In [4]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

In [5]:
```
columns = ["lettr", "x-box", "y-box", "width", "height", "onpix", "x-bar","y-bar", "x2bar", "y2bar", "xybar",
```

In [9]:
```
df = pd.read_csv(r"C:\Users\PRANAV\Downloads\letter.data", names=columns)
```

In [14]:
```
df
```

Out[14]:

	lettr	x-box	y-box	width	height	onpix	x-bar	y-bar	x2bar	y2bar	xybar	x2ybr	xy2br	x-ege	xegvy	y-ege	yegvx
0	T	2	8	3	5	1	8	13	0	6	6	10	8	0	8	0	8
1	I	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10
2	D	4	11	6	8	6	10	6	2	6	10	3	7	3	7	3	9
3	N	7	11	6	6	3	5	9	4	6	4	4	10	6	10	2	8
4	G	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19995	D	2	2	3	3	2	7	7	7	6	6	6	4	2	8	3	7
19996	C	7	10	8	8	4	4	8	6	9	12	9	13	2	9	3	7
19997	T	6	9	6	7	5	6	11	3	7	11	9	5	2	12	2	4
19998	S	2	3	4	2	1	8	7	2	6	10	6	8	1	9	5	8
19999	A	4	9	6	6	2	9	5	3	1	8	1	8	2	7	2	8

In [15]:
```
x = df.drop("lettr", axis=1).values
y = df["lettr"].values
```

In [16]:
```
x.shape
```

Out[16]:
```
(20000, 16)
```

In [17]:
```
y.shape
```

Out[17]:
```
(20000,)
```

In [18]:
```
np.unique(y)
```

Out[18]:
```
array(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
       'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'],
      dtype=object)
```

In [19]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

In [20]:
```
def shape():
    print("Train Shape :",x_train.shape)
    print("Test Shape :",x_test.shape)
    print("y_train shape :",y_train.shape)
    print("y_test shape :",y_test.shape)

shape()
```

```
Train Shape : (16000, 16)
Test Shape : (4000, 16)
y_train shape : (16000,)
y_test shape : (4000,)
```

In [21]:
```
 x_train[0]
```

Out[21]:
```
array([ 6, 10,  8,  8,  7,  7,  4,  8,  4,  6,  7, 10,  5,  7,  7,  9],
      dtype=int64)
```

In [22]:
```
y_train[0]
```

Out[22]:
```
'Q'
```

In [23]:
```
class_names=['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M','N', 'O', 'P', 'Q', 'R', 'S', '
```

In [24]:
```
x_test[10]
```

Out[24]:
```
array([ 3,  3,  3,  5,  1,  7,  5, 13,  5,  7, 14,  8,  3,  9,  0,  8],
      dtype=int64)
```

In [25]:
```
y_test[10]
```

Out[25]:
```
'U'
```

In [26]:
```
x_train = x_train/255
x_test = x_test/255
```

In [27]:
```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

y_train = encoder.fit_transform(y_train)
y_test = encoder.fit_transform(y_test)
```

In [28]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

In [29]:
```
model=Sequential()

model.add(Dense(512, activation='relu', input_shape=(16,)))
model.add(Dropout(0.2))

model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(26, activation='softmax'))

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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               8704

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 26)                6682

=================================================================
Total params: 146,714
Trainable params: 146,714
Non-trainable params: 0
_________________________________________________________________
```

In [30]:
```
model.fit(x_train, y_train, epochs=50, batch_size=128, verbose=1,validation_data=(x_test, y_test))
```

```
322 - val_accuracy: 0.5188
Epoch 5/50
125/125 [==============================] - 1s 9ms/step - loss: 1.6394 - accuracy: 0.4999 - val_loss: 1.5
203 - val_accuracy: 0.5558
Epoch 6/50
125/125 [==============================] - 1s 9ms/step - loss: 1.5405 - accuracy: 0.5364 - val_loss: 1.4
453 - val_accuracy: 0.5745
Epoch 7/50
125/125 [==============================] - 1s 9ms/step - loss: 1.4779 - accuracy: 0.5577 - val_loss: 1.3
735 - val_accuracy: 0.6080
Epoch 8/50
125/125 [==============================] - 1s 9ms/step - loss: 1.4108 - accuracy: 0.5794 - val_loss: 1.3
158 - val_accuracy: 0.6270
Epoch 9/50
125/125 [==============================] - 1s 9ms/step - loss: 1.3523 - accuracy: 0.6022 - val_loss: 1.2
574 - val_accuracy: 0.6405
Epoch 10/50
125/125 [==============================] - 1s 9ms/step - loss: 1.3027 - accuracy: 0.6186 - val_loss: 1.1
993 - val_accuracy: 0.6625
Epoch 11/50
```

In [31]:
```
predictions = model.predict(x_test)
```

```
125/125 [==============================] - 1s 3ms/step
```

In [32]:
```
index=10

print(predictions[index])

final_value=np.argmax(predictions[index])

print("Actual label :",y_test[index])
print("Predicted label :",final_value)
print("Class (A-Z) :",class_names[final_value])
```

```
[3.8210567e-11 2.3441545e-27 1.7369923e-05 4.3192802e-12 3.9025032e-19
 9.3194992e-12 8.0663282e-11 5.5684964e-06 1.0162436e-12 1.5553801e-10
 5.0411372e-16 2.6287056e-12 1.0553427e-15 1.6873156e-10 8.5646207e-05
 1.0225622e-13 6.9662651e-06 1.0563283e-25 1.4153850e-10 3.7627174e-07
 9.9987626e-01 7.5093258e-06 2.2150668e-15 1.9098850e-12 4.6218568e-07
 9.3401656e-22]
Actual label : 20
Predicted label : 20
Class (A-Z) : U
```

In [23]:
```
loss, accuracy = model.evaluate(x_test, y_test)

print("Loss :",loss)
print("Accuracy (Test Data) :",accuracy*100)
```

```
125/125 [==============================] - 0s 2ms/step - loss: 0.4920 - accuracy: 0.8495
Loss : 0.4919666647911072
Accuracy (Test Data) : 84.95000004768372
```