Testing the model

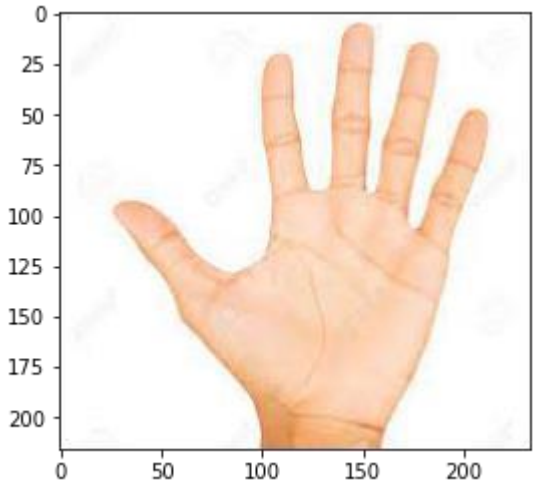
Importing Libraries

```
from tensorflow.keras.models import load_model from tensorflow.keras.preprocessing import
image model = load_model("gesture.h5") #loading the model for testing
path = "C:\\Users\\Anura\\OneDrive\\Desktop\\Gesture-Based-Number-Recognition-main\\im6.j
```

Plotting the image

```
%pylab inline
import matplotlib.pyplot as plt import
matplotlib.image as mpimg imgs =
mpimg.imread(path) imgplot = plt.imshow(imgs)
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```




```
#loading of the image img =
image.load_img(path, color_mode='grayscale',
target_size= (64,64))
x = image.img_to_array(img)#image to array
x.shape
```

```
(64, 64, 1)
```

```
type(x)
```

```
numpy.ndarray
```

```
#changing the shape x = np.expand_dims(x,axis = 0)
```

```
x.shape
```

```
(1, 64, 64, 1)
```

Predicting our results

```
pred = model.predict_classes(x)#predicting the classes pred
```

```
C:\Users\Anura\anaconda3\lib\site-packages\tensorflow\python\keras\engine\sequential.py:4
55: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021-0 1-
01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`, if your model does m ulti-
class classification (e.g. if it uses a `softmax` last-layer activation).* `(mode l.predict(x)
> 0.5).astype("int32")`, if your model does binary classification (e.g.
if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
array([5], dtype=int64)
```

```
index=['0','1','2','3','4','5']
result=str(index[pred[0]])
result
```

```
'5'
```

```
import numpy as np
p = []
for i in range(0,6):
    for j in range(0,5):
        path = "C:\\Users\\Anura\\OneDrive\\Desktop\\Gesture-Based-Number-Recognition-mai
        img = image.load_img(path,color_mode = "grayscale",target_size= (64,64)) x =
        image.img_to_array(img) x = np.expand_dims(x,axis = 0)
        pred = np.argmax(model.predict(x), axis=-1) p.append(pred)
print(p)
```

```
[array([0], dtype=int64), array([0], dtype=int64), array([0], dtype=int64), array([0], dt
ype=int64), array([0], dtype=int64), array([1], dtype=int64), array([1], dtype=int64), ar
ray([1], dtype=int64), array([1], dtype=int64), array([1], dtype=int64), array([2], dtype
=int64), array([2], dtype=int64), array([1], dtype=int64), array([2], dtype=int64), array
([2], dtype=int64), array([3], dtype=int64), array([3], dtype=int64), array([3], dtype=in
t64), array([3], dtype=int64), array([3], dtype=int64), array([4], dtype=int64), array ([4],
dtype=int64), array([4], dtype=int64), array([4], dtype=int64), array([4], dtype=in t64),
array([5], dtype=int64), array([5], dtype=int64), array([5], dtype=int64), array
([5], dtype=int64), array([5], dtype=int64)]
```

```
result = []
index=['0','1','2','3','4','5'] for
i in p:
    result.append(index[i[0]])
print(result)
```

```
['0', '0', '0', '0', '0', '1', '1', '1', '1', '1', '2', '2', '1', '2', '2', '3', '3', '3
', '3', '3', '4', '4', '4', '4', '4', '5', '5', '5', '5', '5']
```

...