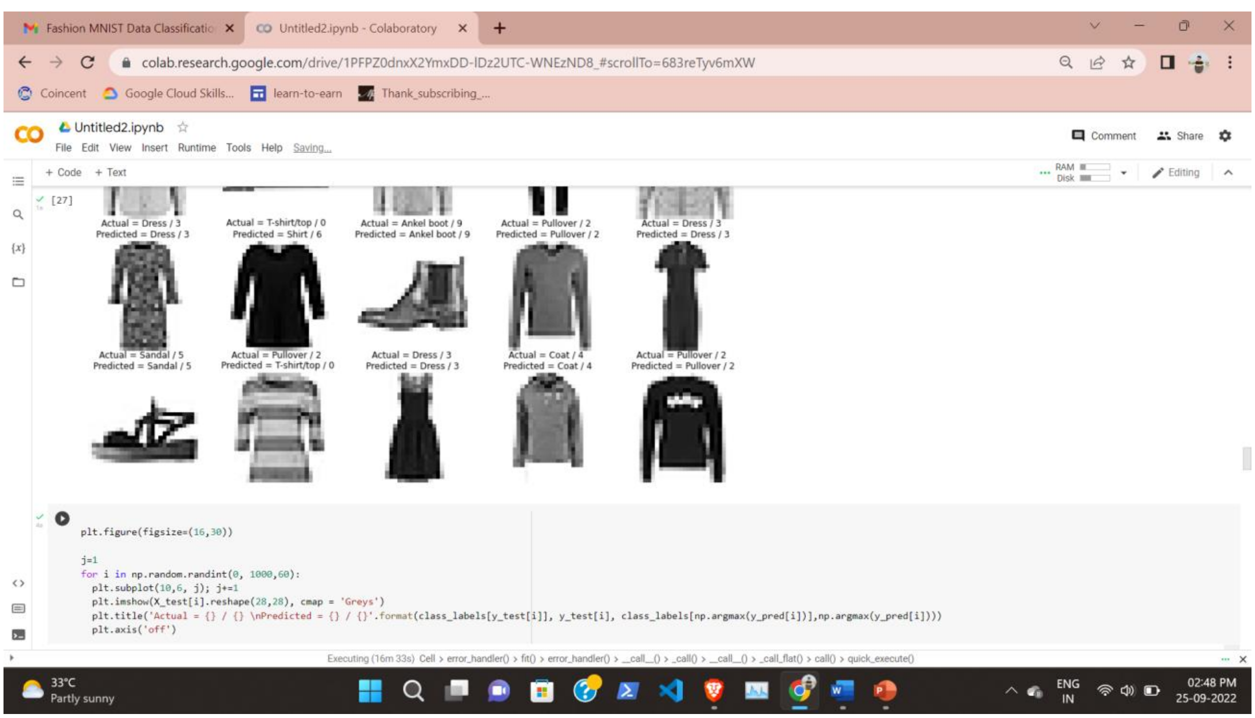

Fashion MNIST Data Classificatio
Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1PFPZ0dnxX2YmxDD-lDz2UTC-WNEzND8_#scrollTo=683reTyv6mXW
Coincent
Google Cloud Skills...
learn-to-earn
Thank_subscribing_...
Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Editing
Split dataset
[18] from sklearn.model_selection import train_test_split
X_train,X_Validation,y_train,y_Validation=train_test_split(X_train,y_train,test_size=0.2,random_state=2020)
[19] X_train.shape,X_Validation.shape,y_train.shape,y_Validation.shape
((48000, 28, 28, 1), (12000, 28, 28, 1), (48000,), (12000,))
Step 3) Building the CNN model
model=keras.models.Sequential([
keras.layers.Conv2D(filters=32,kernel_size=3,strides=(1,1),padding='valid',activation='relu',input_shape=[28,28,1]),
keras.layers.MaxPooling2D(pool_size=(2,2)),
keras.layers.Flatten(),
keras.layers.Dense(units=128,activation='relu'),
keras.layers.Dense(units=10,activation='softmax')
])
[21] model.summary()
Model: "sequential"
Layer (type) Output Shape Param #
conv2d (Conv2D) (None, 26, 26, 32) 320
Executing (13m 35s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()
33°C
Partly sunny
ENG
IN
02:45 PM
25-09-2022

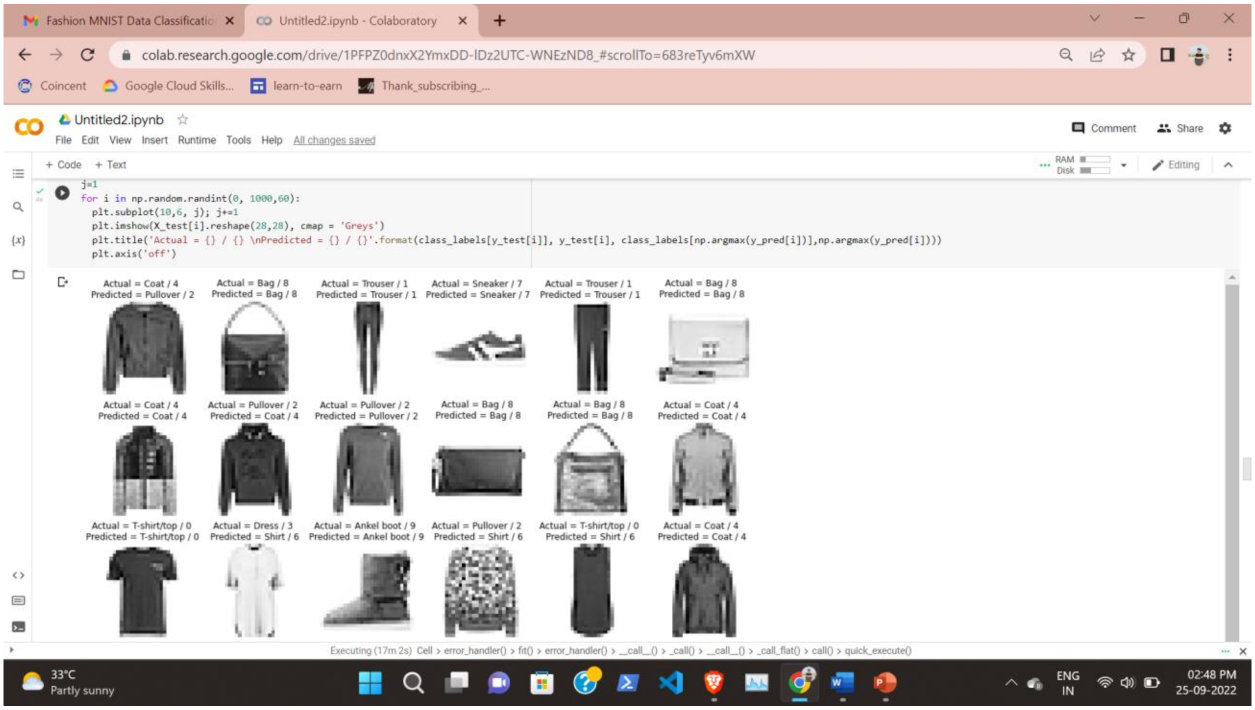

Fashion MNIST Data Classificatio
Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1PFPZ0dnxX2YmxDD-lDz2UTC-WNEzND8_#scrollTo=683reTyv6mXW
Coincent
Google Cloud Skills...
learn-to-earn
Thank_subscribing_...
Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Editing
[21] model.summary()
Model: "sequential"
Layer (type) Output Shape Param #
conv2d (Conv2D) (None, 26, 26, 32) 320
max_pooling2d (MaxPooling2D) (None, 13, 13, 32) 0
flatten (Flatten) (None, 5408) 0
dense (Dense) (None, 128) 692352
dense_1 (Dense) (None, 10) 1290
Total params: 693,962
Trainable params: 693,962
Non-trainable params: 0
Compile the Model
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
Train the model
Executing (13m 48s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()
33°C
Partly sunny
ENG
IN
02:45 PM
25-09-2022

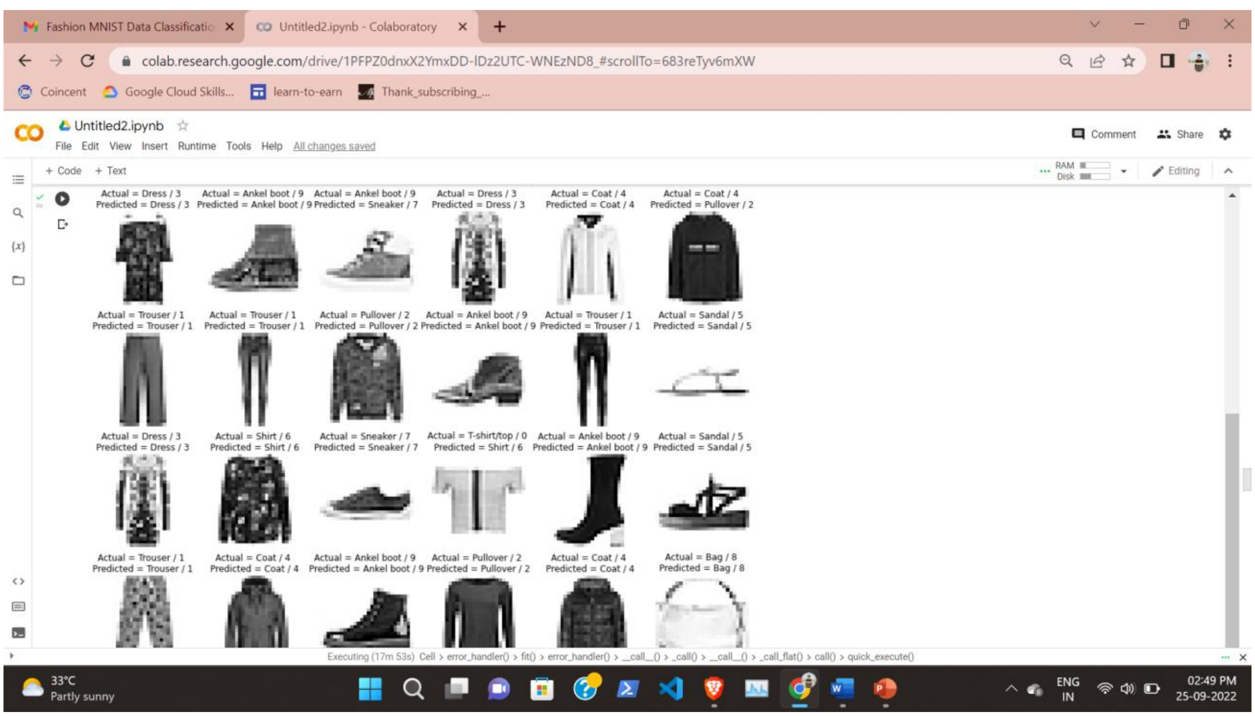

Untitled2.ipynb
Train the model
model.fit(X_train,y_train,epochs=10,batch_size=512,verbose=1,validation_data=(X_Validation,y_Validation))
Epoch 1/10
94/94 [==============================] - 25s 262ms/step - loss: 0.6297 - accuracy: 0.7862 - val_loss: 0.4156 - val_accuracy: 0.8521
Epoch 2/10
94/94 [==============================] - 20s 213ms/step - loss: 0.3668 - accuracy: 0.8723 - val_loss: 0.3638 - val_accuracy: 0.8722
Epoch 3/10
94/94 [==============================] - 18s 191ms/step - loss: 0.3155 - accuracy: 0.8894 - val_loss: 0.3193 - val_accuracy: 0.8890
Epoch 4/10
94/94 [==============================] - 19s 202ms/step - loss: 0.2863 - accuracy: 0.8997 - val_loss: 0.3157 - val_accuracy: 0.8865
Epoch 5/10
94/94 [==============================] - 18s 191ms/step - loss: 0.2656 - accuracy: 0.9058 - val_loss: 0.2887 - val_accuracy: 0.9008
Epoch 6/10
94/94 [==============================] - 18s 190ms/step - loss: 0.2513 - accuracy: 0.9096 - val_loss: 0.2810 - val_accuracy: 0.9021
Epoch 7/10
94/94 [==============================] - 19s 198ms/step - loss: 0.2355 - accuracy: 0.9161 - val_loss: 0.2682 - val_accuracy: 0.9081
Epoch 8/10
94/94 [==============================] - 18s 195ms/step - loss: 0.2241 - accuracy: 0.9196 - val_loss: 0.2925 - val_accuracy: 0.9003
Epoch 9/10
94/94 [==============================] - 18s 190ms/step - loss: 0.2129 - accuracy: 0.9231 - val_loss: 0.2702 - val_accuracy: 0.9070
Epoch 10/10
94/94 [==============================] - 18s 189ms/step - loss: 0.2038 - accuracy: 0.9262 - val_loss: 0.2650 - val_accuracy: 0.9103
<keras.callbacks.History at 0x7f957b77b750>
Test and Evaluate Neural Network Model
[24] y_pred = model.predict(X_test)

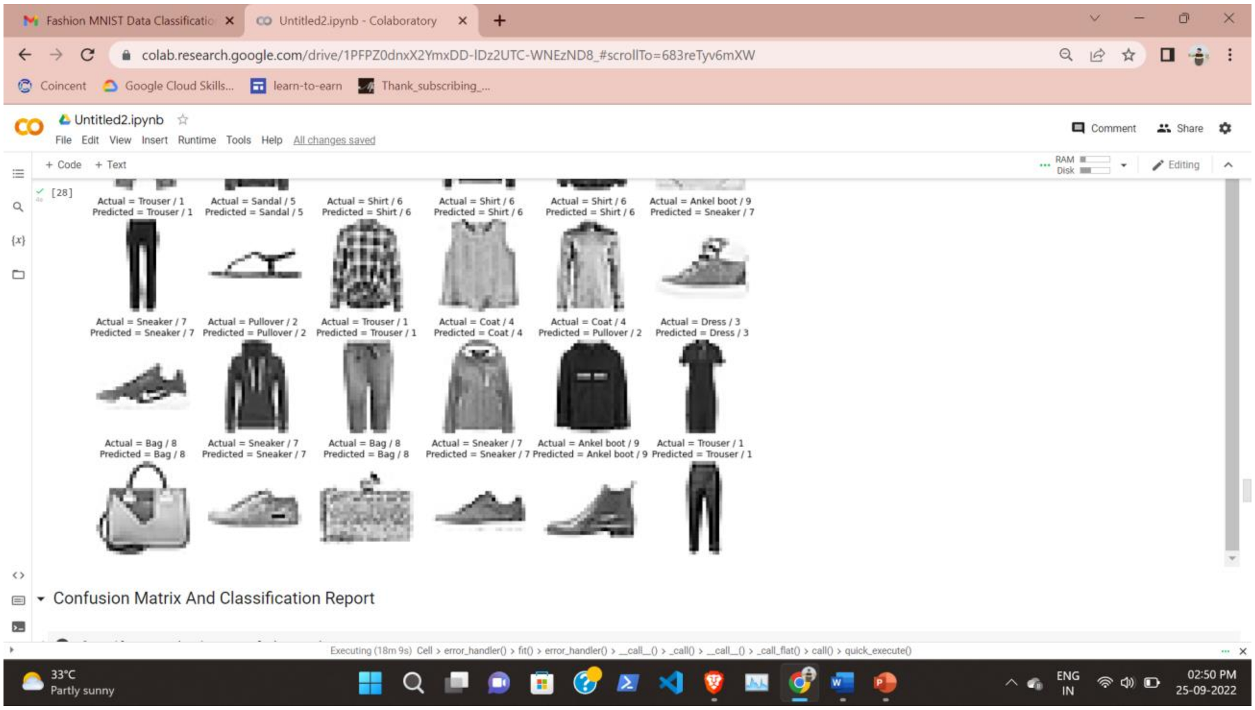

Untitled2.ipynb
Test and Evaluate Neural Network Model
[24] y_pred = model.predict(X_test)
y_pred.round(2)
array([[0. , 0. , 0. , ..., 0.04, 0. , 0.96],
[0. , 0. , 1. , ..., 0. , 0. , 0.],
[0. , 1. , 0. , ..., 0. , 0. , 0.],
...,
[0. , 0. , 0. , ..., 0. , 1. , 0.],
[0. , 1. , 0. , ..., 0. , 0. , 0.],
[0. , 0. , 0. , ..., 0.13, 0.02, 0.]], dtype=float32)
[25] y_test
array([9, 2, 1, ..., 8, 1, 5], dtype=uint8)
model.evaluate(X_test, y_test)
313/313 [==============================] - 2s 6ms/step - loss: 0.2661 - accuracy: 0.9018
[0.26612618565559387, 0.9017999768257141]
[27] plt.figure(figsize=(16,16))
j=1
for i in np.random.randint(0, 1000,25):
plt.subplot(5,5, j); j+=1
plt.imshow(X_test[i].reshape(28,28), cmap = 'Greys')
plt.title('Actual = {} / {} \nPredicted = {} / {}'.format(class_labels[y_test[i]], y_test[i], class_labels[np.argmax(y_pred[i])],np.argmax(y_pred[i])))

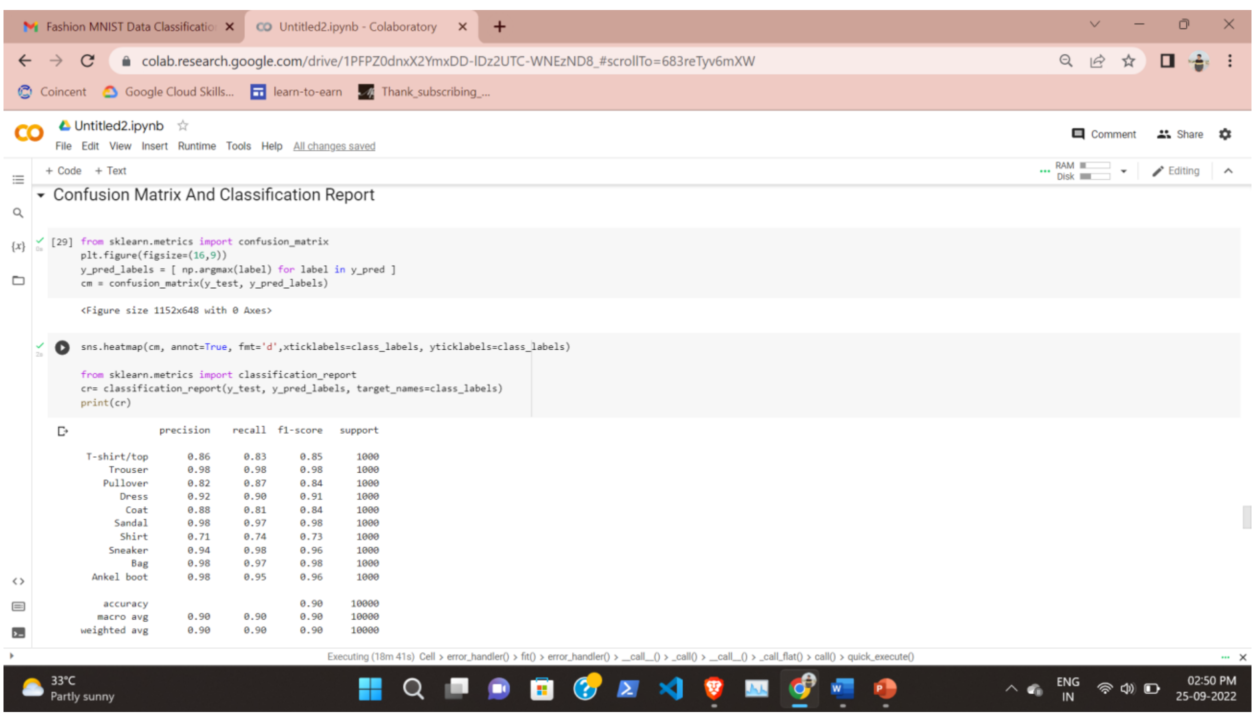

Fashion MNIST Data Classificatio
Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1PFPZ0dnxX2YmxDD-lDz2UTC-WNEzND8_#scrollTo=683reTyv6mXW
Untitled2.ipynb
File Edit View Insert Runtime Tools Help Saving...
plt.imshow(X_test[i].reshape(28,28), cmap = 'Greys')
plt.title('Actual = {} / {} \nPredicted = {} / {}'.format(class_labels[y_test[i]], y_test[i], class_labels[np.argmax(y_pred[i])],np.argmax(y_pred[i])))
plt.axis('off')
Actual = T-shirt/top / 0
Predicted = T-shirt/top / 0
Actual = Dress / 3
Predicted = Dress / 3
Actual = Bag / 8
Predicted = Bag / 8
Actual = Dress / 3
Predicted = T-shirt/top / 0
Actual = Shirt / 6
Predicted = Shirt / 6
Actual = Bag / 8
Predicted = Bag / 8
Actual = Coat / 4
Predicted = Coat / 4
Actual = Shirt / 6
Predicted = Shirt / 6
Actual = Bag / 8
Predicted = Bag / 8
Actual = T-shirt/top / 0
Predicted = Shirt / 6
Actual = Shirt / 6
Predicted = Shirt / 6
Actual = Bag / 8
Predicted = Bag / 8
Actual = Shirt / 6
Predicted = Shirt / 6
Actual = Trouser / 1
Predicted = Trouser / 1
Actual = Pullover / 2
Predicted = Pullover / 2
Executing (16m 3s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()
33°C
Partly sunny
02:47 PM
25-09-2022


Fashion MNIST Data Classificatio
Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1PFPZ0dnxX2YmxDD-lDz2UTC-WNEzND8_#scrollTo=683reTyv6mXW
Untitled2.ipynb
File Edit View Insert Runtime Tools Help Saving...
[27]
Actual = Dress / 3
Predicted = Dress / 3
Actual = T-shirt/top / 0
Predicted = Shirt / 6
Actual = Ankel boot / 9
Predicted = Ankel boot / 9
Actual = Pullover / 2
Predicted = Pullover / 2
Actual = Dress / 3
Predicted = Dress / 3
Actual = Sandal / 5
Predicted = Sandal / 5
Actual = Pullover / 2
Predicted = T-shirt/top / 0
Actual = Dress / 3
Predicted = Dress / 3
Actual = Coat / 4
Predicted = Coat / 4
Actual = Pullover / 2
Predicted = Pullover / 2
plt.figure(figsize=(16,30))
j=1
for i in np.random.randint(0, 1000,60):
plt.subplot(10,6, j); j+=1
plt.imshow(X_test[i].reshape(28,28), cmap = 'Greys')
plt.title('Actual = {} / {} \nPredicted = {} / {}'.format(class_labels[y_test[i]], y_test[i], class_labels[np.argmax(y_pred[i])],np.argmax(y_pred[i])))
plt.axis('off')
Executing (16m 33s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()
33°C
Partly sunny
02:48 PM
25-09-2022


Fashion MNIST Data Classificatio
Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1PFPZ0dnxX2YmxDD-lDz2UTC-WNEzND8_#scrollTo=683reTyv6mXW
Coincent
Google Cloud Skills...
learn-to-earn
Thank_subscribing_...
Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Editing
j=1
for i in np.random.randint(0, 1000,60):
plt.subplot(10,6, j); j+=1
plt.imshow(X_test[i].reshape(28,28), cmap = 'Greys')
plt.title('Actual = {} / {} \nPredicted = {} / {}'.format(class_labels[y_test[i]], y_test[i], class_labels[np.argmax(y_pred[i])],np.argmax(y_pred[i])))
plt.axis('off')
Actual = Coat / 4 Predicted = Pullover / 2
Actual = Bag / 8 Predicted = Bag / 8
Actual = Trouser / 1 Predicted = Trouser / 1
Actual = Sneaker / 7 Predicted = Sneaker / 7
Actual = Trouser / 1 Predicted = Trouser / 1
Actual = Bag / 8 Predicted = Bag / 8
Actual = Coat / 4 Predicted = Coat / 4
Actual = Pullover / 2 Predicted = Coat / 4
Actual = Pullover / 2 Predicted = Pullover / 2
Actual = Bag / 8 Predicted = Bag / 8
Actual = Bag / 8 Predicted = Bag / 8
Actual = Coat / 4 Predicted = Coat / 4
Actual = T-shirt/top / 0 Predicted = T-shirt/top / 0
Actual = Dress / 3 Predicted = Shirt / 6
Actual = Ankel boot / 9 Predicted = Ankel boot / 9
Actual = Pullover / 2 Predicted = Shirt / 6
Actual = T-shirt/top / 0 Predicted = Shirt / 6
Actual = Coat / 4 Predicted = Coat / 4
Executing (17m 2s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()
33°C Partly sunny
ENG IN
02:48 PM 25-09-2022


Fashion MNIST Data Classificatio
Untitled2.ipynb - Colaboratory
colab.research.google.com/drive/1PFPZ0dnxX2YmxDD-lDz2UTC-WNEzND8_#scrollTo=683reTyv6mXW
Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Actual = Dress / 3 Predicted = Dress / 3
Actual = Ankel boot / 9 Predicted = Ankel boot / 9
Actual = Ankel boot / 9 Predicted = Sneaker / 7
Actual = Dress / 3 Predicted = Dress / 3
Actual = Coat / 4 Predicted = Coat / 4
Actual = Coat / 4 Predicted = Pullover / 2
Actual = Trouser / 1 Predicted = Trouser / 1
Actual = Trouser / 1 Predicted = Trouser / 1
Actual = Pullover / 2 Predicted = Pullover / 2
Actual = Ankel boot / 9 Predicted = Ankel boot / 9
Actual = Trouser / 1 Predicted = Trouser / 1
Actual = Sandal / 5 Predicted = Sandal / 5
Actual = Dress / 3 Predicted = Dress / 3
Actual = Shirt / 6 Predicted = Shirt / 6
Actual = Sneaker / 7 Predicted = Sneaker / 7
Actual = T-shirt/top / 0 Predicted = Shirt / 6
Actual = Ankel boot / 9 Predicted = Ankel boot / 9
Actual = Sandal / 5 Predicted = Sandal / 5
Actual = Trouser / 1 Predicted = Trouser / 1
Actual = Coat / 4 Predicted = Coat / 4
Actual = Ankel boot / 9 Predicted = Ankel boot / 9
Actual = Pullover / 2 Predicted = Pullover / 2
Actual = Coat / 4 Predicted = Coat / 4
Actual = Bag / 8 Predicted = Bag / 8
Executing (17m 53s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()
33°C Partly sunny
ENG IN
02:49 PM 25-09-2022



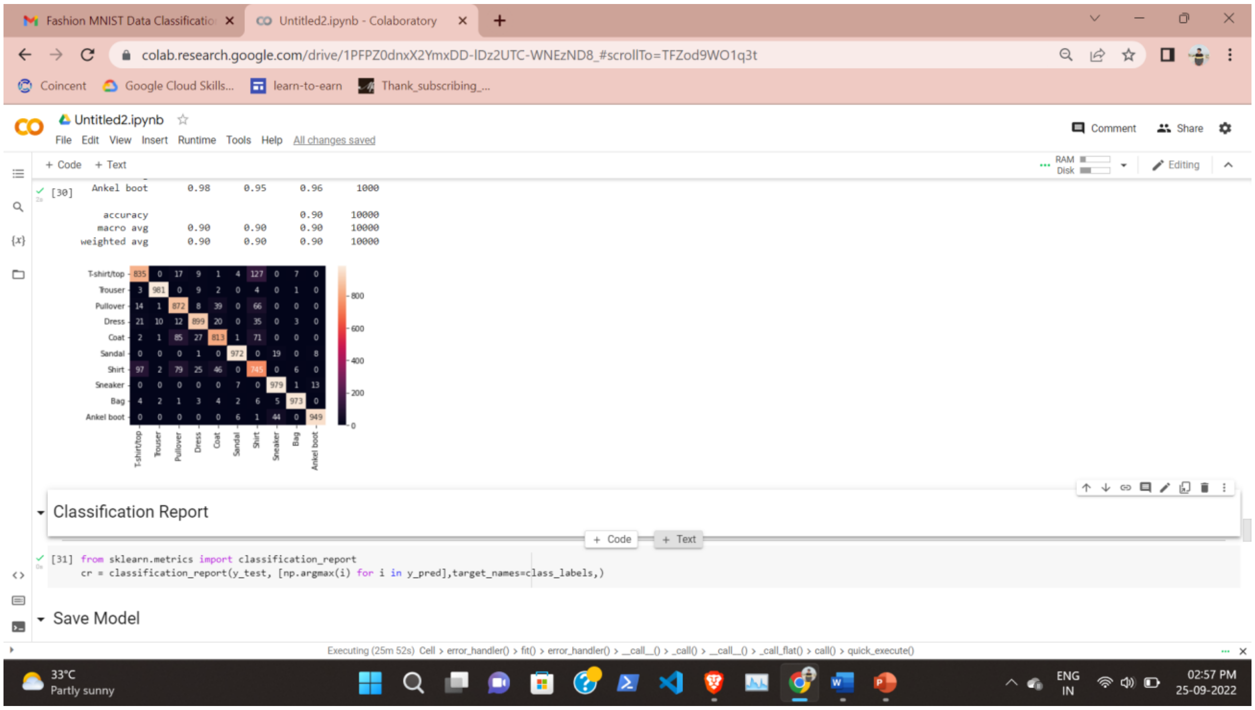


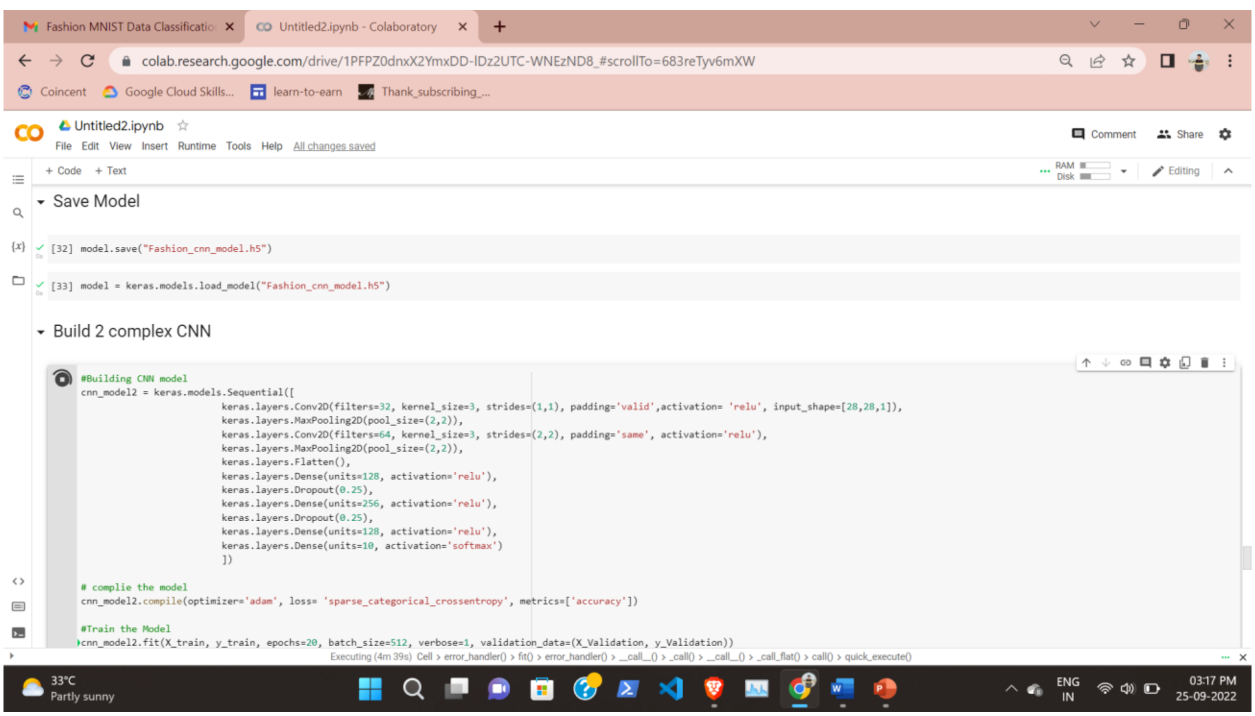

```
from sklearn.metrics import confusion_matrix
plt.figure(figsize=(16,9))
y_pred_labels = [ np.argmax(label) for label in y_pred ]
cm = confusion_matrix(y_test, y_pred_labels)
```

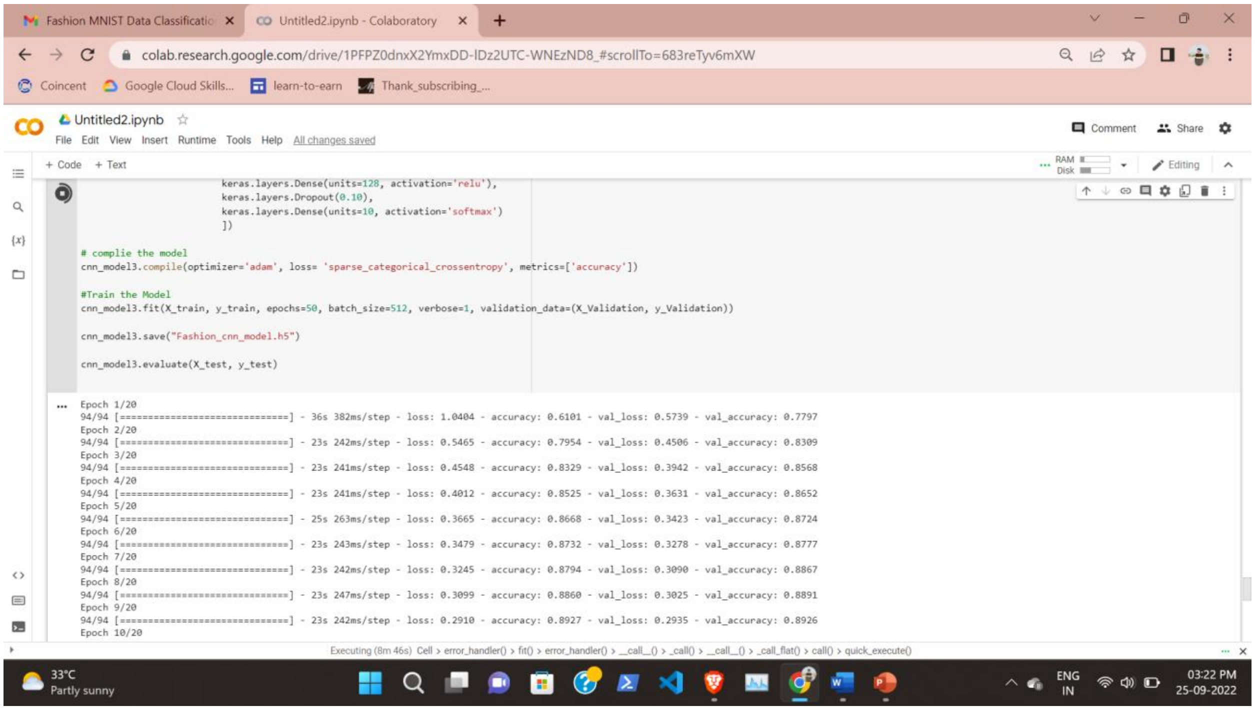
```
<Figure size 1152x648 with 0 Axes>
```

```
sns.heatmap(cm, annot=True, fmt='d',xticklabels=class_labels, yticklabels=class_labels)

from sklearn.metrics import classification_report
cr= classification_report(y_test, y_pred_labels, target_names=class_labels)
print(cr)
```

	precision	recall	f1-score	support
T-shirt/top	0.86	0.83	0.85	1000
Trouser	0.98	0.98	0.98	1000
Pullover	0.82	0.87	0.84	1000
Dress	0.92	0.90	0.91	1000
Coat	0.88	0.81	0.84	1000
Sandal	0.98	0.97	0.98	1000
Shirt	0.71	0.74	0.73	1000
Sneaker	0.94	0.98	0.96	1000
Bag	0.98	0.97	0.98	1000
Ankel boot	0.98	0.95	0.96	1000
accuracy			0.90	10000
macro avg	0.90	0.90	0.90	10000
weighted avg	0.90	0.90	0.90	10000


Untitled2.ipynb
[30]
Ankel boot 0.98 0.95 0.96 1000
accuracy 0.90 10000
macro avg 0.90 0.90 0.90 10000
weighted avg 0.90 0.90 0.90 10000
T-shirt/top 835 0 17 9 1 4 127 0 7 0
Trouser 3 981 0 9 2 0 4 0 1 0
Pullover 14 1 872 8 39 0 66 0 0 0
Dress 21 10 12 899 20 0 35 0 3 0
Coat 2 1 85 27 813 1 71 0 0 0
Sandal 0 0 0 1 0 972 0 19 0 8
Shirt 97 2 79 25 46 0 745 0 6 0
Sneaker 0 0 0 0 0 7 0 979 1 13
Bag 4 2 1 3 4 2 6 5 973 0
Ankel boot 0 0 0 0 0 6 1 44 0 949
T-shirt/top Trouser Pullover Dress Coat Sandal Shirt Sneaker Bag Ankel boot
800 600 400 200 0
Classification Report
[31] from sklearn.metrics import classification_report
cr = classification_report(y_test, [np.argmax(i) for i in y_pred],target_names=class_labels,)
Save Model
Executing (25m 52s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()
33°C Partly sunny
02:57 PM 25-09-2022


Untitled2.ipynb
Save Model
[32] model.save("Fashion_cnn_model.h5")
[33] model = keras.models.load_model("Fashion_cnn_model.h5")
Build 2 complex CNN
#Building CNN model
cnn_model2 = keras.models.Sequential([
keras.layers.Conv2D(filters=32, kernel_size=3, strides=(1,1), padding='valid',activation= 'relu', input_shape=[28,28,1]),
keras.layers.MaxPooling2D(pool_size=(2,2)),
keras.layers.Conv2D(filters=64, kernel_size=3, strides=(2,2), padding='same', activation='relu'),
keras.layers.MaxPooling2D(pool_size=(2,2)),
keras.layers.Flatten(),
keras.layers.Dense(units=128, activation='relu'),
keras.layers.Dropout(0.25),
keras.layers.Dense(units=256, activation='relu'),
keras.layers.Dropout(0.25),
keras.layers.Dense(units=128, activation='relu'),
keras.layers.Dense(units=10, activation='softmax')
])
# complie the model
cnn_model2.compile(optimizer='adam', loss= 'sparse_categorical_crossentropy', metrics=['accuracy'])
#Train the Model
cnn_model2.fit(X_train, y_train, epochs=20, batch_size=512, verbose=1, validation_data=(X_Validation, y_Validation))
Executing (4m 39s) Cell > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()
33°C Partly sunny
03:17 PM 25-09-2022



CONCLUSION:-

Conclusions In this article, we applied various classification methods on an image classification problem. We have explained why the CNNs are the best method we can employ out of considered ones, and why do the other methods fail. Some of the reasons why CNNs are the most practical and usually the most accurate method are:

They can transfer learning through layers, saving inferences, and making new ones on subsequent layers.

No need for feature extraction before using the algorithm, it is done during training.

It recognizes important features.

However, they also have their caveats. They are known to fail on images that are rotated and scaled differently, which is not the case here, as the data was pre-processed. And, although the other methods fail to give that good results on this dataset, they are still used for other tasks related to image processing (sharpening, smoothing etc.).