Sejal Sanas

Task - 4

In [1]:
```
import warnings
warnings.filterwarnings('ignore')

import keras
import matplotlib.pyplot as plt # for plotting
import os # provides a way of using operating system dependent functionality
import cv2 #Image handling library
import numpy as np
```

In [2]:
```
from keras.layers import Conv2D, Activation, MaxPool2D, Dense, Flatten, Dropout
```

In [3]:
```
CATEGORIES = ["01_palm", '02_l','03_fist','04_fist_moved','05_thumb','06_index','07_ok','08_palm_moved
IMG_SIZE = 50

# paths for dataset
data_path = "C:/Users/Shejal Sanas/Downloads/archive/leapGestRecog/leapGestRecog"
```

In [4]:
```
image_data = []
for dr in os.listdir(data_path):
    for category in CATEGORIES:
        class_index = CATEGORIES.index(category)
        path = os.path.join(data_path, dr, category)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                image_data.append([cv2.resize(img_arr, (IMG_SIZE, IMG_SIZE)), class_index])
            except Exception as e:
                pass
image_data[0]
```

Out[4]:
```
[array([[5, 4, 4, ..., 3, 4, 2],
        [5, 4, 5, ..., 3, 3, 3],
        [4, 5, 4, ..., 4, 5, 3],
        ...,
        [4, 5, 5, ..., 5, 5, 5],
        [5, 5, 6, ..., 5, 7, 4],
        [4, 7, 5, ..., 5, 4, 4]], dtype=uint8),
 0]
```

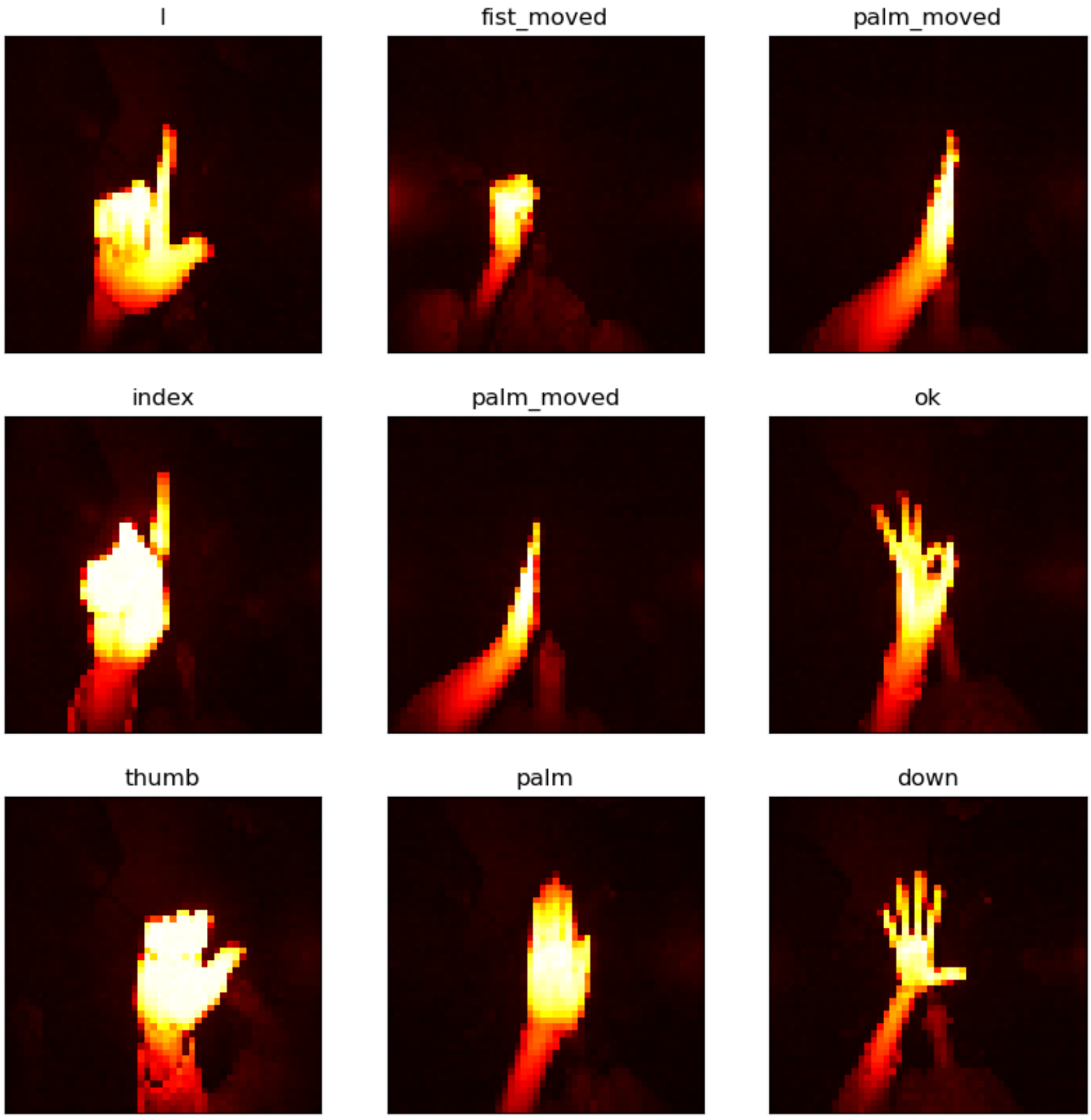
In [5]:
```
import random
random.shuffle(image_data)
```

In [6]:
```
input_data = []
label = []
for X, y in image_data:
    input_data.append(X)
    label.append(y)
```

In [7]:
```
label[:10]
```

Out[7]:
```
[9, 1, 3, 7, 5, 7, 6, 4, 0, 9]
```

In [8]:
```
plt.figure(1, figsize=(10,10))
for i in range(1,10):
    plt.subplot(3,3,i)
    plt.imshow(image_data[i][0], cmap='hot')
    plt.xticks([])
    plt.yticks([])
    plt.title(CATEGORIES[label[i]][3:])
plt.show()
```



In [9]:
```
# Normalizing the data
input_data = np.array(input_data)
label = np.array(label)
input_data = input_data/255.0
input_data.shape
```

Out[9]: (20000, 50, 50)

In [10]:
```
label = keras.utils.to_categorical(label, num_classes=10)
label[0]
```

Out[10]: array([0., 0., 0., 0., 0., 0., 0., 0., 0., 1.])

In [11]:
```
# reshaping the data
input_data.shape = (-1, IMG_SIZE, IMG_SIZE, 1)
```

In [12]:
```
# splitting the input_data to train and test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(input_data, label, test_size = 0.3, random_state=0
```

In [13]:
```
model = keras.models.Sequential()

model.add(Conv2D(filters = 32, kernel_size = (3,3), input_shape = (IMG_SIZE, IMG_SIZE, 1)))
model.add(Activation('relu'))


model.add(Conv2D(filters = 32, kernel_size = (3,3)))
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.3))

model.add(Conv2D(filters = 64, kernel_size = (3,3)))
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))

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

In [14]:
```
model.fit(X_train, y_train, epochs = 7, batch_size=32, validation_data=(X_test, y_test))
```

```
Epoch 1/7
438/438 ━━━━━━━━━━━━━━━━━━━━ 42s 89ms/step - accuracy: 0.6738 - loss: 0.9456 - val_accuracy: 0.9968 -
val_loss: 0.0132
Epoch 2/7
438/438 ━━━━━━━━━━━━━━━━━━━━ 38s 88ms/step - accuracy: 0.9934 - loss: 0.0251 - val_accuracy: 0.9987 -
val_loss: 0.0041
Epoch 3/7
438/438 ━━━━━━━━━━━━━━━━━━━━ 39s 88ms/step - accuracy: 0.9964 - loss: 0.0116 - val_accuracy: 0.9992 -
val_loss: 0.0024
Epoch 4/7
438/438 ━━━━━━━━━━━━━━━━━━━━ 40s 91ms/step - accuracy: 0.9980 - loss: 0.0065 - val_accuracy: 0.9992 -
val_loss: 0.0032
Epoch 5/7
438/438 ━━━━━━━━━━━━━━━━━━━━ 42s 97ms/step - accuracy: 0.9990 - loss: 0.0059 - val_accuracy: 0.9987 -
val_loss: 0.0070
Epoch 6/7
438/438 ━━━━━━━━━━━━━━━━━━━━ 43s 99ms/step - accuracy: 0.9985 - loss: 0.0035 - val_accuracy: 0.9997 -
val_loss: 0.0032
Epoch 7/7
438/438 ━━━━━━━━━━━━━━━━━━━━ 40s 92ms/step - accuracy: 0.9990 - loss: 0.0027 - val_accuracy: 0.9993 -
val_loss: 0.0016
```

Out[14]: <keras.src.callbacks.history.History at 0x1fd63da5b50>

In [15]:
```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 32)	320
activation (Activation)	(None, 48, 48, 32)	0
conv2d_1 (Conv2D)	(None, 46, 46, 32)	9,248
activation_1 (Activation)	(None, 46, 46, 32)	0
max_pooling2d (MaxPooling2D)	(None, 23, 23, 32)	0
dropout (Dropout)	(None, 23, 23, 32)	0
conv2d_2 (Conv2D)	(None, 21, 21, 64)	18,496
activation_2 (Activation)	(None, 21, 21, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 10, 10, 64)	0
dropout_1 (Dropout)	(None, 10, 10, 64)	0
flatten (Flatten)	(None, 6400)	0
dense (Dense)	(None, 256)	1,638,656
dense_1 (Dense)	(None, 10)	2,570

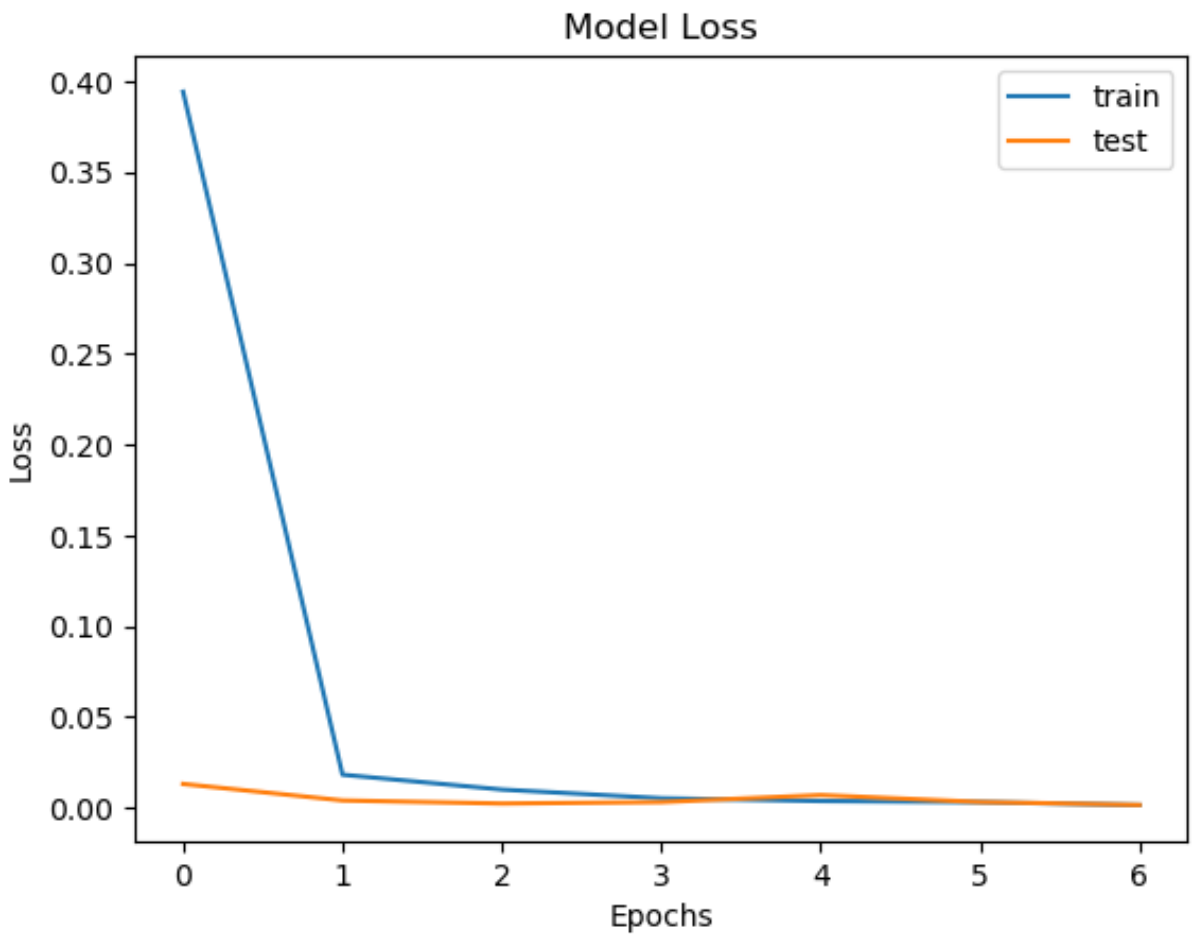
Total params: 3,338,582 (12.74 MB)

Trainable params: 1,669,290 (6.37 MB)

Non-trainable params: 0 (0.00 B)

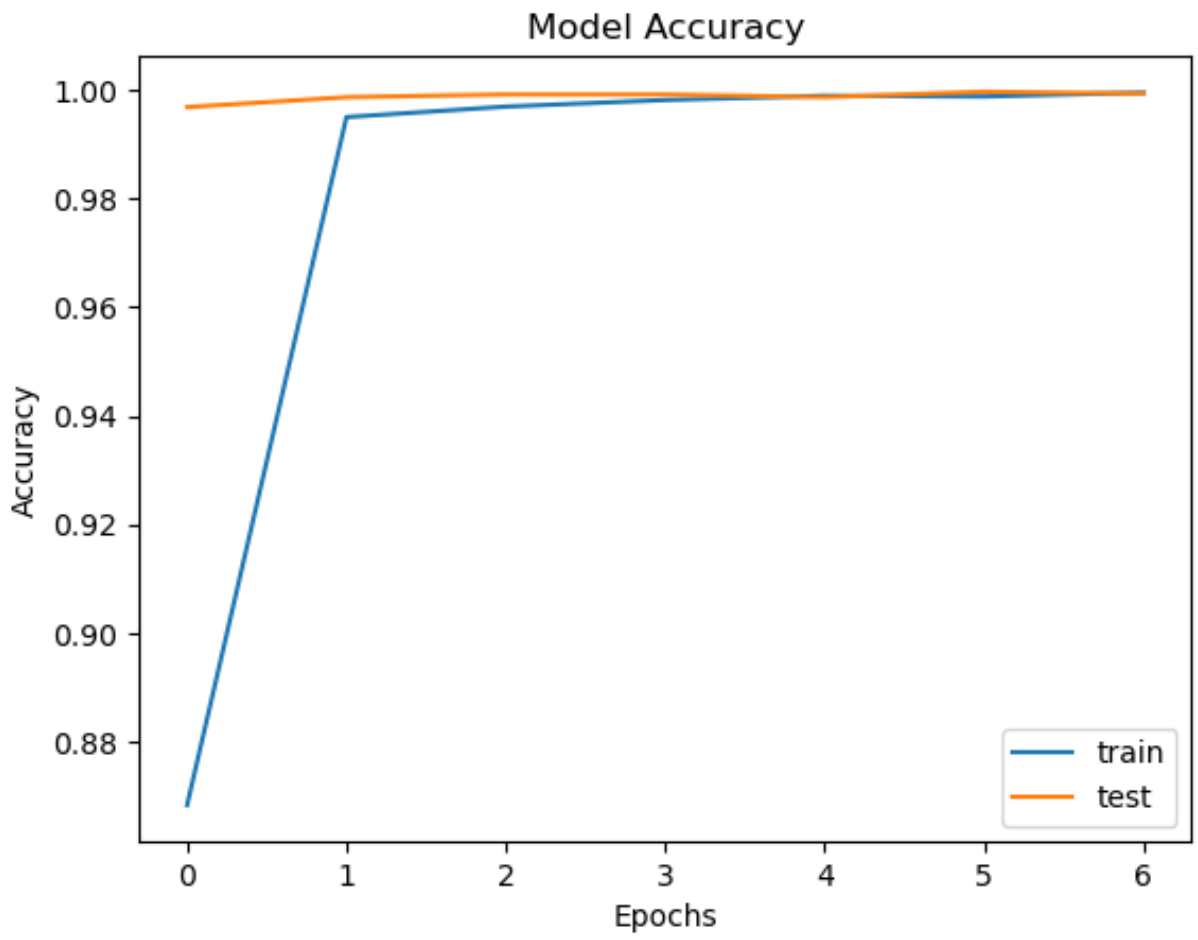
Optimizer params: 1,669,292 (6.37 MB)

In [16]:
```
plt.plot(model.history.history['loss'])
plt.plot(model.history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'test'])
plt.show()
```



Loading [MathJax]/extensions/Safe.js

In [17]:
```
plt.plot(model.history.history['accuracy'])
plt.plot(model.history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'test'])
plt.show()
```



In [18]:
```
test_loss, test_accuracy = model.evaluate(X_test, y_test)

print('Test accuracy: {:2.2f}%'.format(test_accuracy*100))
```

```
188/188 ━━━━━━━━━━━━━━━━━━━━ 4s 21ms/step - accuracy: 0.9991 - loss: 0.0038
Test accuracy: 99.93%
```

In [19]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sn
cat = [c[3:] for c in CATEGORIES]
plt.figure(figsize=(10,10))
cm = confusion_matrix(np.argmax(y_test, axis=1), np.argmax(model.predict(X_test), axis=1))
sn.heatmap(cm, annot=True,xticklabels=cat, yticklabels=cat)
plt.plot()
```

```
188/188 ━━━━━━━━━━━━━━━━━━━━ 4s 22ms/step
```

Out[19]: []