```
# Image classification using Sequential NN
#MNIST character dataset classification
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
```

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

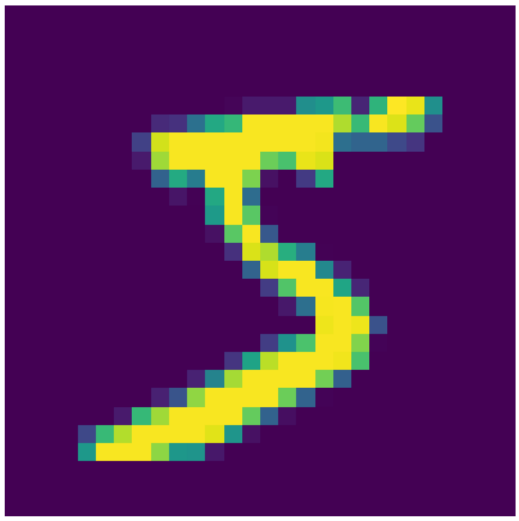
```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

```
print(x_train[0])
```

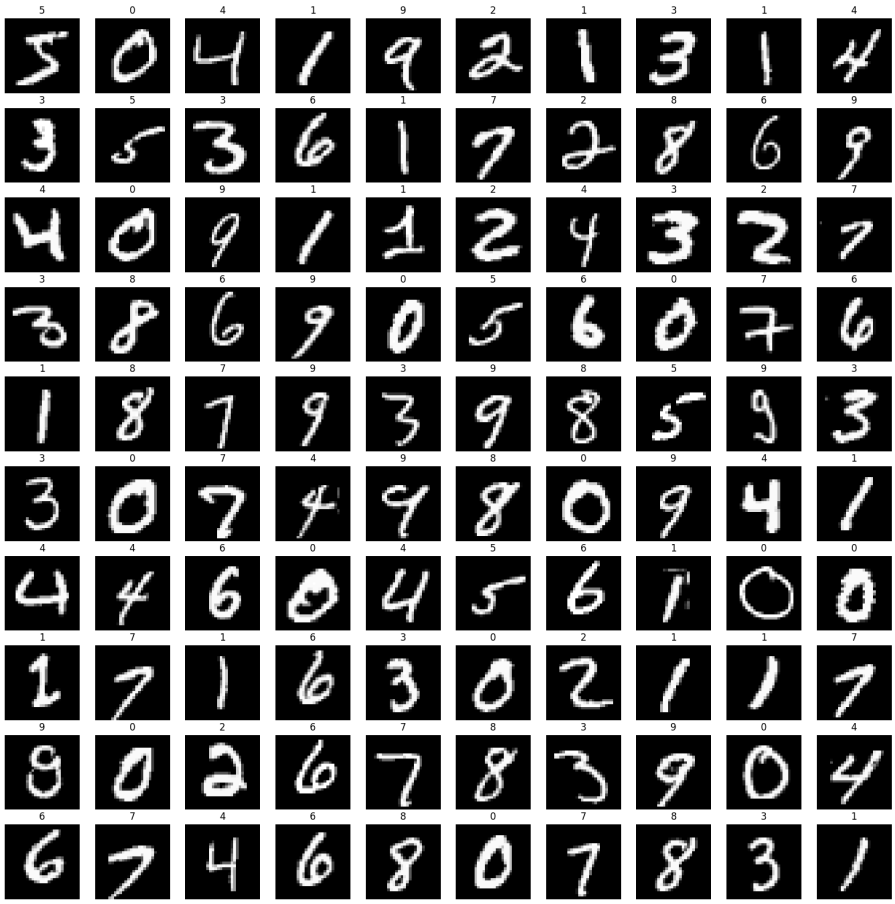
```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   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   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   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  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
   25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   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   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]]
```

```
plt.imshow(x_train[0])
plt.axis('off')
```

```
(-0.5, 27.5, 27.5, -0.5)
```



```
plt.figure(figsize=(20,20))
for i in range(100):
  plt.subplot(10,10,i+1)
  plt.imshow(x_train[i],cmap='gray')
  plt.axis('off')
  plt.title(''+str(y_train[i]))
```



```
############# MNIST Fashion dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

```
class_label=['T-shirt/top','Trouser','Pullover','Dress','Coat','Sandal','Shirt',
             'Sneaker','Bag','Ankle boot']
plt.figure(figsize=(20,20))
for i in range(100):
  plt.subplot(10,10,i+1)
  plt.imshow(x_train[i],cmap='gray')
  plt.axis('off')
  plt.title(class_label[y_train[i]])
```


```
#preprocessing- scaling
x_train=x_train/255.0
x_test=x_test/255.0


# Instantiate the model.
model = tf.keras.models.Sequential()

# Build the model.
model.add(tf.keras.layers.Flatten(input_shape=(28,28)))
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(10, activation="softmax"))

# Display the model summary.
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 128)               100480

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model.compile(
    optimizer="rmsprop", # 'sgd','adam'
    loss="sparse_categorical_crossentropy", # binary_crossentropy,categorical_crossentropy
    metrics=["accuracy"],
)
```

```
Start coding or generate with AI.
```

```
hist = model.fit(x_train,y_train,epochs=21,batch_size=64,validation_split=0.2)# validation_data=(X_valid, y_valid));
```

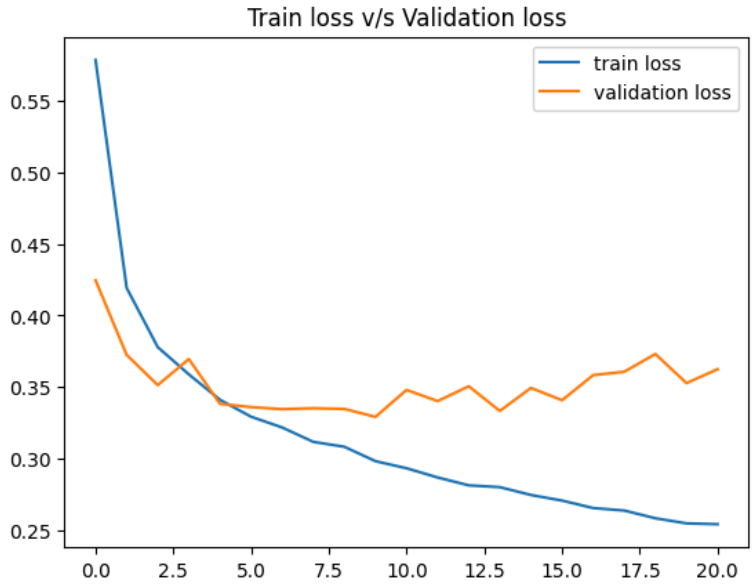
```
Epoch 1/21
750/750 [==============================] - 9s 11ms/step - loss: 0.5784 - accuracy: 0.7967 - val_loss: 0.4245 - val_accuracy: 0.
Epoch 2/21
750/750 [==============================] - 6s 8ms/step - loss: 0.4192 - accuracy: 0.8493 - val_loss: 0.3725 - val_accuracy: 0.8
Epoch 3/21
750/750 [==============================] - 6s 8ms/step - loss: 0.3778 - accuracy: 0.8640 - val_loss: 0.3513 - val_accuracy: 0.8
Epoch 4/21
750/750 [==============================] - 3s 4ms/step - loss: 0.3588 - accuracy: 0.8701 - val_loss: 0.3695 - val_accuracy: 0.8
Epoch 5/21
750/750 [==============================] - 3s 4ms/step - loss: 0.3411 - accuracy: 0.8778 - val_loss: 0.3383 - val_accuracy: 0.8
Epoch 6/21
750/750 [==============================] - 3s 4ms/step - loss: 0.3293 - accuracy: 0.8810 - val_loss: 0.3359 - val_accuracy: 0.8
Epoch 7/21
750/750 [==============================] - 4s 5ms/step - loss: 0.3217 - accuracy: 0.8837 - val_loss: 0.3346 - val_accuracy: 0.8
Epoch 8/21
750/750 [==============================] - 3s 4ms/step - loss: 0.3117 - accuracy: 0.8874 - val_loss: 0.3352 - val_accuracy: 0.8
Epoch 9/21
750/750 [==============================] - 3s 4ms/step - loss: 0.3083 - accuracy: 0.8897 - val_loss: 0.3347 - val_accuracy: 0.8
Epoch 10/21
750/750 [==============================] - 3s 4ms/step - loss: 0.2982 - accuracy: 0.8906 - val_loss: 0.3291 - val_accuracy: 0.8
Epoch 11/21
750/750 [==============================] - 4s 5ms/step - loss: 0.2932 - accuracy: 0.8938 - val_loss: 0.3479 - val_accuracy: 0.8
Epoch 12/21
750/750 [==============================] - 3s 4ms/step - loss: 0.2868 - accuracy: 0.8966 - val_loss: 0.3402 - val_accuracy: 0.8
Epoch 13/21
750/750 [==============================] - 3s 4ms/step - loss: 0.2813 - accuracy: 0.8979 - val_loss: 0.3505 - val_accuracy: 0.8
Epoch 14/21
750/750 [==============================] - 3s 4ms/step - loss: 0.2800 - accuracy: 0.8992 - val_loss: 0.3333 - val_accuracy: 0.8
Epoch 15/21
750/750 [==============================] - 4s 5ms/step - loss: 0.2746 - accuracy: 0.9026 - val_loss: 0.3494 - val_accuracy: 0.8
Epoch 16/21
750/750 [==============================] - 3s 4ms/step - loss: 0.2707 - accuracy: 0.9034 - val_loss: 0.3407 - val_accuracy: 0.8
Epoch 17/21
750/750 [==============================] - 3s 4ms/step - loss: 0.2654 - accuracy: 0.9045 - val_loss: 0.3583 - val_accuracy: 0.8
Epoch 18/21
750/750 [==============================] - 3s 4ms/step - loss: 0.2637 - accuracy: 0.9049 - val_loss: 0.3607 - val_accuracy: 0.8
Epoch 19/21
750/750 [==============================] - 4s 5ms/step - loss: 0.2583 - accuracy: 0.9057 - val_loss: 0.3731 - val_accuracy: 0.8
Epoch 20/21
750/750 [==============================] - 3s 4ms/step - loss: 0.2548 - accuracy: 0.9087 - val_loss: 0.3527 - val_accuracy: 0.8
Epoch 21/21
750/750 [==============================] - 3s 4ms/step - loss: 0.2542 - accuracy: 0.9083 - val_loss: 0.3624 - val_accuracy: 0.8
```

```
dict=hist.history
print(dict.keys())
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

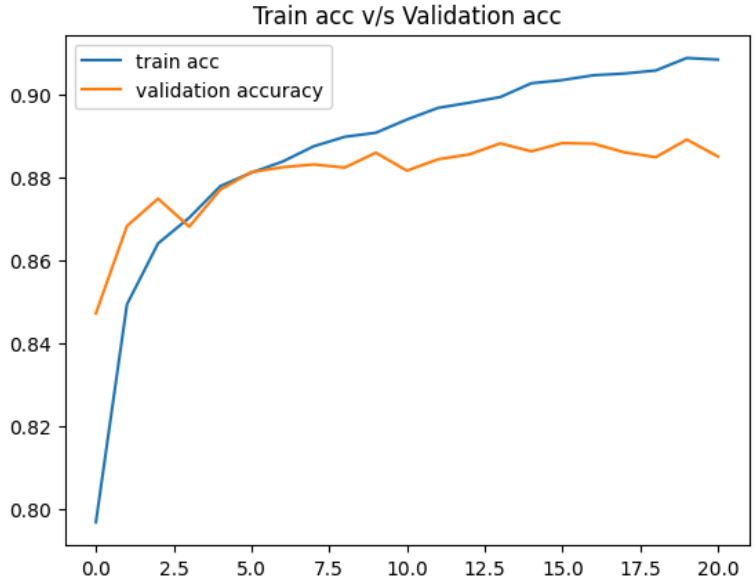
```
plt.plot(dict['loss'],label='train loss')
plt.plot(dict['val_loss'],label='validation loss')
plt.legend()
plt.title('Train loss v/s Validation loss')
```

```
Text(0.5, 1.0, 'Train loss v/s Validation loss')
```




```
plt.plot(dict['accuracy'],label='train acc')
plt.plot(dict['val_accuracy'],label='validation accuracy')
plt.legend()
plt.title('Train acc v/s Validation acc')
```

```
Text(0.5, 1.0, 'Train acc v/s Validation acc')
```




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

```
313/313 [==============================] - 1s 2ms/step
```

```
print(y_test[0])
y_pred[0]
for i in range(10):
  print('Digit ',i,' probability ', y_pred[0][i])
```

```
9
Digit  0  probability  2.989326e-09
Digit  1  probability  1.8071179e-12
Digit  2  probability  1.0312593e-11
Digit  3  probability  9.180091e-11
Digit  4  probability  6.3363247e-13
Digit  5  probability  0.00034762218
Digit  6  probability  1.4115183e-10
Digit  7  probability  0.001556198
Digit  8  probability  5.7087608e-08
```

```
Digit  9  probability  0.9980961
```

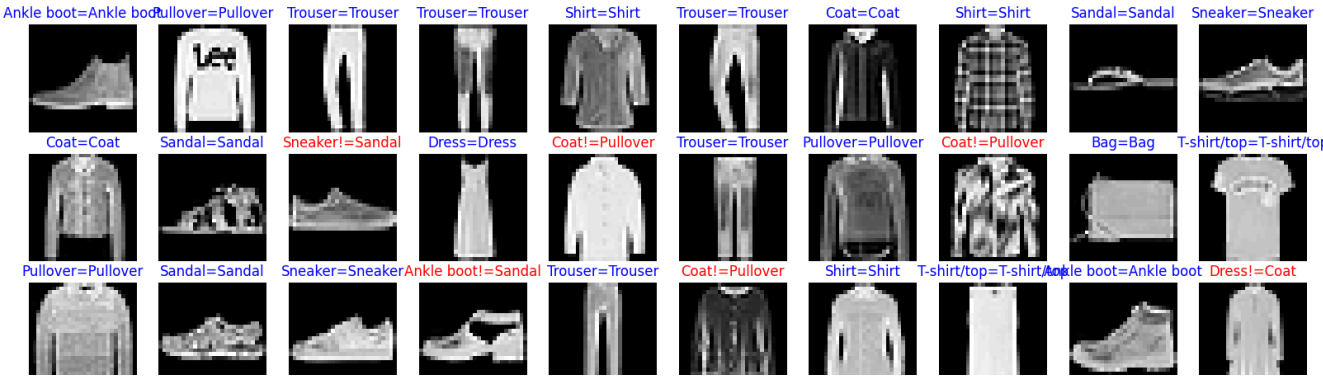
```
import numpy as np
print(np.argmax(y_pred[0]))
y_label=np.argmax(y_pred,axis=1)
```

```
9
```

```
###### 1st sample
i=100
print('True label=',class_label[y_test[i]])
print('Predicted label=',class_label[y_label[i]])
plt.figure(figsize=(1,1))
plt.imshow(x_test[i],cmap='gray')
plt.axis('off')
if y_test[i]==y_label[i]:
  plt.title('correct prediction',c='blue')
else:
  plt.title('Wrong prediction',c='red')
```

```
True label= Dress
Predicted label= Dress
```


```
plt.figure(figsize=(20,20))
for i in range(100):
  plt.subplot(10,10,i+1)
  plt.imshow(x_test[i],cmap='gray')
  plt.axis('off')
  if y_test[i]==y_label[i]:
    plt.title(class_label[y_test[i]]+'='+class_label[y_label[i]],c='blue')
  else:
    plt.title(class_label[y_test[i]]+'!='+class_label[y_label[i]],c='red')
```




```
# Generate a confusion matrix for the test dataset.
cm = tf.math.confusion_matrix(labels=y_test, predictions=y_label)

# Plot the confusion matrix as a heatmap.
plt.figure(figsize=(10,10))
import seaborn as sn
sn.heatmap(cm, annot=True, fmt="d", annot_kws={"size": 14})
plt.xlabel("Predicted")
plt.ylabel("Truth/Target")
plt.show()
```