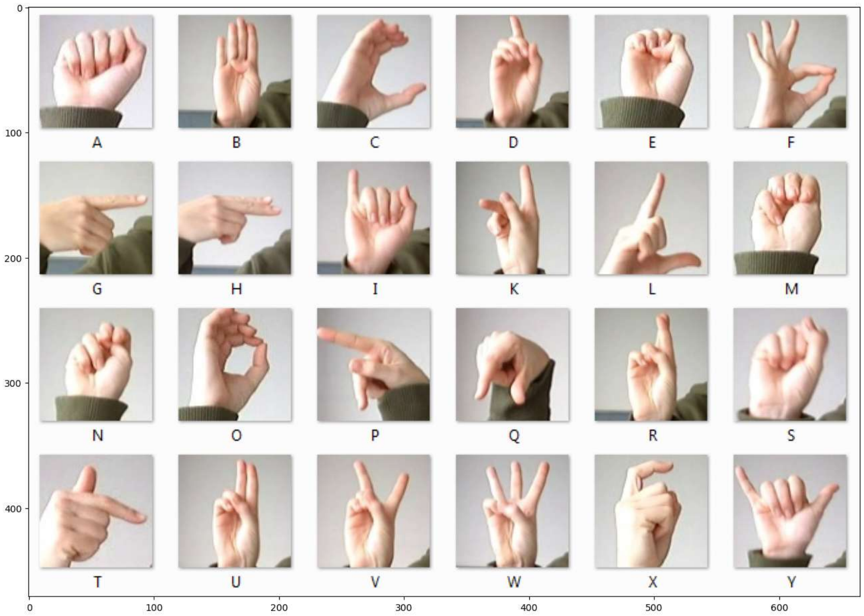
```
import matplotlib.pyplot as plt
plt.figure(figsize = (16,16))
img = plt.imread('/content/amer_sign2.png')
plt.imshow(img)
plt.show()
```



```
import keras
from keras.models import Sequential
from keras.layers import Dense,Flatten,Conv2D,MaxPool2D,Dropout
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import pandas as pd
```

```
train_df=pd.read_csv('/content/sign_mnist_train.csv')
test_df=pd.read_csv('/content/sign_mnist_test.csv')
```

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27455 entries, 0 to 27454
Columns: 785 entries, label to pixel784
dtypes: int64(785)
memory usage: 164.4 MB
```

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7172 entries, 0 to 7171
Columns: 785 entries, label to pixel784
dtypes: int64(785)
memory usage: 43.0 MB
```

```
train_df.describe()
```

	label	pixel1	pixel2	pixel3	pixel4	pixe
count	27455.000000	27455.000000	27455.000000	27455.000000	27455.000000	27455.0000
mean	12.318813	145.419377	148.500273	151.247714	153.546531	156.2108
std	7.287552	41.358555	39.942152	39.056286	38.595247	37.1111
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	6.000000	121.000000	126.000000	130.000000	133.000000	137.0000
50%	13.000000	150.000000	153.000000	156.000000	158.000000	160.0000
75%	19.000000	174.000000	176.000000	178.000000	179.000000	181.0000
max	24.000000	255.000000	255.000000	255.000000	255.000000	255.0000

8 rows × 785 columns

```
train_df.head(6)
```

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...
0	3	107	118	127	134	139	143	146	150	153	...
1	6	155	157	156	156	156	157	156	158	158	...
2	2	187	188	188	187	187	186	187	188	187	...
3	2	211	211	212	212	211	210	211	210	210	...
4	13	164	167	170	172	176	179	180	184	185	...
5	16	161	168	172	173	178	184	189	193	196	...

6 rows × 785 columns

```
train_label=train_df['label']
train_label.head()
trainset=train_df.drop(['label'],axis=1)
trainset.head()
```

	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	..
0	107	118	127	134	139	143	146	150	153	156	
1	155	157	156	156	156	157	156	158	158	157	
2	187	188	188	187	187	186	187	188	187	186	
3	211	211	212	212	211	210	211	210	210	211	
4	164	167	170	172	176	179	180	184	185	186	

5 rows × 784 columns

```
X_train = trainset.values
X_train = trainset.values.reshape(-1,28,28,1)
print(X_train.shape)
```

```
    (27455, 28, 28, 1)
```

```
test_label=test_df['label']
X_test=test_df.drop(['label'],axis=1)
print(X_test.shape)
X_test.head()
```

```
(7172, 784)
```

	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	..
0	149	149	150	150	150	151	151	150	151	152	
1	126	128	131	132	133	134	135	135	136	138	
2	85	88	92	96	105	123	135	143	147	152	
3	203	205	207	206	207	209	210	209	210	209	
4	188	191	193	195	199	201	202	203	203	203	

```
from sklearn.preprocessing import LabelBinarizer
lb=LabelBinarizer()
y_train=lb.fit_transform(train_label)
y_test=lb.fit_transform(test_label)
y_train
```

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

```
X_test=X_test.values.reshape(-1,28,28,1)
print(X_train.shape,y_train.shape,X_test.shape,y_test.shape)
```

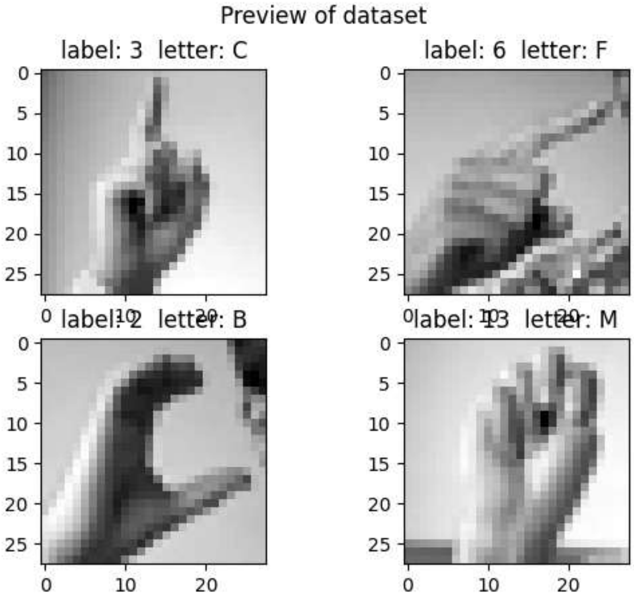
```
(27455, 28, 28, 1) (27455, 24) (7172, 28, 28, 1) (7172, 24)
```

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                  rotation_range = 0,
                                  height_shift_range=0.2,
                                  width_shift_range=0.2,
                                  shear_range=0,
                                  zoom_range=0.2,
                                  horizontal_flip=True,
                                  fill_mode='nearest')
```

```
X_test=X_test/255
```

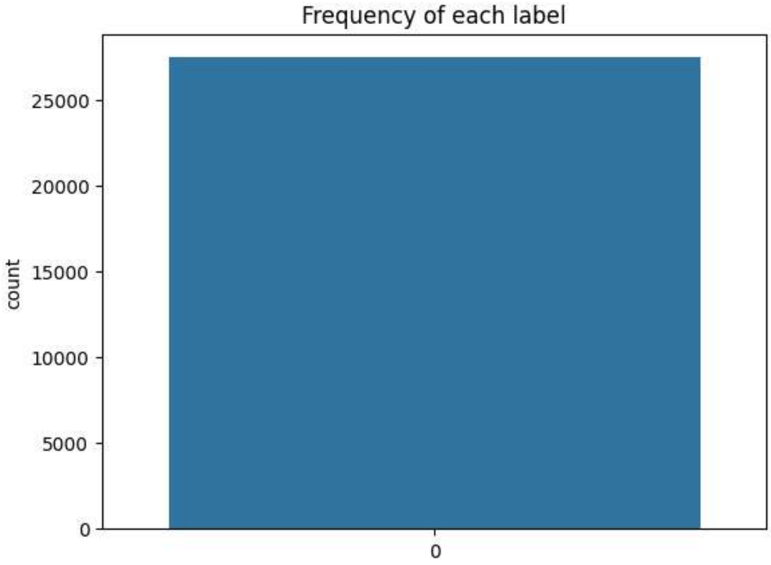
```
fig,axe=plt.subplots(2,2)
fig.suptitle('Preview of dataset')
axe[0,0].imshow(X_train[0].reshape(28,28),cmap='gray')
axe[0,0].set_title('label: 3  letter: C')
axe[0,1].imshow(X_train[1].reshape(28,28),cmap='gray')
axe[0,1].set_title('label: 6  letter: F')
axe[1,0].imshow(X_train[2].reshape(28,28),cmap='gray')
axe[1,0].set_title('label: 2  letter: B')
axe[1,1].imshow(X_train[4].reshape(28,28),cmap='gray')
axe[1,1].set_title('label: 13  letter: M')
```

```
Text(0.5, 1.0, 'label: 13  letter: M')
```



```
sns.countplot(train_label)
plt.title("Frequency of each label")
```

```
Text(0.5, 1.0, 'Frequency of each label')
```



```
model=Sequential()
model.add(Conv2D(128,kernel_size=(5,5),
                 strides=1,padding='same',activation='relu',input_shape=(28,28,1)))
model.add(MaxPool2D(pool_size=(3,3),strides=2,padding='same'))
model.add(Conv2D(64,kernel_size=(2,2),
                strides=1,activation='relu',padding='same'))
model.add(MaxPool2D((2,2),2,padding='same'))
model.add(Conv2D(32,kernel_size=(2,2),
                strides=1,activation='relu',padding='same'))
model.add(MaxPool2D((2,2),2,padding='same'))

model.add(Flatten())


model.add(Dense(units=512,activation='relu'))
model.add(Dropout(rate=0.25))
model.add(Dense(units=24,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 128)       3328

 max_pooling2d (MaxPooling2D  (None, 14, 14, 128)      0
 )

 conv2d_1 (Conv2D)           (None, 14, 14, 64)        32832

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 7, 7, 32)          8224

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 32)         0
 2D)

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 512)               262656

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 24)                12312

=================================================================
Total params: 319,352
Trainable params: 319,352
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
model.fit(train_datagen.flow(X_train,y_train,batch_size=200),
         epochs = 10,
          validation_data=(X_test,y_test),
          shuffle=1
         )
```

```
Epoch 1/10
138/138 [==============================] - 25s 87ms/step - loss: 3.0165 - accuracy: 0.0942 - val_loss: 2.4650 - val_accuracy: 0.210
Epoch 2/10
138/138 [==============================] - 12s 84ms/step - loss: 2.3250 - accuracy: 0.2641 - val_loss: 1.5261 - val_accuracy: 0.493
Epoch 3/10
138/138 [==============================] - 11s 80ms/step - loss: 1.7195 - accuracy: 0.4381 - val_loss: 1.1787 - val_accuracy: 0.580
Epoch 4/10
138/138 [==============================] - 11s 80ms/step - loss: 1.3519 - accuracy: 0.5481 - val_loss: 0.8437 - val_accuracy: 0.695
Epoch 5/10
138/138 [==============================] - 15s 105ms/step - loss: 1.1240 - accuracy: 0.6223 - val_loss: 0.7494 - val_accuracy: 0.73
Epoch 6/10
138/138 [==============================] - 11s 81ms/step - loss: 0.9567 - accuracy: 0.6755 - val_loss: 0.6245 - val_accuracy: 0.788
Epoch 7/10
138/138 [==============================] - 11s 80ms/step - loss: 0.8168 - accuracy: 0.7231 - val_loss: 0.4389 - val_accuracy: 0.858
Epoch 8/10
138/138 [==============================] - 11s 78ms/step - loss: 0.7042 - accuracy: 0.7603 - val_loss: 0.3777 - val_accuracy: 0.875
Epoch 9/10
138/138 [==============================] - 11s 80ms/step - loss: 0.6298 - accuracy: 0.7837 - val_loss: 0.3142 - val_accuracy: 0.888
Epoch 10/10
138/138 [==============================] - 11s 79ms/step - loss: 0.5628 - accuracy: 0.8115 - val_loss: 0.2483 - val_accuracy: 0.919
<keras.callbacks.History at 0x791a734ef6d0>
```

```
(ls,acc)=model.evaluate(x=X_test,y=y_test)
```

```
225/225 [==============================] - 1s 3ms/step - loss: 0.2483 - accuracy: 0.9197
```

```
print('MODEL ACCURACY = {}%'.format(acc*100))
```

```
MODEL ACCURACY = 91.96876883506775%
```

✓ 0s completed at 08:24