we train the model on the mnist dataset to predict new given number value

```
#importing the relevent packages.
import numpy as np
import pandas as pd
import tensorflow as tf
import tensorflow_datasets as tfds
```

```
#load the dataset
mnist_data,mnist_info=tfds.load(name='mnist',with_info=True,as_supervised=True)
```

```
mnist_train,mnist_test=mnist_data['train'],mnist_data['test']

mnist_val=0.1*mnist_info.splits['train'].num_examples
mnist_val=tf.cast(mnist_val,tf.int64)

test_sample=mnist_info.splits['test'].num_examples
test_sample=tf.cast(test_sample,tf.int64)
```

```
def scale(image,label):
  image=tf.cast(image,tf.float32)
  image/=255.
  return image,label
```

```
scale_train_val=mnist_train.map(scale)
scale_test=mnist_test.map(scale)
```

```
buffer=1000
shuffer_train_and_val=scale_train_val.shuffle(buffer)
val_data=shuffer_train_and_val.take(mnist_val)
train_data=shuffer_train_and_val.skip(mnist_val)
```

```
BATCH_SIZE=100
train_data=train_data.batch(BATCH_SIZE)
val_data=val_data.batch(mnist_val)
val_inputs,val_targets=next(iter(val_data))
```

model building

there are 784 input layer and 10 output nodes

```
inputs_size=784
outputs_size=10
hidden_layer_size=50
```

```
model=tf.keras.Sequential([
        tf.keras.layers.Flatten(input_shape=(28,28,1)),
        tf.keras.layers.Dense(hidden_layer_size,activation='relu'),
        tf.keras.layers.Dense(hidden_layer_size,activation='relu'),
        tf.keras.layers.Dense(outputs_size,activation='softmax')
    ])


#run the model and fit the optimizer
model.compile(optimizer='Adam',loss='sparse_categorical_crossentropy',metrics=['accurac


epoch=20
model.fit(train_data,epochs=epoch,validation_data=(val_inputs,val_targets),verbose=2)
```

```
Epoch 1/20
540/540 - 7s - loss: 0.4157 - accuracy: 0.8813 - val_loss: 0.2033 - val_accuracy:
Epoch 2/20
540/540 - 5s - loss: 0.1757 - accuracy: 0.9483 - val_loss: 0.1595 - val_accuracy:
Epoch 3/20
540/540 - 5s - loss: 0.1316 - accuracy: 0.9612 - val_loss: 0.1462 - val_accuracy:
Epoch 4/20
540/540 - 5s - loss: 0.1051 - accuracy: 0.9692 - val_loss: 0.1305 - val_accuracy:
Epoch 5/20
540/540 - 5s - loss: 0.0885 - accuracy: 0.9737 - val_loss: 0.1268 - val_accuracy:
Epoch 6/20
540/540 - 5s - loss: 0.0741 - accuracy: 0.9778 - val_loss: 0.1195 - val_accuracy:
Epoch 7/20
540/540 - 5s - loss: 0.0643 - accuracy: 0.9815 - val_loss: 0.1211 - val_accuracy:
Epoch 8/20
540/540 - 5s - loss: 0.0557 - accuracy: 0.9832 - val_loss: 0.1164 - val_accuracy:
Epoch 9/20
540/540 - 5s - loss: 0.0487 - accuracy: 0.9855 - val_loss: 0.1062 - val_accuracy:
Epoch 10/20
540/540 - 4s - loss: 0.0427 - accuracy: 0.9874 - val_loss: 0.1025 - val_accuracy:
Epoch 11/20
540/540 - 5s - loss: 0.0384 - accuracy: 0.9884 - val_loss: 0.1059 - val_accuracy:
Epoch 12/20
540/540 - 4s - loss: 0.0345 - accuracy: 0.9902 - val_loss: 0.1088 - val_accuracy:
Epoch 13/20
540/540 - 5s - loss: 0.0316 - accuracy: 0.9905 - val_loss: 0.1036 - val_accuracy:
Epoch 14/20
540/540 - 4s - loss: 0.0282 - accuracy: 0.9916 - val_loss: 0.1089 - val_accuracy:
Epoch 15/20
540/540 - 5s - loss: 0.0250 - accuracy: 0.9928 - val_loss: 0.1083 - val_accuracy:
Epoch 16/20
540/540 - 4s - loss: 0.0217 - accuracy: 0.9935 - val_loss: 0.1098 - val_accuracy:
Epoch 17/20
540/540 - 5s - loss: 0.0187 - accuracy: 0.9948 - val_loss: 0.1098 - val_accuracy:
Epoch 18/20
540/540 - 5s - loss: 0.0200 - accuracy: 0.9944 - val_loss: 0.1230 - val_accuracy:
Epoch 19/20
540/540 - 4s - loss: 0.0160 - accuracy: 0.9958 - val_loss: 0.1187 - val_accuracy:
Epoch 20/20
540/540 - 4s - loss: 0.0163 - accuracy: 0.9952 - val_loss: 0.1154 - val_accuracy:
<keras.callbacks.History at 0x7f77a500a910>
```

```
#test our model
test_loss,test_accuracy=model.evaluate(scale_test)


print("Test loss : {0:2f}. Test Accuracy : {1:2f}% ".format(test_loss,test_accuracy*100
```


0s completed at 13:32