```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
import csv
from keras.layers import *
from keras import optimizers
```

```
o=pd.read_csv('/content/sample_data/mnist_train_small.csv')
p=pd.read_csv('/content/sample_data/mnist_test.csv')
```

```
o.head(10)
```

	6	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	...	0.581	0.582	0.583	0.584	0.585	0.586	0.587	0.588	0.589	0.590
0	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6	6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

10 rows × 785 columns

```
p.head()
```

	7	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	...	0.658	0.659	0.660	0.661	0.662	0.663	0.664	0.665	0.666	0.667
0	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

```
o.shape
```

```
(19999, 785)
```

```
f=o.iloc[:,1:785]
l=o.iloc[:,0]
```

```
set(l)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
sum(f.isnull().sum())
```

```
0
```

```
from sklearn.model_selection import train_test_split
X_train, X_cv, y_train, y_cv = train_test_split(f, l, test_size = 0.2)
```

```
X_train = np.array(X_train)
X_cv = np.array(X_cv)
```

```
print((min(X_train[1]), max(X_train[1])))
```

```
(0, 255)
```

```
X_train.shape
```

```
(15999, 784)
```

```
model=Sequential([Dense(units=300, activation='relu'),
                  Dense(units=150, activation='relu'),
                  Dense(units=180, activation='linear'),
                  Dense(10, activation='softmax')])
```

```
model.compile(tf.keras.optimizers.Adam(learning_rate=0.001),loss='SparseCategoricalCrossentropy',metrics=['accuracy'])
```

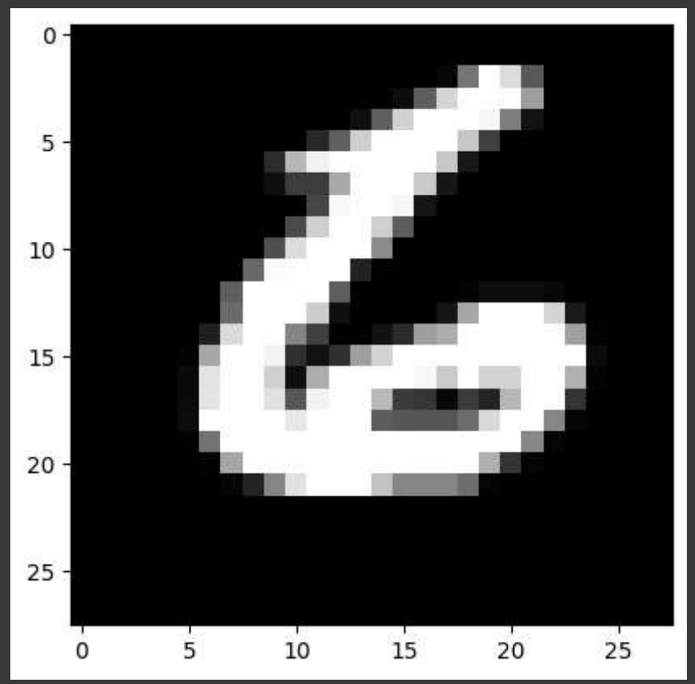
```
hist=model.fit(X_train, y_train,epochs=50,validation_data=(X_cv, y_cv))
```

```
Epoch 28/50
500/500 [==============================] - 4s 8ms/step - loss: 0.0968 - accuracy: 0.9763 - val_loss: 0.4384 - val_accuracy: 0.9540
Epoch 29/50
500/500 [==============================] - 4s 9ms/step - loss: 0.1114 - accuracy: 0.9779 - val_loss: 0.6575 - val_accuracy: 0.9433
Epoch 30/50
500/500 [==============================] - 6s 11ms/step - loss: 0.0959 - accuracy: 0.9772 - val_loss: 0.4663 - val_accuracy: 0.9477
Epoch 31/50
500/500 [==============================] - 4s 9ms/step - loss: 0.0603 - accuracy: 0.9844 - val_loss: 0.4624 - val_accuracy: 0.9523
Epoch 32/50
500/500 [==============================] - 4s 8ms/step - loss: 0.0597 - accuracy: 0.9862 - val_loss: 0.5111 - val_accuracy: 0.9520
Epoch 33/50
500/500 [==============================] - 6s 12ms/step - loss: 0.0577 - accuracy: 0.9859 - val_loss: 0.4509 - val_accuracy: 0.9557
Epoch 34/50
500/500 [==============================] - 6s 12ms/step - loss: 0.0636 - accuracy: 0.9856 - val_loss: 0.5409 - val_accuracy: 0.9532
Epoch 35/50
500/500 [==============================] - 5s 10ms/step - loss: 0.0926 - accuracy: 0.9822 - val_loss: 0.3755 - val_accuracy: 0.9530
Epoch 36/50
500/500 [==============================] - 6s 12ms/step - loss: 0.0638 - accuracy: 0.9849 - val_loss: 0.3976 - val_accuracy: 0.9530
Epoch 37/50
500/500 [==============================] - 4s 9ms/step - loss: 0.0465 - accuracy: 0.9882 - val_loss: 0.5228 - val_accuracy: 0.9523
Epoch 38/50
500/500 [==============================] - 6s 12ms/step - loss: 0.0505 - accuracy: 0.9878 - val_loss: 0.8176 - val_accuracy: 0.9467
Epoch 39/50
500/500 [==============================] - 4s 9ms/step - loss: 0.0870 - accuracy: 0.9829 - val_loss: 0.4667 - val_accuracy: 0.9530
Epoch 40/50
500/500 [==============================] - 4s 9ms/step - loss: 0.0545 - accuracy: 0.9892 - val_loss: 0.4236 - val_accuracy: 0.9575
Epoch 41/50
500/500 [==============================] - 7s 13ms/step - loss: 0.0433 - accuracy: 0.9899 - val_loss: 0.4174 - val_accuracy: 0.9555
Epoch 42/50
500/500 [==============================] - 4s 9ms/step - loss: 0.0422 - accuracy: 0.9894 - val_loss: 0.5175 - val_accuracy: 0.9565
Epoch 43/50
500/500 [==============================] - 4s 9ms/step - loss: 0.0520 - accuracy: 0.9884 - val_loss: 0.5389 - val_accuracy: 0.9492
Epoch 44/50
500/500 [==============================] - 5s 11ms/step - loss: 0.0675 - accuracy: 0.9847 - val_loss: 0.4956 - val_accuracy: 0.9510
Epoch 45/50
500/500 [==============================] - 4s 8ms/step - loss: 0.0487 - accuracy: 0.9901 - val_loss: 0.6096 - val_accuracy: 0.9455
Epoch 46/50
500/500 [==============================] - 4s 9ms/step - loss: 0.0618 - accuracy: 0.9876 - val_loss: 0.4406 - val_accuracy: 0.9515
Epoch 47/50
500/500 [==============================] - 6s 12ms/step - loss: 0.0343 - accuracy: 0.9917 - val_loss: 0.4321 - val_accuracy: 0.9557
Epoch 48/50
500/500 [==============================] - 4s 9ms/step - loss: 0.0349 - accuracy: 0.9914 - val_loss: 0.4968 - val_accuracy: 0.9548
Epoch 49/50
500/500 [==============================] - 5s 10ms/step - loss: 0.0739 - accuracy: 0.9886 - val_loss: 0.5425 - val_accuracy: 0.9405
Epoch 50/50
500/500 [==============================] - 5s 10ms/step - loss: 0.0638 - accuracy: 0.9859 - val_loss: 0.4255 - val_accuracy: 0.9553
```

```
x_test=p.iloc[:,0:784]
ans=model.predict(x_test)
ans=list(ans)
for x in range(len(ans)):
  ans[x]=list(ans[x])
for x in range(len(ans)):
  ans[x]=ans[x].index(max(ans[x]))
ans=np.array(ans)
ans
```

```
313/313 [==============================] - 1s 3ms/step
array([2, 1, 0, ..., 4, 5, 6])
```

```
sample2=-1
image = X_test[sample2]
fig = plt.figure
plt.imshow(image, cmap='gray')
plt.show()
```



Start coding or generate with AI.