5.Result and Discussion:

The outcomes of our experimental examination are presented in this section. The purpose of these tests is to gain insight into the performance of detection approaches , as well as to assess the performance of identification models that. As a result of our project, the accuracy level is 99.90%.

The accuracy when the model works on an image from test section.

```
[loss, acc] = model.evaluate(x_test,y_test,verbose=1)
print("Accuracy:" + str(acc))
```

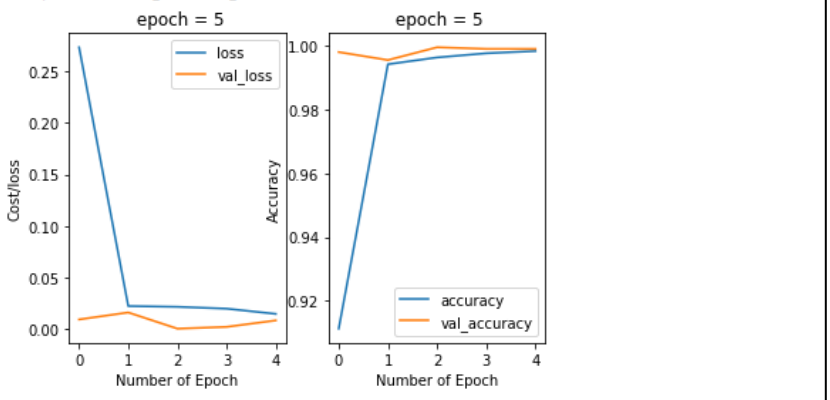
```
63/63 [==============================] - 6s 90ms/step - loss: 0.0000e+00 - accuracy: 1.0000
Accuracy:1.0
```

Accuracy vs Loss graph for different scenarios.

```
[21] model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
     history = model.fit(x_train, y_train, epochs=5, batch_size=16, verbose=1, validation_data=(x_validate, y_validate))

     Epoch 1/5
     1000/1000 [==============================] - 165s 164ms/step - loss: 0.2731 - accuracy: 0.9113 - val_loss: 0.0096 - val_accuracy: 0.9980
     Epoch 2/5
     1000/1000 [==============================] - 234s 234ms/step - loss: 0.0225 - accuracy: 0.9942 - val_loss: 0.0163 - val_accuracy: 0.9955
     Epoch 3/5
     1000/1000 [==============================] - 168s 168ms/step - loss: 0.0218 - accuracy: 0.9963 - val_loss: 6.3329e-04 - val_accuracy: 0.9995
     Epoch 4/5
     1000/1000 [==============================] - 167s 167ms/step - loss: 0.0200 - accuracy: 0.9976 - val_loss: 0.0023 - val_accuracy: 0.9990
     Epoch 5/5
     1000/1000 [==============================] - 174s 174ms/step - loss: 0.0150 - accuracy: 0.9983 - val_loss: 0.0086 - val_accuracy: 0.9990
```

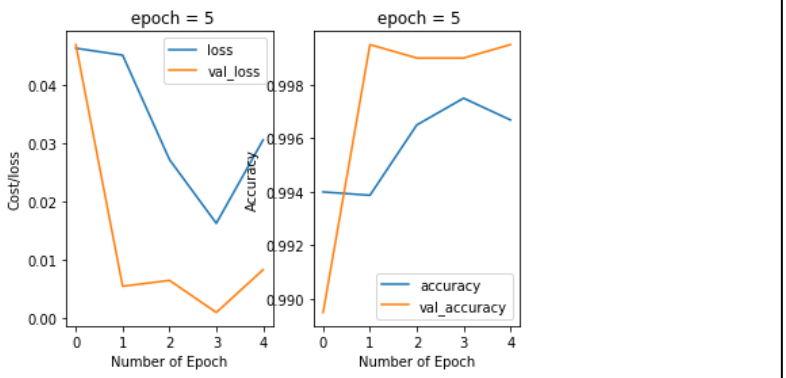
```
<matplotlib.legend.Legend at 0x7f73fe5a8790>
```



```
[25] model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.002), loss='categorical_crossentropy', metrics=['accuracy'])
     history = model.fit(x_train, y_train, epochs=5, batch_size=32, verbose=1, validation_data=(x_validate, y_validate))

     Epoch 1/5
     500/500 [==============================] - 170s 339ms/step - loss: 0.0464 - accuracy: 0.9940 - val_loss: 0.0470 - val_accuracy: 0.9895
     Epoch 2/5
     500/500 [==============================] - 161s 321ms/step - loss: 0.0452 - accuracy: 0.9939 - val_loss: 0.0055 - val_accuracy: 0.9995
     Epoch 3/5
     500/500 [==============================] - 161s 321ms/step - loss: 0.0273 - accuracy: 0.9965 - val_loss: 0.0065 - val_accuracy: 0.9990
     Epoch 4/5
     500/500 [==============================] - 157s 314ms/step - loss: 0.0163 - accuracy: 0.9975 - val_loss: 0.0010 - val_accuracy: 0.9990
     Epoch 5/5
     500/500 [==============================] - 288s 577ms/step - loss: 0.0307 - accuracy: 0.9967 - val_loss: 0.0084 - val_accuracy: 0.9995
```

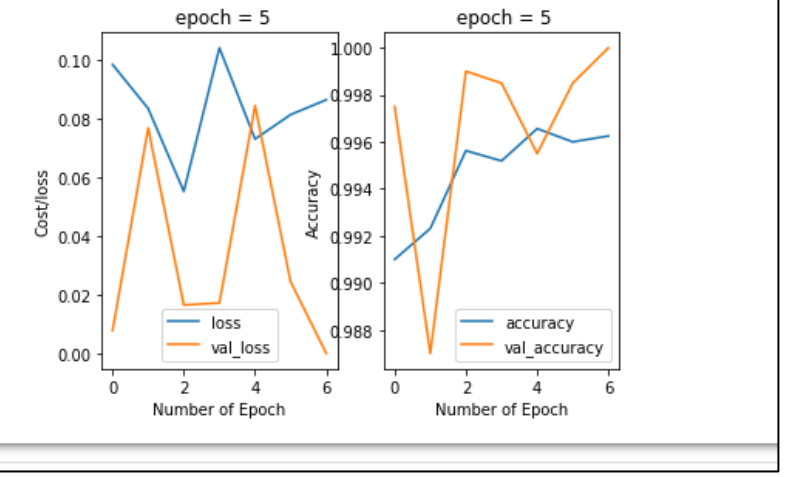
```
<matplotlib.legend.Legend at 0x7f74023ebdd0>
```




```
[30] model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.003), loss='categorical_crossentropy', metrics=['accuracy'])
     history = model.fit(x_train, y_train, epochs=7, batch_size=16, verbose=1, validation_data=(x_validate, y_validate))

     Epoch 1/7
     1000/1000 [==============================] - 169s 168ms/step - loss: 0.0985 - accuracy: 0.9910 - val_loss: 0.0078 - val_accuracy: 0.9975
     Epoch 2/7
     1000/1000 [==============================] - 194s 194ms/step - loss: 0.0835 - accuracy: 0.9923 - val_loss: 0.0768 - val_accuracy: 0.9870
     Epoch 3/7
     1000/1000 [==============================] - 173s 173ms/step - loss: 0.0552 - accuracy: 0.9956 - val_loss: 0.0165 - val_accuracy: 0.9990
     Epoch 4/7
     1000/1000 [==============================] - 169s 169ms/step - loss: 0.1042 - accuracy: 0.9952 - val_loss: 0.0172 - val_accuracy: 0.9985
     Epoch 5/7
     1000/1000 [==============================] - 166s 166ms/step - loss: 0.0731 - accuracy: 0.9966 - val_loss: 0.0845 - val_accuracy: 0.9955
     Epoch 6/7
     1000/1000 [==============================] - 175s 175ms/step - loss: 0.0814 - accuracy: 0.9960 - val_loss: 0.0244 - val_accuracy: 0.9985
     Epoch 7/7
     1000/1000 [==============================] - 331s 331ms/step - loss: 0.0865 - accuracy: 0.9962 - val_loss: 1.1921e-10 - val_accuracy: 1.0000
```

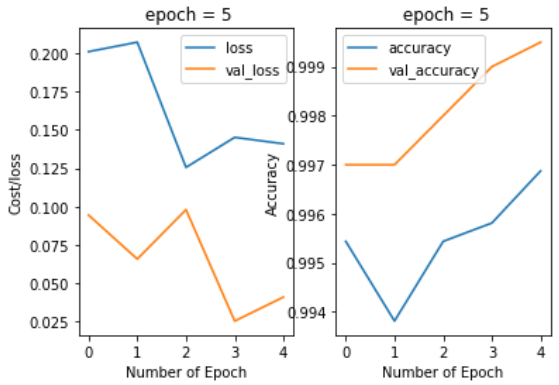
```
<matplotlib.legend.Legend at 0x7f73fd94d950>
```



```
[32] model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.004), loss='categorical_crossentropy', metrics=['accuracy'])
     history = model.fit(x_train, y_train, epochs=5, batch_size=32, verbose=1, validation_data=(x_validate, y_validate))

     Epoch 1/5
     500/500 [==============================] - 166s 329ms/step - loss: 0.2010 - accuracy: 0.9954 - val_loss: 0.0945 - val_accuracy: 0.9970
     Epoch 2/5
     500/500 [==============================] - 158s 315ms/step - loss: 0.2072 - accuracy: 0.9938 - val_loss: 0.0659 - val_accuracy: 0.9970
     Epoch 3/5
     500/500 [==============================] - 172s 343ms/step - loss: 0.1256 - accuracy: 0.9954 - val_loss: 0.0980 - val_accuracy: 0.9980
     Epoch 4/5
     500/500 [==============================] - 160s 319ms/step - loss: 0.1452 - accuracy: 0.9958 - val_loss: 0.0255 - val_accuracy: 0.9990
     Epoch 5/5
     500/500 [==============================] - 162s 325ms/step - loss: 0.1410 - accuracy: 0.9969 - val_loss: 0.0411 - val_accuracy: 0.9995
```

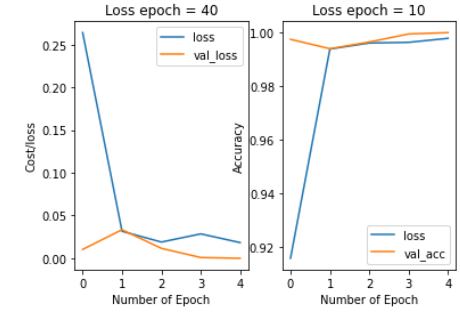
```
<matplotlib.legend.Legend at 0x7f7403ba0490>
```



Size= 200,200 epoch = 15, batch size= 16, learning rate= 0.001

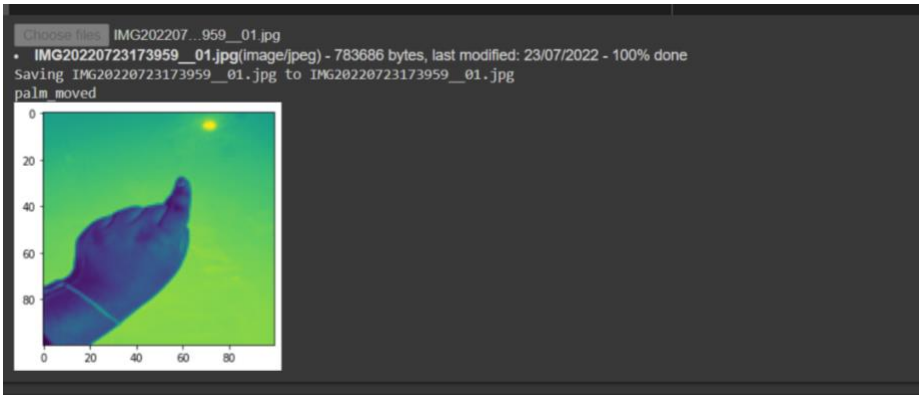
```
plt.xlabel('Number of Epoch')
plt.ylabel('Cost/loss')
plt.legend(['loss', 'val_loss'])
plt.subplot(122)
plt.plot(histry.history['accuracy'])
plt.plot(histry.history['val_accuracy'])
plt.title("Loss epoch = 10")
plt.xlabel('Number of Epoch')
plt.ylabel('Accuracy')
plt.legend(['loss', 'val_acc'])
```

```
<matplotlib.legend.Legend at 0x7f654c353510>
```



Predictions made by detector when images were provided externally-

1. **Batch size=16, epochs=2, image size = (100,100)**

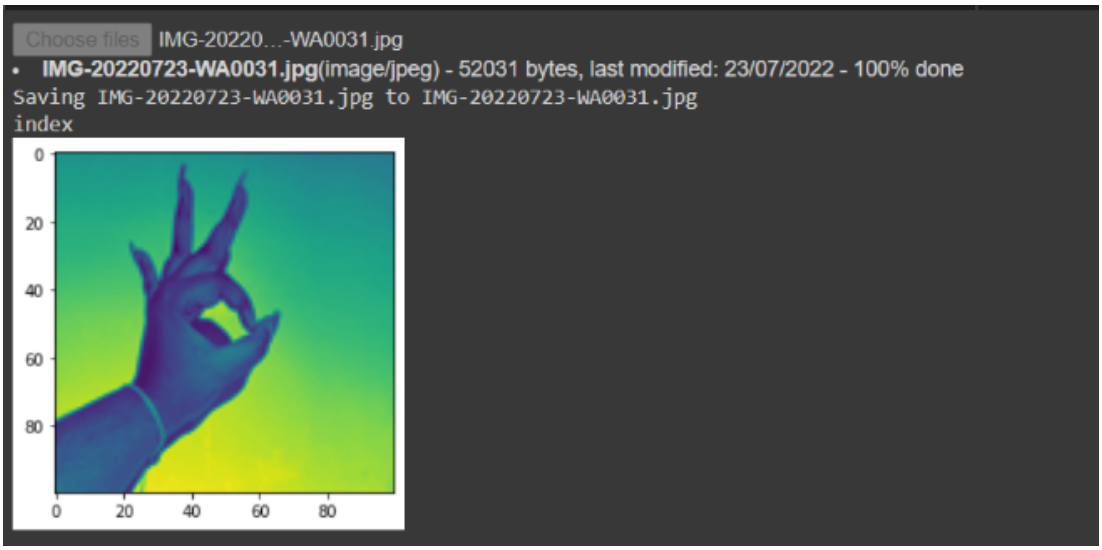


Given prediction- palm_moved

More accurate prediction- index

Given prediction- fist_moved

More accurate prediction- thumb



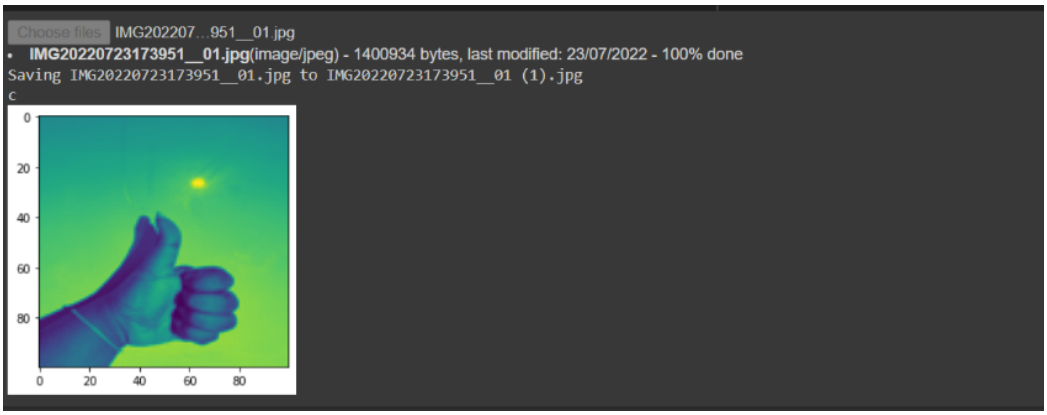
Given prediction- index

More accurate prediction – ok

Given prediction-fist

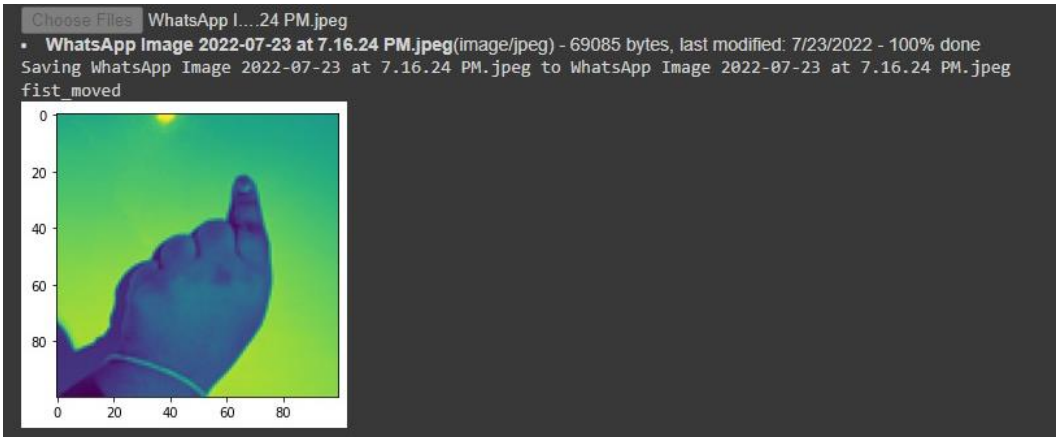
More accurate prediction- c

2. Batch size=16, epochs=5, image size = (100,100)



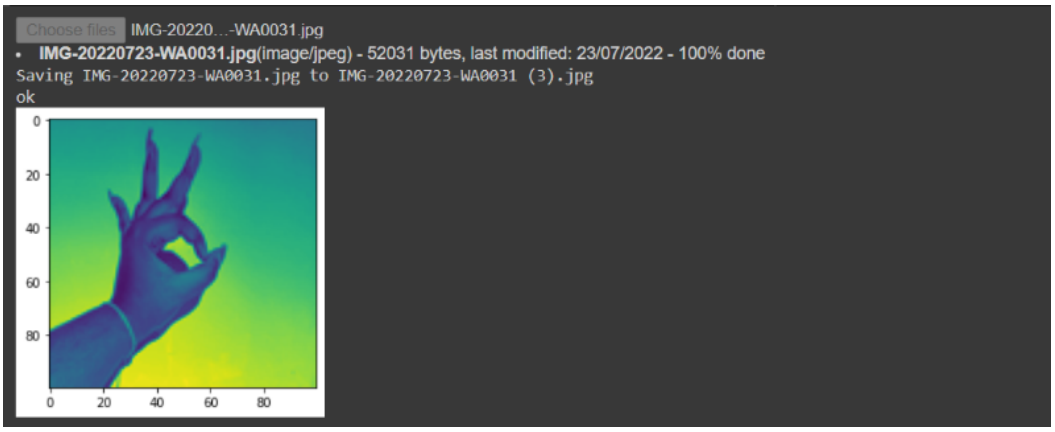
Given prediction- c

More accurate prediction- thumb



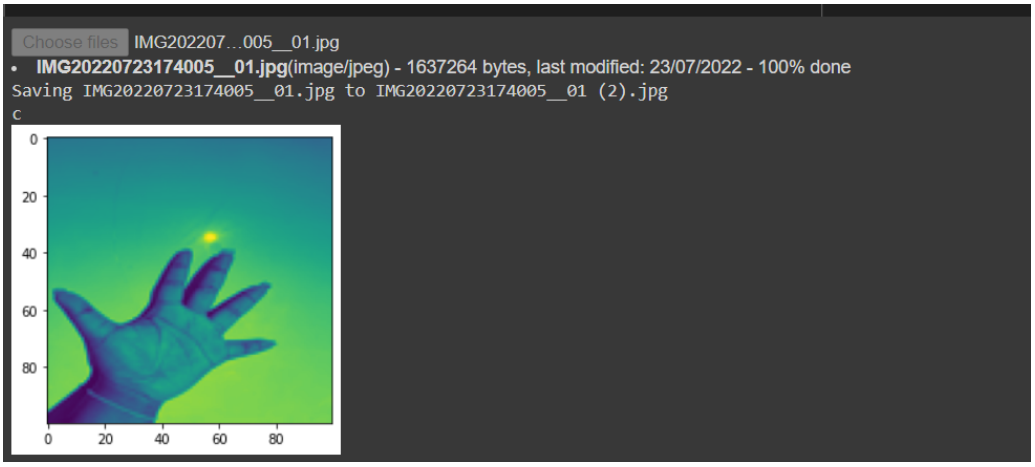
Given prediction- fist_moved

More accurate prediction – index



Given prediction – ok

Accurate prediction – ok



Given prediction- c

Accurate prediction- palm/palm_moved

DISCUSSIONS:

From the above results, it is evident that despite of having very high accuracy during training, the model failed to detect external images accurately.

Probable explanations for this result-

1. Our model is overfit. Thus biased to the dataset.
2. The model was trained for very few number of epochs.
   (Note: for 5 epochs the model worked better than for 2 epochs)

Reasons for which the model might have become overfit –

1. Too many layers are there in the model.
2. There were too many images in training set and the data has been trained "too well" hence performs well only with known data.

Ways we implemented that prevent overfitting –

1. We have followed the 80% - 20% ratio of training set and test set. This is a known method for avoiding overfitting.
2. We have used dropouts in our model. This again is another suggested method to avoid overfitting.

There are other ways of preventing overfitting such as regularization. The model can be made simpler. Other methods like data augmentation are also implemented. We will try to improve our model in future so that we can reduce overfitting and detect unseen images better.