

1. Importing Modules

```
In [1]: # Importing tensorflow and File Management
import tensorflow as tf
import os

# Data Management and Display
import numpy as np
from matplotlib import pyplot as plt

# Image Processing
import cv2
import imghdr

# Model Training Modules
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout

# Metrics from Keras
from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy

# For Loading Model
from tensorflow.keras.models import load_model
```

2. Data Cleaning

```
data_dir = 'data'
```

```
image_exts = ['.jpeg', '.jpg', '.bmp', '.png']
```

```
for image_class in os.listdir(data_dir):
    for image in os.listdir(os.path.join(data_dir, image_class)):
        image_path = os.path.join(data_dir, image_class, image)
        try:
            img = cv2.imread(image_path)
            tip = imghdr.what(image_path)
            if tip not in image_exts:
                print('Unsupported Extension: {}'.format(image_path))
                os.remove(image_path)
        except Exception as e:
            print('Error with Image: {}'.format(image_path))
            # os.remove(image_path)
```

3. Loading Data

```
In [2]: #Import numpy Library
import numpy as np
from matplotlib import pyplot as plt
```

```
In [3]: #Loading images from dataset
data = tf.keras.utils.image_dataset_from_directory('F:\\DataSets\\Train')
```

Found 2326 files belonging to 2 classes.

```
In [4]: #Convert dataset to numpy iterator
data_iterator = data.as_numpy_iterator()
```

```
In [5]: #get a batch of data
batch = data_iterator.next()
```

batch

```
[[255.      , 255.      , 252.78125 ],
 [194.53706 , 194.664   , 192.78706 ]],

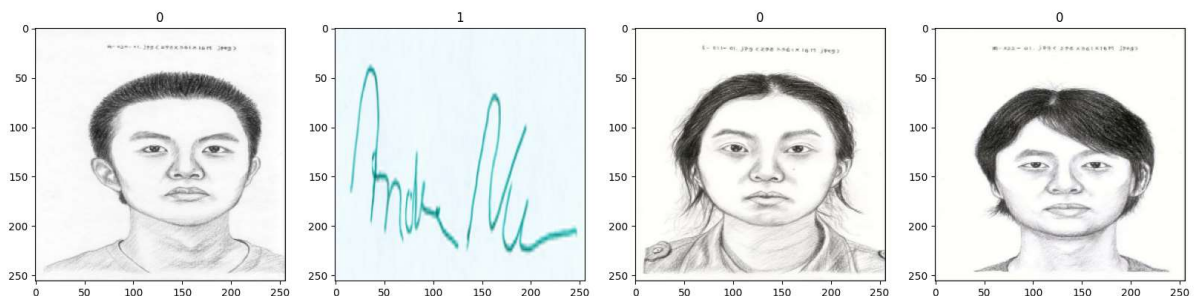
[[196.05821 , 197.05821 , 192.05821 ],
 [250.06526 , 251.06526 , 246.06526 ],
 [254.40825 , 254.95122 , 250.40825 ],
 ...,
 [253.46475 , 253.46475 , 253.46475 ],
 [253.73714 , 253.73714 , 253.73714 ],
 [195.8043  , 195.8043  , 195.8043  ]],

[[180.5922  , 181.5922  , 176.5922  ],
 [227.45589 , 228.45589 , 223.45589 ],
 [231.91994 , 232.62894 , 227.91994 ],
 ...,
 [231.20312 , 231.20312 , 231.20312 ],
 [230.36137 , 230.36137 , 230.36137 ],
 [182.8211  , 182.8211  , 182.8211  ]],
```

```
In [6]: #displaying images from the block
fig, ax = plt.subplots(ncols=4, figsize=(20,20))
for idx, img in enumerate(batch[0][:4]):
    ax[idx].imshow(img.astype(int))
    ax[idx].title.set_text(batch[1][idx])

#maximum pixel value
batch[0].max()
```

Out[6]: 255.0



4. Data Scaling

```
In [7]: data = data.map(lambda x,y: (x/255, y))
```

WARNING:tensorflow:From C:\Users\chana\AppData\Local\Programs\Python\Python39\lib\site-packages\tensorflow\python\autograph\pyct\static_analysis\liveness.py:83: Analyzer.lamba_check (from tensorflow.python.autograph.pyct.static_analysis.liveness) is deprecated and will be removed after 2023-09-23.
Instructions for updating:
Lambda fuctions will be no more assumed to be used in the statement where the y are used, or at least in the same block. <https://github.com/tensorflow/tensorflow/issues/56089> (<https://github.com/tensorflow/tensorflow/issues/56089>)

```
In [8]: data.as_numpy_iterator().next()
```

```
Out[8]: (array([[ [0.60750324, 0.6584836 , 0.7212287 ],
                  [0.6139696 , 0.66102844, 0.72377354],
                  [0.6086675 , 0.6557263 , 0.7184714 ],
                  ...,
                  [0.512927 , 0.52861327, 0.5752097 ],
                  [0.4836981 , 0.49548292, 0.5594334 ],
                  [0.4648562 , 0.4948807 , 0.56418216]],
                [ [0.6128887 , 0.6599475 , 0.7226926 ],
                  [0.6117044 , 0.65876323, 0.7215083 ],
                  [0.6158825 , 0.6585957 , 0.72278935],
                  ...,
                  [0.47263998, 0.48875037, 0.54612535],
                  [0.506429 , 0.5297746 , 0.592581 ],
                  [0.46948242, 0.49946672, 0.5687883 ]],
                [ [0.61583656, 0.66289544, 0.72564054],
                  [0.60923046, 0.65194356, 0.71613723],
                  [0.6183948 , 0.6536889 , 0.7203556 ],
```

5. Data Splitting

```
In [9]: len(data)
```

```
Out[9]: 73
```

```
In [10]: train_size = int(len(data)*.7)
          val_size = int(len(data)*.2) + 1
          test_size = int(len(data)*.1)
```

```
In [11]: # Checking if len(data) = sum(test, train, val sizes)
          train_size + val_size + test_size
```

```
Out[11]: 73
```

```
In [12]: train = data.take(train_size)
val = data.skip(train_size).take(val_size)
test = data.skip(train_size+val_size).take(test_size)
```

Building DL Model

```
In [13]: #Creating Empty Sequential Model
```

```
model = Sequential()
```

```
In [14]: #defining a CNN in Keras using the Sequential model and several layers
```

```
model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))
model.add(MaxPooling2D())
```

```
model.add(Conv2D(32, (3,3), 1, activation='relu'))
model.add(MaxPooling2D())
```

```
model.add(Conv2D(16, (3,3), 1, activation='relu'))
model.add(MaxPooling2D())
```

```
model.add(Flatten())
```

```
model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
In [15]: #Compiling the defined Keras mode
```

```
model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

```
In [16]: model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 254, 254, 16)	448
max_pooling2d (MaxPooling2D)	(None, 127, 127, 16)	0
conv2d_1 (Conv2D)	(None, 125, 125, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 32)	0
conv2d_2 (Conv2D)	(None, 60, 60, 16)	4624
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 16)	0
flatten (Flatten)	(None, 14400)	0
dense (Dense)	(None, 256)	3686656
dense_1 (Dense)	(None, 1)	257
=====		
Total params: 3,696,625		
Trainable params: 3,696,625		
Non-trainable params: 0		

7. Training the Model

```
In [17]: logdir='logs'
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
```

```
In [18]: hist = model.fit(train, epochs=25, validation_data=val, callbacks=[tensorboard.]
```

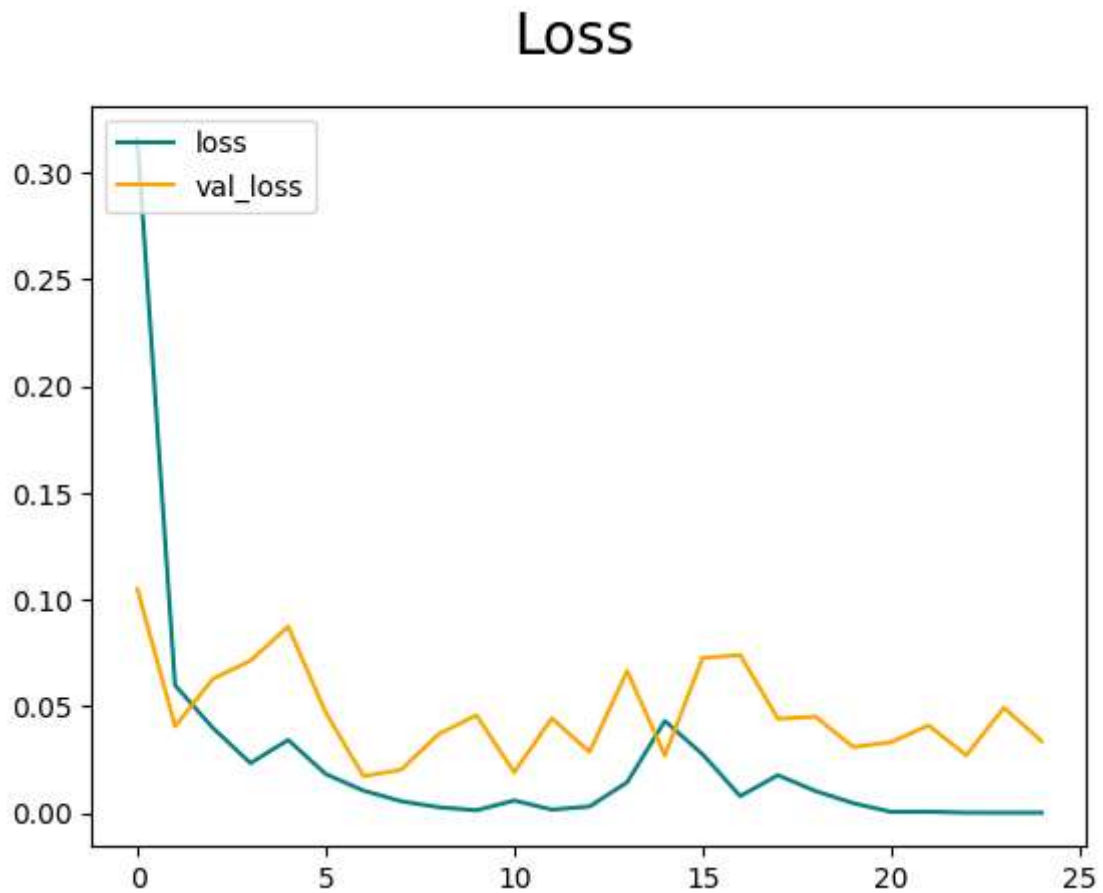
```
Epoch 1/25
51/51 [=====] - 69s 1s/step - loss: 0.3156 - accuracy: 0.8517 - val_loss: 0.1048 - val_accuracy: 0.9833
Epoch 2/25
51/51 [=====] - 57s 1s/step - loss: 0.0598 - accuracy: 0.9841 - val_loss: 0.0405 - val_accuracy: 0.9937
Epoch 3/25
51/51 [=====] - 58s 1s/step - loss: 0.0399 - accuracy: 0.9884 - val_loss: 0.0628 - val_accuracy: 0.9875
Epoch 4/25
51/51 [=====] - 64s 1s/step - loss: 0.0233 - accuracy: 0.9933 - val_loss: 0.0713 - val_accuracy: 0.9917
Epoch 5/25
51/51 [=====] - 74s 1s/step - loss: 0.0342 - accuracy: 0.9926 - val_loss: 0.0873 - val_accuracy: 0.9875
Epoch 6/25
51/51 [=====] - 57s 1s/step - loss: 0.0182 - accuracy: 0.9945 - val_loss: 0.0470 - val_accuracy: 0.9896
Epoch 7/25
51/51 [=====] - 57s 1s/step - loss: 0.0104 - accuracy: 0.9975 - val_loss: 0.0172 - val_accuracy: 0.9979
Epoch 8/25
51/51 [=====] - 57s 1s/step - loss: 0.0054 - accuracy: 0.9975 - val_loss: 0.0202 - val_accuracy: 0.9958
Epoch 9/25
51/51 [=====] - 57s 1s/step - loss: 0.0027 - accuracy: 0.9988 - val_loss: 0.0370 - val_accuracy: 0.9917
Epoch 10/25
51/51 [=====] - 57s 1s/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0457 - val_accuracy: 0.9896
Epoch 11/25
51/51 [=====] - 57s 1s/step - loss: 0.0058 - accuracy: 0.9982 - val_loss: 0.0190 - val_accuracy: 0.9958
Epoch 12/25
51/51 [=====] - 57s 1s/step - loss: 0.0015 - accuracy: 1.0000 - val_loss: 0.0443 - val_accuracy: 0.9917
Epoch 13/25
51/51 [=====] - 57s 1s/step - loss: 0.0030 - accuracy: 0.9988 - val_loss: 0.0285 - val_accuracy: 0.9937
Epoch 14/25
51/51 [=====] - 57s 1s/step - loss: 0.0143 - accuracy: 0.9951 - val_loss: 0.0665 - val_accuracy: 0.9917
Epoch 15/25
51/51 [=====] - 57s 1s/step - loss: 0.0431 - accuracy: 0.9890 - val_loss: 0.0268 - val_accuracy: 0.9896
Epoch 16/25
51/51 [=====] - 57s 1s/step - loss: 0.0275 - accuracy: 0.9896 - val_loss: 0.0726 - val_accuracy: 0.9833
Epoch 17/25
51/51 [=====] - 57s 1s/step - loss: 0.0078 - accuracy: 0.9975 - val_loss: 0.0739 - val_accuracy: 0.9875
Epoch 18/25
51/51 [=====] - 57s 1s/step - loss: 0.0177 - accuracy: 0.9939 - val_loss: 0.0441 - val_accuracy: 0.9896
Epoch 19/25
51/51 [=====] - 57s 1s/step - loss: 0.0102 - accuracy: 0.9963 - val_loss: 0.0451 - val_accuracy: 0.9917
```

```
Epoch 20/25
51/51 [=====] - 57s 1s/step - loss: 0.0045 - accuracy: 0.9988 - val_loss: 0.0309 - val_accuracy: 0.9958
Epoch 21/25
51/51 [=====] - 57s 1s/step - loss: 4.8169e-04 - accuracy: 1.0000 - val_loss: 0.0331 - val_accuracy: 0.9937
Epoch 22/25
51/51 [=====] - 57s 1s/step - loss: 5.3487e-04 - accuracy: 1.0000 - val_loss: 0.0410 - val_accuracy: 0.9937
Epoch 23/25
51/51 [=====] - 59s 1s/step - loss: 8.9725e-05 - accuracy: 1.0000 - val_loss: 0.0270 - val_accuracy: 0.9958
Epoch 24/25
51/51 [=====] - 58s 1s/step - loss: 7.3278e-05 - accuracy: 1.0000 - val_loss: 0.0493 - val_accuracy: 0.9917
Epoch 25/25
51/51 [=====] - 57s 1s/step - loss: 6.6464e-05 - accuracy: 1.0000 - val_loss: 0.0335 - val_accuracy: 0.9937
```

8. Performance Evaluation

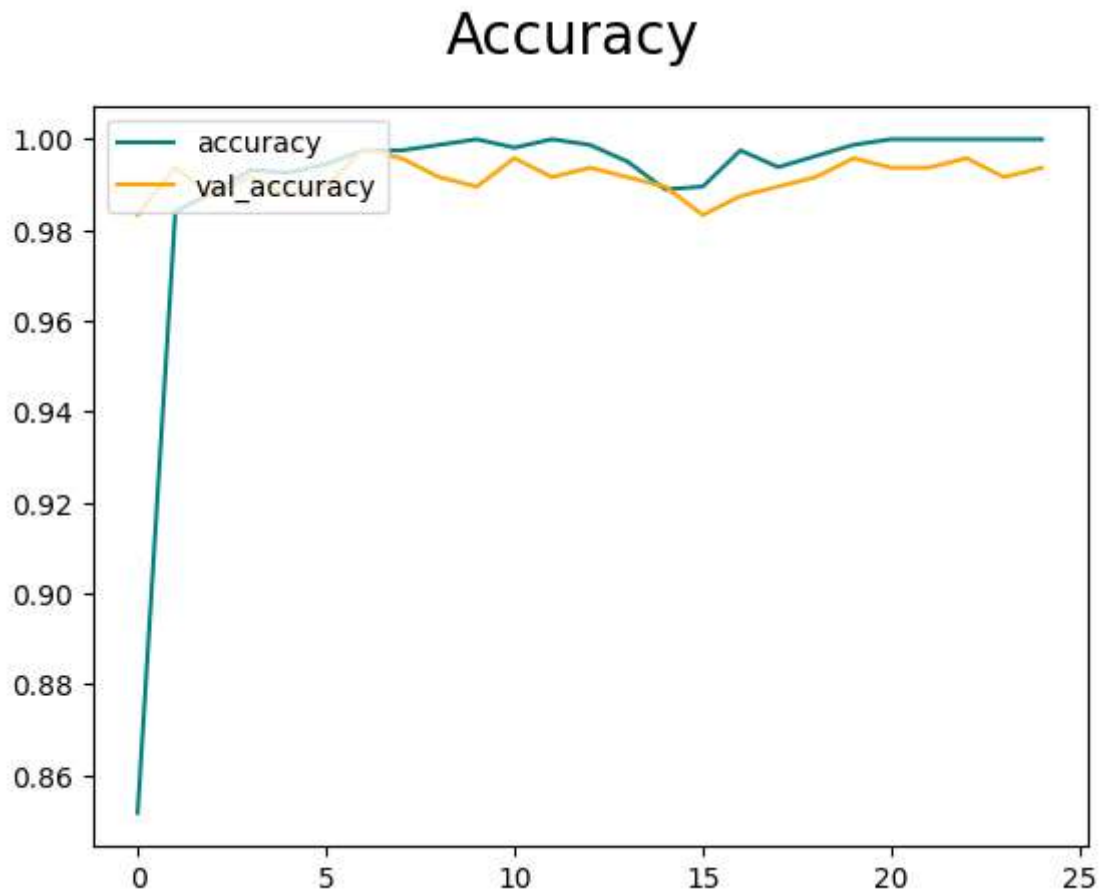
i. Loss

```
In [19]: fig = plt.figure()
plt.plot(hist.history['loss'], color='teal', label='loss')
plt.plot(hist.history['val_loss'], color='orange', label='val_loss')
fig.suptitle('Loss', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```



ii. Accuracy

```
In [20]: fig = plt.figure()
plt.plot(hist.history['accuracy'], color='teal', label='accuracy')
plt.plot(hist.history['val_accuracy'], color='orange', label='val_accuracy')
fig.suptitle('Accuracy', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```



iii. Evaluation Metrics

```
In [21]: pre = Precision()
re = Recall()
acc = BinaryAccuracy()

for batch in test.as_numpy_iterator():
    X, y = batch
    yhat = model.predict(X)
    pre.update_state(y, yhat)
    re.update_state(y, yhat)
    acc.update_state(y, yhat)
```

```
1/1 [=====] - 0s 323ms/step
1/1 [=====] - 0s 211ms/step
1/1 [=====] - 0s 217ms/step
1/1 [=====] - 0s 224ms/step
1/1 [=====] - 0s 213ms/step
1/1 [=====] - 0s 217ms/step
1/1 [=====] - 0s 233ms/step
```

```
In [22]: print(pre.result(), re.result(), acc.result())
```

```
tf.Tensor(1.0, shape=(), dtype=float32) tf.Tensor(0.96363634, shape=(), dtype=float32) tf.Tensor(0.99065423, shape=(), dtype=float32)
```

10. Testing

```
In [33]: img = cv2.imread('test5.jfif')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```

0

200

400

600

800

1000

1200

1400

0 250 500 750 1000

GSA

Supplemental Lease Agreement

Number 1

Lease Number: 00-00000000 Date: 1/1/2000

ADDRESS OF PREMISES: Port of Los Angeles, Block 140, The Rice Estate #0

THIS AGREEMENT, made and entered into this date by and between Negotia Imperial, LLC,

whose address is: 1625 Fay Ave., Suite 200

Los Angeles, CA 90007

hereinafter called the Lessor, and the UNITED STATES OF AMERICA, hereinafter called the Government;

WITNESSES, the parties hereto desire to amend the above Lease.

NOW THEREFORE, these parties for the considerations hereinafter mentioned consent and agree that Lease Number

00-00000000 be amended, effective December 6, 2000, as follows:

Paragraph 21 of the lease is hereby deleted in its entirety and the following is inserted in lieu thereof:

21. Continuation and Continuation Credit:

The Lessor and the Lessee have agreed to a continuing lease arrangement in which the first five years of the term of this lease will

be the first five years of the first term of this lease. The total amount of the continuation is \$1,000,000. The Lessor shall pay the

amount of the continuation as provided in the lease agreement. In accordance with the "Notice of Continuation and Continuation Credit"

paragraph, the Lessor has agreed to pay the amount of the continuation as provided in the lease agreement.

(Continuation Credit) The Continuation Credit is \$1,000,000. The Lessor agrees to pay the Continuation Credit to the

Lessee in accordance with the "Notice of Continuation and Continuation Credit" paragraph in the 2010 schedule to and being a part of this lease.

Notwithstanding Paragraph 1 of this Schedule Part 2, the first rental payments due and being under this lease shall be reduced to fully recognize

the Continuation Credit. The reduction in first year shall commence with the first month of the rental payments and continue as provided in the

schedule for adjusted monthly rent.

First Month's Rental Payment \$24,000.00 minus pro-rated Continuation Credit of \$1,000,000 adjusted First Month's Rent

Second Month's Rental Payment \$24,000.00 minus pro-rated Continuation Credit of \$1,000,000 adjusted Second Month's Rent

Third Month's Rental Payment \$24,000.00 minus pro-rated Continuation Credit of \$1,000,000 adjusted Third Month's Rent

All other terms and conditions of the Lease shall remain in force and effect. WITNESSES HEREOF, the parties

subscribed their names on the above date.

LESSOR: Negotia Imperial, LLC

BY: [Signature] (Type)

IN THE PRESENCE OF

BY: [Signature] (Type)

UNITED STATES OF AMERICA

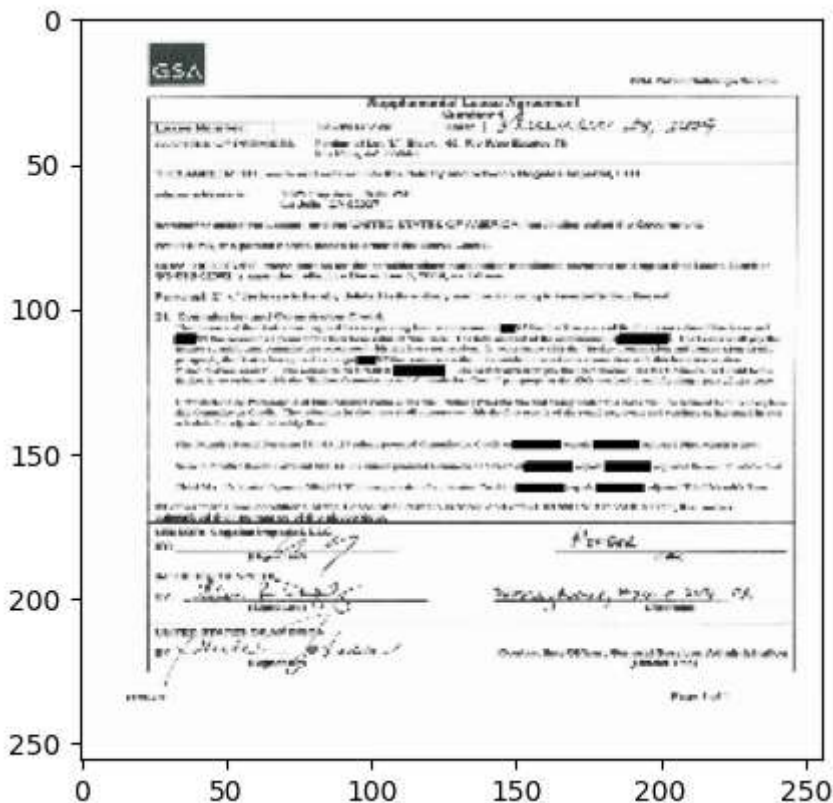
BY: [Signature] (Type)

Contracting Officer, General Services Administration

(Print Name)

Page 1 of 1

```
In [34]: resize = tf.image.resize(cv2.cvtColor(img, cv2.COLOR_BGR2RGB), (256,256))
plt.imshow(resize.numpy().astype(int))
plt.show()
```



```
In [35]: yhat = model.predict(np.expand_dims(resize/255, 0))
```

```
1/1 [=====] - 0s 43ms/step
```

```
In [36]: if yhat > 0.5:
    print(f'Predicted class is Signature')
    print(f"Certainty = {float(yhat)*100}")
else:
    print(f'Predicted class is Portrait')
    print(f"Certainty = {float(1-yhat)*100}")
```

Predicted class is Signature
Certainty = 84.3981683254242

11. Saving the Model

```
In [27]: model.save(os.path.join('models', 'model1.h5'))
```

```
In [28]: #new_model = Load_model('C:\\Users\\chana\\Downloads\\models\\imageclassifiere
```

 Created **Deepnote**(https://deepnote.com?utm_source=created-in-deepnote-cell&projectId=e5eb307f-57c9-4fe9-995d-1913de4a6e89)
in