```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.6/dist-packages (1.5.6)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from kaggle
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages (from kaggle)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.6/dist-packages (from kaggl
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from kaggle) (4
Requirement already satisfied: python-slugify in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from re
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.6/dist-packages (
```

```
from google.colab import files
files.upload()
```

Choose Files kaggle.json

- **kaggle.json**(application/json) - 72 bytes, last modified: 7/14/2020 - 100% done

```
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"nandinireddypolu","key":"c083705854369c414f2a9881df943765"}'}
```

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

```
  !chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d gti-upm/leapgestrecog
```

```
Downloading leapgestrecog.zip to /content
100% 2.12G/2.13G [00:39<00:00, 79.0MB/s]
100% 2.13G/2.13G [00:39<00:00, 58.2MB/s]
```

```
!pip install zip_files
```

```
Collecting zip_files
  Downloading https://files.pythonhosted.org/packages/f3/99/fc7f65a052d30e62b946924ed6334f8f1e
Requirement already satisfied: click in /usr/local/lib/python3.6/dist-packages (from zip_files
Installing collected packages: zip-files
Successfully installed zip-files-0.3.0
```

```
from zipfile import ZipFile
file_name="leapgestrecog.zip"
with ZipFile(file_name,'r') as zip:
  zip.extractall()
  print('Done')
```

```
Done
```

```
import os
import cv2
import pandas as pd
import numpy as np
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tqdm import tqdm
from keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Activation
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint
from sklearn.metrics import confusion_matrix, accuracy_score


data_dir ='leapgestrecog/leapGestRecog/'
main_dir = os.listdir(data_dir)
```

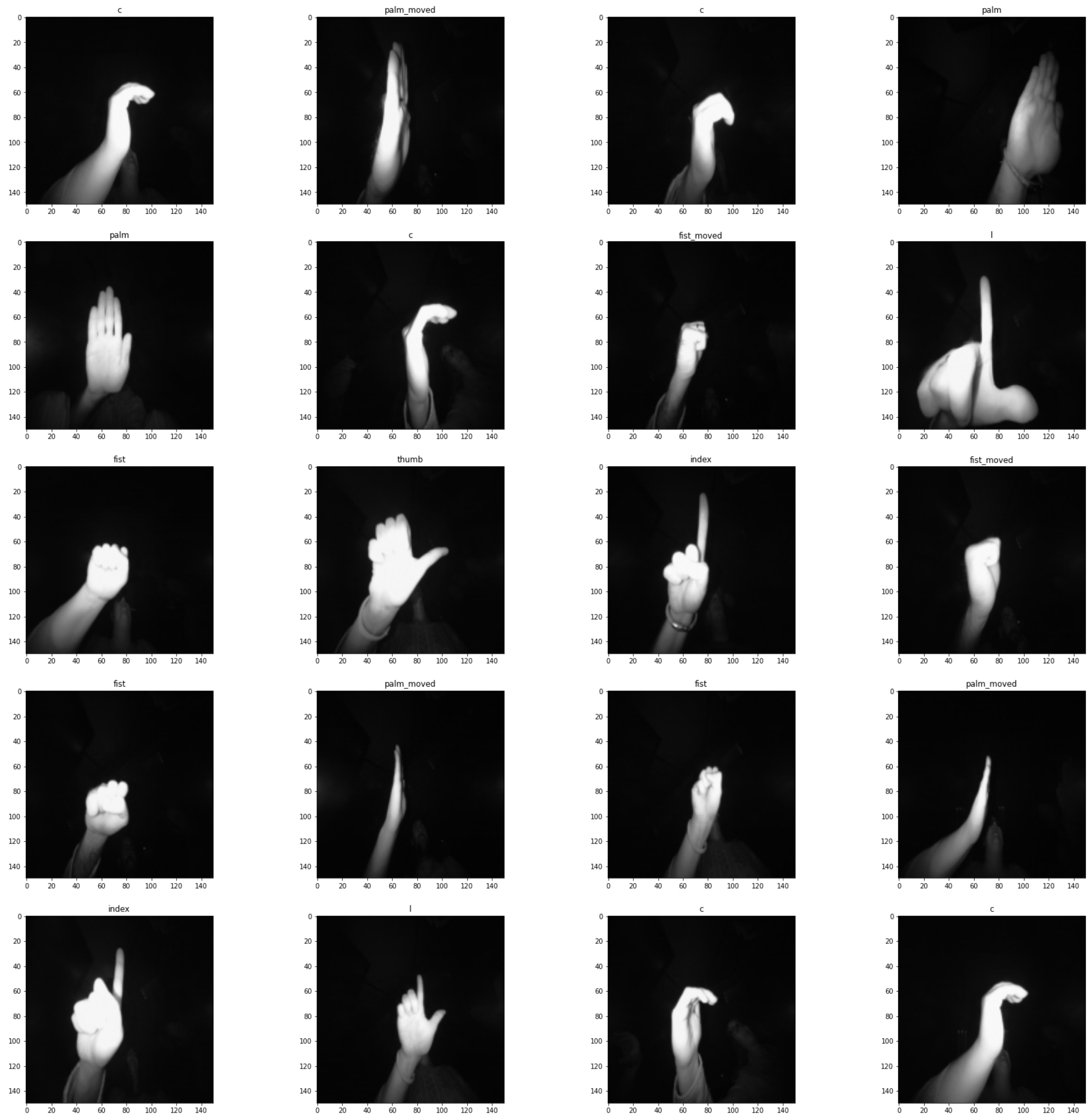
Creating Features & Target Variables

```
x=[]
y=[]

for folder in tqdm(main_dir):
    main_sub_dir = os.listdir(data_dir+folder)
    for subfolder in main_sub_dir:
        gesture = subfolder.split('_',1)[-1]
        images = os.listdir(data_dir+folder+'/'+subfolder+'/')
        for image in images:
            data = cv2.imread(data_dir+folder+'/'+subfolder+'/'+image,cv2.IMREAD_COLOR)
            data = cv2.resize(data,(150,150),interpolation = cv2.INTER_AREA)
            x.append(np.array(data))
            y.append(gesture)
```

```
100%|██████████| 10/10 [00:58<00:00,  5.82s/it]
```

Displaying Images With Labels

```
fig, ax = plt.subplots(5,4,figsize=(30,30))
for i in range(5):
    for j in range(4):
        l = np.random.randint(0,len(x))
        ax[i,j].imshow(x[l])
        ax[i,j].set_title(y[l])
```



Convert Features List into a numpy array

```
x = np.array(x)
```

```
x = np.array(x)
x.shape
```

```
(20000, 150, 150, 3)
```

```
le = LabelEncoder()
y = le.fit_transform(y)
y=to_categorical(y)
```

Creating Train,validation and test set

```
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.80,random_state=21)
x_train, x_val, y_train, y_val = train_test_split(x_train,y_train, test_size=0.20, random_state=21)
```

Defining Model Architecture

```
model = Sequential()
model.add(Conv2D(filters = 32, kernel_size = (4,4),padding = 'Same',activation ='relu', input_shape
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))


model.add(Conv2D(filters =96, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters = 128, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters = 256, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters = 256, kernel_size = (3,3),padding = 'Same',activation ='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dense(10, activation = "softmax"))
```

Instantiating Callbacks

```
callback =ModelCheckpoint('weights.hdf5',verbose=1,monitor='val_accuracy',save_best_only=True)
```

Data Augmentation

```
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True)
```

Compiling the Model

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

```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

Fitting the Model

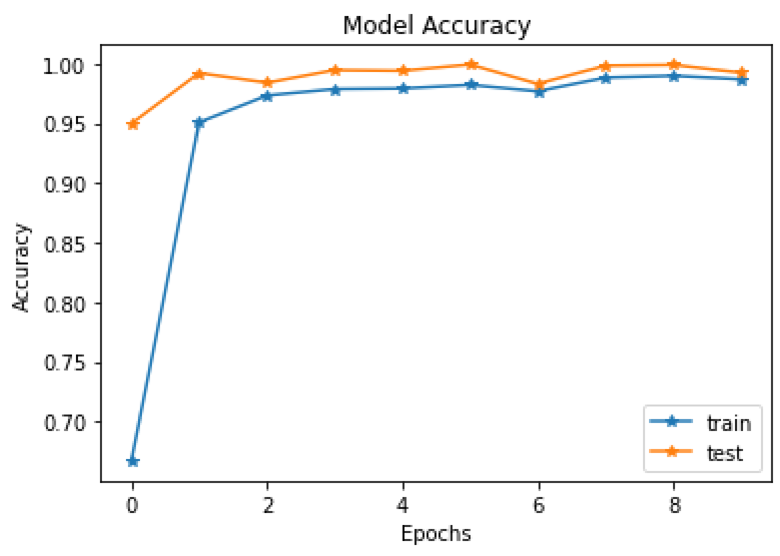
```
conv2d 4 (Conv2D)            (None, 75, 75, 64)        18496
```

```
mymodel=model.fit_generator(datagen.flow(x_train, y_train, batch_size=32),
                    epochs=10,validation_data=(x_val,y_val),callbacks=[callback])
```

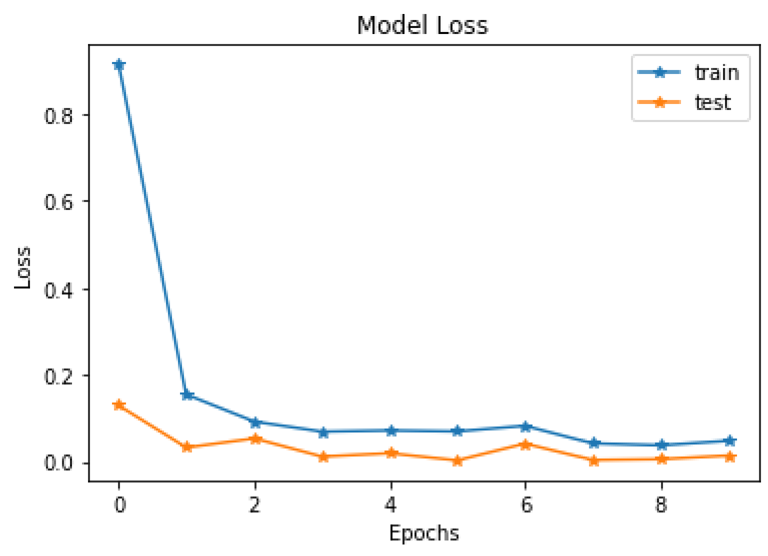
```
Epoch 1/10
400/400 [==============================] - 68s 170ms/step - loss: 0.9146 - accuracy: 0.6670 -

Epoch 00001: val_accuracy improved from -inf to 0.94969, saving model to weights.hdf5
Epoch 2/10
400/400 [==============================] - 61s 153ms/step - loss: 0.1544 - accuracy: 0.9509 -

Epoch 00002: val_accuracy improved from 0.94969 to 0.99219, saving model to weights.hdf5
Epoch 3/10
400/400 [==============================] - 60s 151ms/step - loss: 0.0923 - accuracy: 0.9734 -

Epoch 00003: val_accuracy did not improve from 0.99219
Epoch 4/10
400/400 [==============================] - 61s 151ms/step - loss: 0.0692 - accuracy: 0.9787 -

Epoch 00004: val_accuracy improved from 0.99219 to 0.99469, saving model to weights.hdf5
Epoch 5/10
400/400 [==============================] - 61s 152ms/step - loss: 0.0719 - accuracy: 0.9792 -

Epoch 00005: val_accuracy did not improve from 0.99469
Epoch 6/10
400/400 [==============================] - 61s 153ms/step - loss: 0.0697 - accuracy: 0.9823 -

Epoch 00006: val_accuracy improved from 0.99469 to 0.99937, saving model to weights.hdf5
Epoch 7/10
400/400 [==============================] - 61s 153ms/step - loss: 0.0826 - accuracy: 0.9769 -

Epoch 00007: val_accuracy did not improve from 0.99937
Epoch 8/10
400/400 [==============================] - 61s 152ms/step - loss: 0.0420 - accuracy: 0.9884 -

Epoch 00008: val_accuracy did not improve from 0.99937
Epoch 9/10
400/400 [==============================] - 61s 153ms/step - loss: 0.0376 - accuracy: 0.9899 -

Epoch 00009: val_accuracy did not improve from 0.99937
Epoch 10/10
400/400 [==============================] - 61s 152ms/step - loss: 0.0480 - accuracy: 0.9870 -

Epoch 00010: val_accuracy did not improve from 0.99937
```

Model Performance

```
plt.plot(mymodel.history['accuracy'],'*-')
plt.plot(mymodel.history['val_accuracy'],'*-')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'test'])
plt.show()
```



```
plt.plot(mymodel.history['loss'],'*-')
plt.plot(mymodel.history['val_loss'],'*-')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'test'])
plt.show()
```



Predictions on Test Set

```
pred = model.predict_classes(x_test)
pred
```

```
array([5, 8, 7, ..., 1, 6, 8])
```

Confusion Matrix & Accuracy Score

```
actual=[]
for i in range(len(y_test)):
    actual.append(np.argmax(y_test[i]))
print(confusion_matrix(actual,pred))
accuracy_score(actual,pred)
```

```
[[392   0   0   0   0   0   0   0   0   0]
 [  0 386   0   0   0   0   0   0   0   0]
 [  0   0 397   0   0   0   0   2   0   0]
 [  0  31   1 365   0   0   0   0   0   0]
 [  0   0   0   0 396   0   0   1   0   0]
 [  0   0   0   0   0 408   0   0   0   0]
 [  0   0   0   0   0   0 391   0   0   0]
 [  0   0   0   0   0   0   0 391   0   0]
 [  0   0   0   0   0   0   0   0 436   0]
 [  0   0   1   0   0   0   0   0   0 402]]
0.991
```