```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
Brain=Sequential([
     Dense(units=9,activation='relu',input_shape=[8]),
     Dense(units=9,activation='relu'),
     Dense(units=9,activation='relu'),
     Dense(units=1)


])
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from google.colab import auth
import gspread
from google.auth import default
```

```
auth.authenticate_user()
creds, _ = default()
gc = gspread.authorize(creds)
```

```
worksheet=gc.open('Untitled spreadsheet').sheet1
data=worksheet.get_all_values()
```

```
dataset1=pd.DataFrame(data[1:],columns=data[0])
dataset1=dataset1.astype({'Input':float})
dataset1=dataset1.astype({'Output':float})
dataset1.head()
```

	Input	Output
0	1.0	98.0
1	2.0	97.0
2	3.0	82.0
3	4.0	83.0
4	5.0	94.0

```
X=dataset1[['Input']].values
y=dataset1[['Output']].values
```

Double-click (or enter) to edit

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.33,random_state=33)
```

```
Scaler=MinMaxScaler()
Scaler.fit(X_train)
X_train1=Scaler.transform(X_train)
```

```
ai_brain=Sequential([
     Dense(units=8,activation='relu'),
     Dense(units=10,activation='relu'),
     Dense(1)
])
```

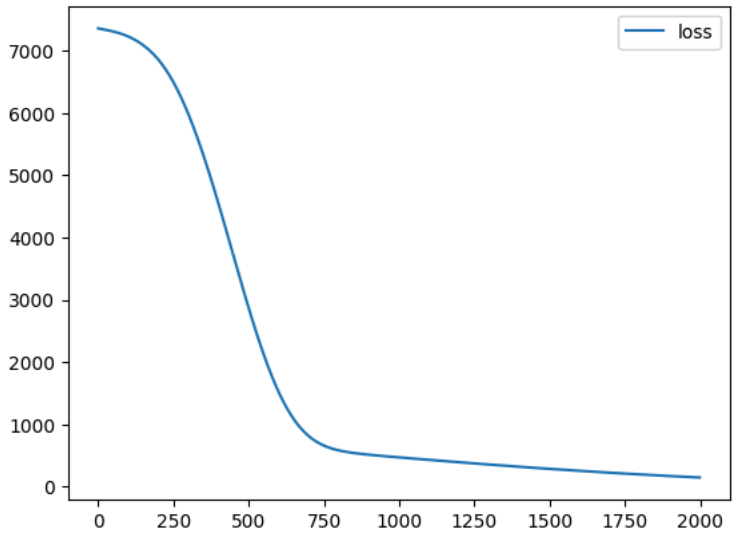
```
ai_brain.compile(optimizer='adam',loss='mse')
```

```
ai_brain.fit(X_train1,y_train,epochs=2000)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step - loss: 3481.8687
Epoch 465/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 3465.0251
Epoch 466/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 3448.1931
Epoch 467/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 3431.3740
Epoch 468/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - loss: 3414.5676
Epoch 469/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - loss: 3397.7747
Epoch 470/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - loss: 3380.9961
Epoch 471/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 3364.2319
Epoch 472/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - loss: 3347.4839
Epoch 473/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - loss: 3330.7512
Epoch 474/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 3314.0352
Epoch 475/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - loss: 3297.3372
Epoch 476/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - loss: 3280.6560
Epoch 477/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - loss: 3263.9939
Epoch 478/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - loss: 3247.3511
Epoch 479/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 3230.7278
Epoch 480/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 3214.1245
Epoch 481/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 3197.5425
Epoch 482/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - loss: 3180.9819
Epoch 483/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - loss: 3164.4436
Epoch 484/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - loss: 3147.9275
Epoch 485/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 3131.4351
Epoch 486/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 3114.9670
Epoch 487/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - loss: 3098.5229
Epoch 488/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 3082.1042
Epoch 489/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 3065.7114
Epoch 490/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 3049.3450
Epoch 491/2000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 3033.0056
```

```
loss_df=pd.DataFrame(ai_brain.history.history)
loss_df.plot()
```

```
<Axes: >
```




```
X_test1=Scaler.transform(X_test)
ai_brain.evaluate(X_test1,y_test)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 189ms/step - loss: 211.1086
211.108642578125
```

```
X_n1=[[4]]
X_n1=Scaler.transform(X_n1)
ai_brain.predict(X_n1)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step
array([[76.86949]], dtype=float32)
```

Start coding or generate with AI.

EXP 3

```
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import mnist
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import utils
import pandas as pd
from sklearn.metrics import classification_report,confusion_matrix
from tensorflow.keras.preprocessing import image
```

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
X_train.shape
```

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

```
X_test.shape
```

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

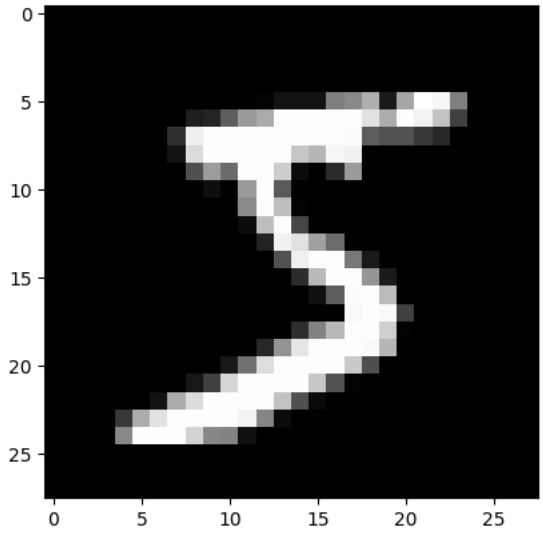
```
single_image= X_train[0]
```

```
single_image.shape
```

```
(28, 28)
```

```
plt.imshow(single_image,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7d8206628a60>
```




```
y_train.shape
```

```
(60000,)
```

```
X_train.min()
```

```
0
```

```
X_train.max()
```

```
255
```

```
X_train_scaled = X_train/255.0
X_test_scaled = X_test/255.0
```

```
5
```

Start coding or generate with AI.

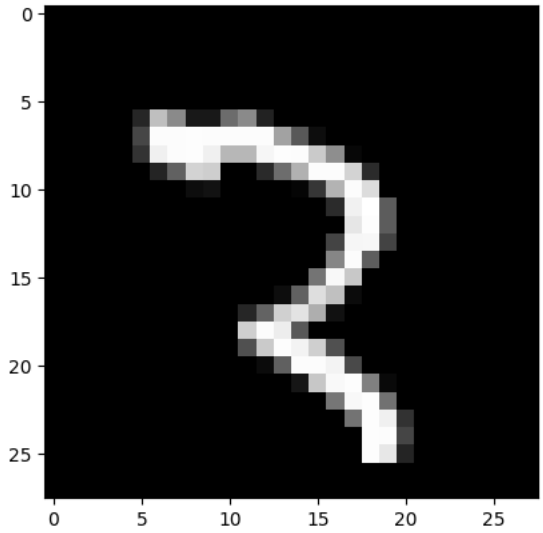
Start coding or generate with AI.

```
numpy.ndarray
```

```
(60000, 10)
```

Start coding or generate with AI.

```
<matplotlib.image.AxesImage at 0x7d8205de0b80>
```



Start coding or generate with AI.

```
array([0., 0., 0., 1., 0., 0., 0., 0., 0., 0.])
```

Start coding or generate with AI.

Start coding or generate with AI.

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 64)	18,496
flatten_1 (Flatten)	(None, 7744)	0
dense_2 (Dense)	(None, 32)	247,840
dense_3 (Dense)	(None, 10)	330

Total params: 266,986 (1.02 MB)
Trainable params: 266,986 (1.02 MB)
Non-trainable params: 0 (0.00 B)

Start coding or generate with AI.

Start coding or generate with AI.

```
Epoch 1/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 60s 62ms/step - accuracy: 0.8862 - loss: 0.3714 - val_accuracy: 0.9827 - val_loss: 0.0535
Epoch 2/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 54s 58ms/step - accuracy: 0.9847 - loss: 0.0511 - val_accuracy: 0.9885 - val_loss: 0.0358
Epoch 3/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 85s 61ms/step - accuracy: 0.9910 - loss: 0.0313 - val_accuracy: 0.9878 - val_loss: 0.0369
Epoch 4/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 63s 67ms/step - accuracy: 0.9920 - loss: 0.0249 - val_accuracy: 0.9896 - val_loss: 0.0320
Epoch 5/5
938/938 ━━━━━━━━━━━━━━━━━━━━ 73s 57ms/step - accuracy: 0.9949 - loss: 0.0159 - val_accuracy: 0.9865 - val_loss: 0.0467
<keras.src.callbacks.history.History at 0x7d820587d270>
```

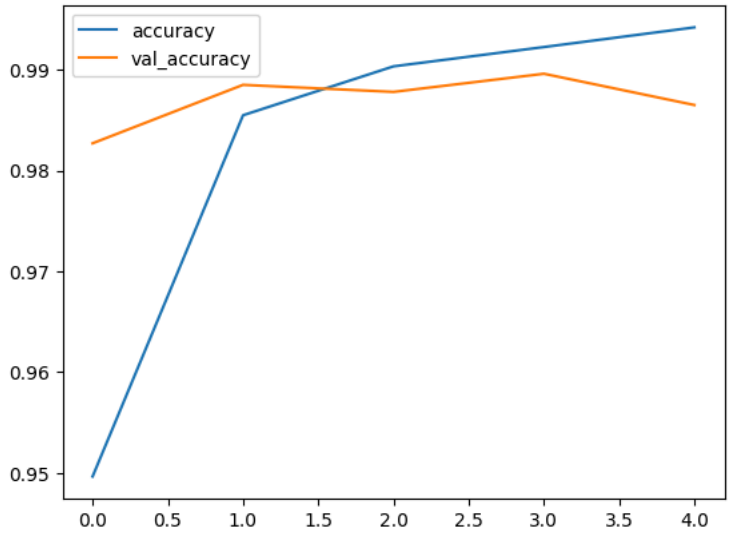
Start coding or generate with AI.

Start coding or generate with AI.

	accuracy	loss	val_accuracy	val_loss
0	0.949650	0.166132	0.9827	0.053535
1	0.985483	0.047670	0.9885	0.035774
2	0.990333	0.032237	0.9878	0.036949
3	0.992250	0.024217	0.9896	0.031987
4	0.994200	0.017570	0.9865	0.046662

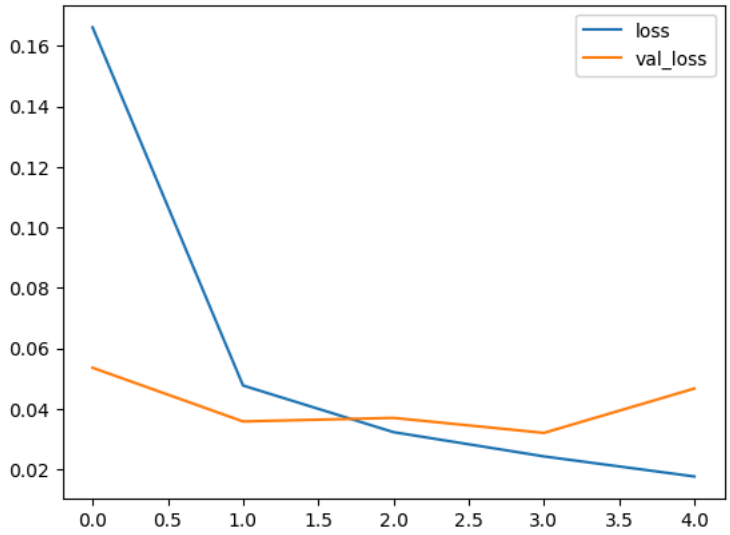
Start coding or generate with AI.

```
<Axes: >
```



Start coding or generate with AI.

```
<Axes: >
```



Start coding or generate with AI.

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step
```

Start coding or generate with AI.

```
[[ 975    1    0    0    1    1    1    1    0    0]
 [    0 1130    0    1    1    0    2    0    0    1]
 [    0    4 1009    1    1    0    0   16    1    0]
 [    0    1    0  993    0    9    0    6    1    0]
 [    0    0    0    0  971    0    0    2    0    9]
 [    0    0    0    2    0  889    1    0    0    0]
 [    3    2    0    0    1   14  936    0    2    0]
 [    0    2    1    0    0    0    0 1022    1    2]
 [    7    0    0    5    0    3    1    4  949    5]
 [    1    2    1    1    4    7    0    2    0  991]]
```

Start coding or generate with AI.

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       0.99      1.00      0.99      1135
           2       1.00      0.98      0.99      1032
           3       0.99      0.98      0.99      1010
           4       0.99      0.99      0.99       982
           5       0.96      1.00      0.98       892
           6       0.99      0.98      0.99       958
           7       0.97      0.99      0.98      1028
           8       0.99      0.97      0.98       974
           9       0.98      0.98      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

Start coding or generate with AI.

```
PIL.WebPImagePlugin.WebPImageFile
def __init__(fp=None, filename=None)
```

Base class for image file format handlers.

Start coding or generate with AI.

Start coding or generate with AI.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
```

Start coding or generate with AI.

```
[0]
```

Start coding or generate with AI.

```
<matplotlib.image.AxesImage at 0x7d82049e8400>
```