In [31]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
import os
```

In [32]:
```
import cv2
import matplotlib.pyplot as plt
```

In [33]:
```
import random
```

In [34]:
```
from tqdm import tqdm
```

In [35]:
```
path = "C:/Users/mukun/Downloads/Image_PRO2/ImagePro"
```

In [36]:
```
files = os.listdir(path)
```

In [37]:
```
files.sort()
```

In [38]:
```
print(files)
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O',
'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'ZZ']
```

In [39]:
```
#image and its lable
image_array = []
label_array = []
```

In [40]:
```
for i in tqdm(range(len(files))):
    sub_files = os.listdir(path+"/"+files[i])
    #print(len(sub_files))
    for j in range(len(sub_files)):
        file_path = path+"/"+files[i]+"/"+sub_files[j]
        #CV2 read image
        image = cv2.imread(file_path)
        #resize 96*96
        image = cv2.resize(image,(96,96))
        if random.randint(1,10) > 6:
            image = cv2.flip(image,1)
        #color blackwhite
        #image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
        #color black to RGB
        #image=cv2.cvtColor(image, cv2.COLOR_GRAY2RGB)
        image=cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image_array.append(image)

        label_array.append(i)
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 27/27 [01:39<00:00,  3.68s/it]
```

In [41]:
```
image_array = np.array(image_array)
label_array = np.array(label_array,dtype="float")
```

In [42]:
```
from sklearn.model_selection import train_test_split
```

In [43]:
```
X_train,X_test,Y_train,Y_test= train_test_split(image_array,label_array,test_size
```

In [44]:
```
del image_array,label_array
import gc
gc.collect()
```

Out[44]: 22

In [21]:
```
from keras import layers,callbacks,utils,applications,optimizers
from keras.models import Sequential, Model, load_model
```

In [22]:
```
model = Sequential()
pretrained_model = tf.keras.applications.EfficientNetB4(input_shape=(96,96,3),inc
model.add(pretrained_model)
```

In [23]:
```
model.add(layers.GlobalAveragePooling2D())
```

In [24]:
```
model.add(layers.Dropout(0.3))
model.add(layers.Dense(1))
model.build(input_shape=(None,96,96,2))
```

In [25]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb4 (Functional)  (None, 3, 3, 1792)       17673823

 global_average_pooling2d (G  (None, 1792)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 1792)              0

 dense (Dense)               (None, 1)                 1793

=================================================================
Total params: 17,675,616
Trainable params: 17,550,409
Non-trainable params: 125,207
_________________________________________________________________
```

In [26]:
```
model.compile(loss='mse', optimizer='adam', metrics=['mae'])
```

In [27]:
```
ckg_path = "trained_model/model"
model_checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath= ckg_path, monitor
```

In [28]:
```
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(
factor = 0.9, monitor = "val_mae", mode = "auto", cooldown = 0, patience = 5, ver
)
```

In [23]:
```
Epoch = 70
Batch_size = 32
```

In [24]:
```
_train,validation_data=(X_test,Y_test),batch_size = Batch_size, epochs= Epoch, cal
```

```
Epoch 1/70
306/306 [==============================] - ETA: 0s - loss: 17.9826 - mae: 2.6
672INFO:tensorflow:Assets written to: trained_model\model\assets
306/306 [==============================] - 180s 510ms/step - loss: 17.9826 -
mae: 2.6672 - val_loss: 2.0945 - val_mae: 0.9258 - lr: 0.0010
Epoch 2/70
306/306 [==============================] - ETA: 0s - loss: 1.9341 - mae: 0.97
73INFO:tensorflow:Assets written to: trained_model\model\assets
306/306 [==============================] - 152s 497ms/step - loss: 1.9341 - m
ae: 0.9773 - val_loss: 1.1994 - val_mae: 0.6305 - lr: 0.0010
Epoch 3/70
306/306 [==============================] - ETA: 0s - loss: 1.5190 - mae: 0.84
37INFO:tensorflow:Assets written to: trained_model\model\assets
306/306 [==============================] - 149s 486ms/step - loss: 1.5190 - m
ae: 0.8437 - val_loss: 0.8763 - val_mae: 0.4720 - lr: 0.0010
Epoch 4/70
306/306 [==============================] - 76s 248ms/step - loss: 1.9800 - ma
e: 0.8962 - val_loss: 1.6122 - val_mae: 0.8552 - lr: 0.0010
Epoch 5/70
```

In [ ]:
```
results = model.evaluate(X_test,Y_test, batch_size=32)
```

In [29]:
```
model.load_weights(ckg_path)
```

Out[29]:
```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x19d0b863ac0
>
```

In [27]:
```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model=converter.convert()
```

```
INFO:tensorflow:Assets written to: C:\Users\mukun\AppData\Local\Temp\tmpnz_fqg0
z\assets

WARNING:absl:Buffer deduplication procedure will be skipped when flatbuffer lib
rary is not properly loaded
```

In [28]:
```
with open("model2withflips10echB4.tflite","wb") as f:
    f.write(tflite_model)
```

In [47]:
```
import math
```

In [53]:
```
prediction_val = model.predict(X_test,batch_size=32)
print(prediction_val)
print(Y_test)
```

```
[[20.777805 ]
 [ 3.9893966]
 [19.925682 ]
 ...
 [25.997211 ]
 [ 1.0067749]
 [10.974005 ]]
[21.  4. 20. ... 26.  1. 11.]
```

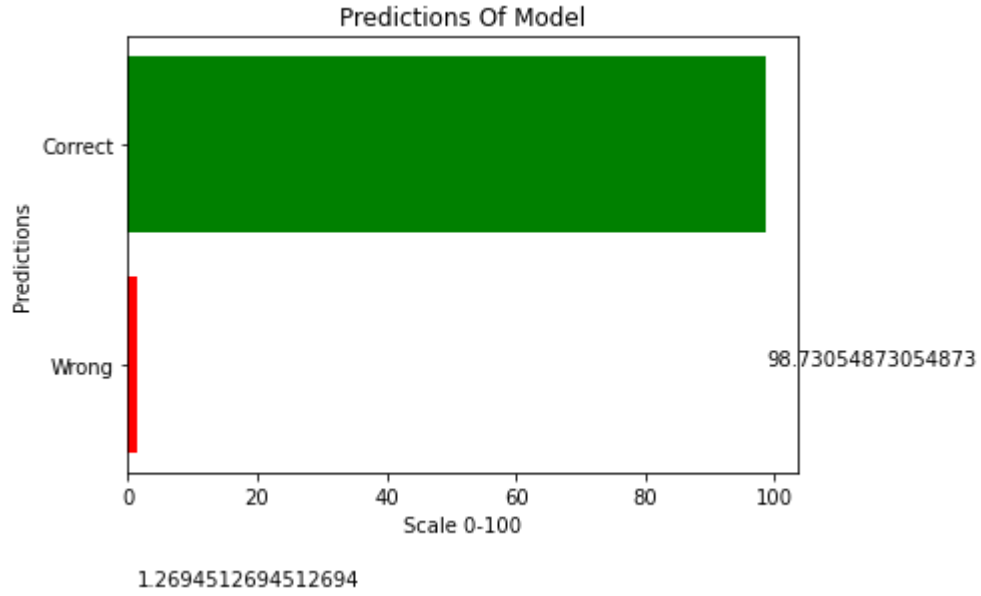
In [54]:
```
pre = prediction_val.tolist()
y_tes = Y_test.tolist()
```

In [62]:
```
correct = 0
wrong = 0
for i in range(len(pre)):
    if round(pre[i][0]) == y_tes[i]:
        correct += 1
    else:
        wrong += 1
a = correct/len(pre)*100
b = wrong/len(pre)*100
print(a,b)
```

```
98.73054873054873 1.2694512694512694
```

In [66]:
```
import matplotlib.pyplot as plt
```

In [82]:
```
left = [1, 2]
height = [b,a]
tick_label = ['Wrong','Correct']
for index, value in enumerate(height):
    plt.text(value, index,
             str(value))
plt.barh(left, height, tick_label = tick_label, color = ['red','green'])
plt.ylabel('Predictions')
plt.xlabel('Scale 0-100')
plt.title('Predictions Of Model')
plt.figure(figsize=(10, 10))
plt.show()
```



1.2694512694512694

<Figure size 720x720 with 0 Axes>