0. Outline

1. Setup & Load Data
   - Load required libraries
   - Setting up the directory
   - Loading and exploring the dataset
2. The deep learning model
   - Build the model
   - Splitting the data
   - Fitting the model
   - Training the model and visualizing average scores
   - Plotting model's preformance
3. Evaluating model's performance
   - Evaluate the model

1. Setup & Load Data

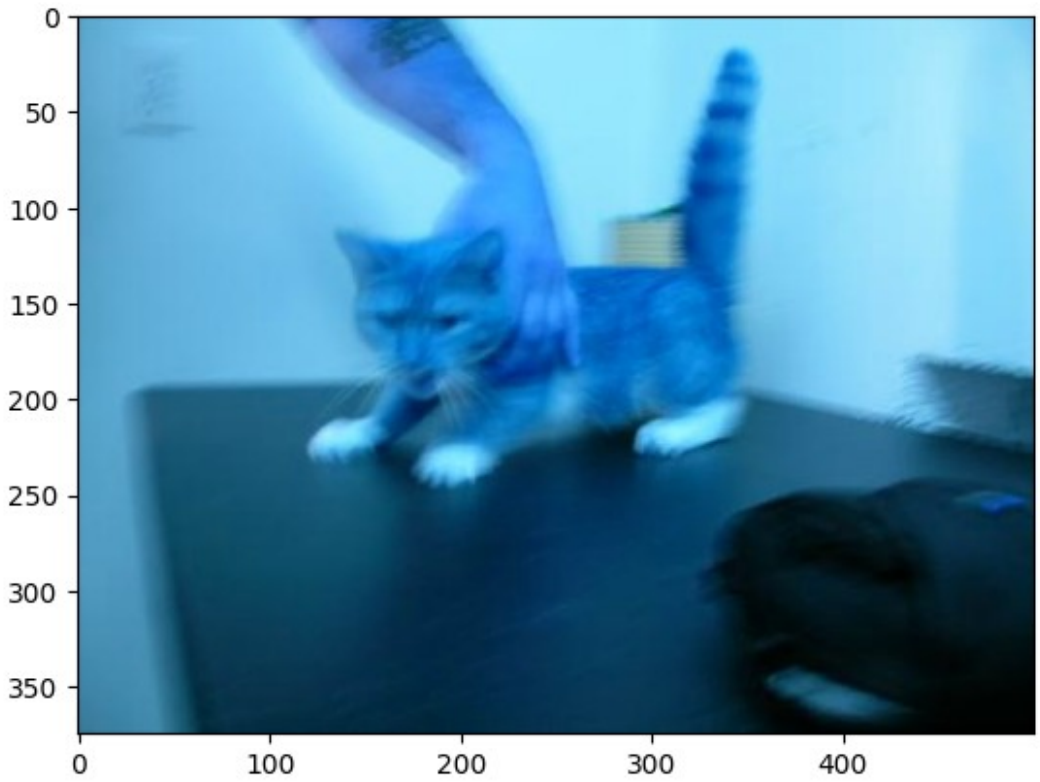
1.1 Load Required Libraries:

```
import tensorflow as tf
import os
from matplotlib import pyplot as plt
import cv2
import imghdr
import numpy as np
from tensorflow import keras
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
from keras.callbacks import ReduceLROnPlateau
from silence_tensorflow import silence_tensorflow
silence_tensorflow()
from keras.layers import Rescaling, RandomFlip, RandomRotation,
RandomZoom, Resizing
import numpy as np
import splitfolders
import seaborn as sns
from keras.metrics import Precision, Recall, BinaryAccuracy
```

1.2 Setting Up The Directory:

```
data_dir = 'CatsDogs_ds'

os.listdir(data_dir)

['Cats', 'Dogs']

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!= 'jpeg':
                print('image does not have 3 channels
{}'.format(image_path))
                os.remove(image_path)
            if tip not in image_exts:
                print('image is not in ext list
{}'.format(image_path))
                os.remove(image_path)
        except Exception as e:
            print('Issue with image {}'.format(image_path))

splitfolders.ratio(data_dir, output='CatsDogs_Split', seed= 1234,
ratio = (0.8, 0.2))

Train_dir = 'CatsDogs_Split/train'
Test_dir = 'CatsDogs_Split/test'

#Visualizing an image
img = cv2.imread(os.path.join('CatsDogs_Split/train','Cats', '0.jpg'))
plt.imshow(img)
plt.show()
```



1.3 Loading & Exploring The Dataset:

```
#Building an image dataset on the fly, no need to build the labels,
the classes.
#Note: this will resize the images to 64x64.
#Images will be shuffled as well.
data =
tf.keras.utils.image_dataset_from_directory('CatsDogs_Split/train',
image_size=(64,64))
```

```
Found 19785 files belonging to 2 classes.
```

```
#This will allow us to access the generator from our data pipeline
data_iterator = data.as_numpy_iterator()
```

```
#Get another batch from the iterator
batch = data_iterator.next()
```

```
len(batch)
```

```
2
```

- There are 2 parts of the dataset:
  - The actual dataset images stored as numpy arrays
  - labels

```
batch[0].shape
```

```
(32, 64, 64, 3)
```

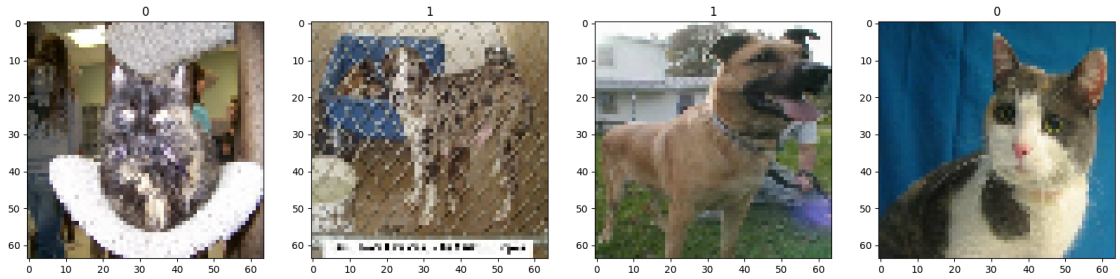
- Batch size is 32, image size is 256 by 256 by 3 channels

```
batch[1]
```

```
array([1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0])
```

- Labels are either 0 or 1, meaning Cat or Dog.

```
#Plotting the images to determine which value means Cat and which
value means Dog
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])
```



- From the above plot, we can determine that label 1 means dog and label 0 means cat.

2. The Deep Learning Model

2.1 Building The Model:

```
def cnn_model():
    model = Sequential()
    #Resizing the images
    model.add(Resizing(100,100))
    #- Scaling the image values between 0 & 1 instead of 0 to 255.
This will help our Deep learning model to optimize faster and produce
better results.
    model.add(Rescaling(1./255))
    #Data Augmentation to prevent overfitting
    model.add(RandomFlip('horizontal'))
    model.add(RandomRotation(0.2))
    model.add(RandomZoom(0.2))

    #Adding a convolutional layer and a MaxPooling layer
    #Each filter is going to be 3x3 in size and we are moving by 1
step/stride each time
    #Activation is going to be Rectified Linear Unit(relu), meaning
any output below zero is going to be equal 0 and any other positive
value is going to be preserved
    #MaxPooling helps us condense the information we get after the
activation with the aim of getting the maximum values
    #MaxPooling halves the output of the Convolutional layer

    #First Layer
    model.add(Conv2D(32, (3,3), 1, activation = 'relu'))
    model.add(MaxPooling2D(2,2))

    #Second Layer
    model.add(Conv2D(64, (3,3), 1, activation = 'relu'))
    model.add(MaxPooling2D(2,2))

    #Third Layer
    model.add(Conv2D(128, (3,3), 1, activation = 'relu'))
    model.add(MaxPooling2D(2,2))

    model.add(Dropout(0.2))

    #We have condensed the rows and the width and the number of
filters will form the channel value
    #Now the aim when flattening the data is condense the channel
value to a single value
    model.add(Flatten())
```

```
    model.add(Dense(128, activation = 'relu'))

    model.add(Dropout(0.5))
    #Now we are condensing them even more to get the final value (0 or
1 --> cat or dog)
    model.add(Dense(1, activation = 'sigmoid'))

    #List of optimizers can be checked --> (tf.optimizers.)
    model.compile('adam', loss = tf.losses.BinaryCrossentropy(),
metrics = ['accuracy'])

    return model
```

2.2 Splitting The Data:

```
splitfolders.ratio(Train_dir, output='train_val_split', seed=
np.random.randint(1,1000,1)[0], ratio = (0.8, 0.2))
```

2.3 Fitting The Model:

```
def fit_model(tr, val):
    model = None
    model = cnn_model()

    #fitting the model
    hist = model.fit(tr, validation_data = val, epochs=25, verbose =
1, callbacks = [ReduceLROnPlateau()])

    #Saving our model
    model.save("Trials/100x100_3layers_stride1")
    return hist
```

2.4 Training The Model & Visualizing Average Scores:

```
train_data =
tf.keras.utils.image_dataset_from_directory('train_val_split/train',
image_size = (100,100))
val_data =
tf.keras.utils.image_dataset_from_directory('train_val_split/val',
image_size = (100,100))
hist = fit_model(train_data, val_data)
print("=======*10, end="\n\n\n")
```

```
Found 15827 files belonging to 2 classes.
Found 3958 files belonging to 2 classes.
Epoch 1/25
495/495 [==============================] - 212s 422ms/step - loss:
0.6683 - accuracy: 0.5831 - val_loss: 0.6227 - val_accuracy: 0.6599 -
lr: 0.0010
Epoch 2/25
495/495 [==============================] - 212s 428ms/step - loss:
0.6143 - accuracy: 0.6668 - val_loss: 0.5562 - val_accuracy: 0.7031 -
lr: 0.0010
```

```
Epoch 3/25
495/495 [==============================] - 208s 420ms/step - loss:
0.5797 - accuracy: 0.6984 - val_loss: 0.5267 - val_accuracy: 0.7340 -
lr: 0.0010
Epoch 4/25
495/495 [==============================] - 196s 395ms/step - loss:
0.5576 - accuracy: 0.7153 - val_loss: 0.5029 - val_accuracy: 0.7461 -
lr: 0.0010
Epoch 5/25
495/495 [==============================] - 215s 434ms/step - loss:
0.5367 - accuracy: 0.7315 - val_loss: 0.4948 - val_accuracy: 0.7587 -
lr: 0.0010
Epoch 6/25
495/495 [==============================] - 217s 438ms/step - loss:
0.5192 - accuracy: 0.7450 - val_loss: 0.5087 - val_accuracy: 0.7481 -
lr: 0.0010
Epoch 7/25
495/495 [==============================] - 211s 426ms/step - loss:
0.5058 - accuracy: 0.7516 - val_loss: 0.4670 - val_accuracy: 0.7741 -
lr: 0.0010
Epoch 8/25
495/495 [==============================] - 209s 422ms/step - loss:
0.4946 - accuracy: 0.7636 - val_loss: 0.4756 - val_accuracy: 0.7673 -
lr: 0.0010
Epoch 9/25
495/495 [==============================] - 189s 381ms/step - loss:
0.4792 - accuracy: 0.7708 - val_loss: 0.4591 - val_accuracy: 0.7807 -
lr: 0.0010
Epoch 10/25
495/495 [==============================] - 208s 420ms/step - loss:
0.4719 - accuracy: 0.7774 - val_loss: 0.4332 - val_accuracy: 0.7946 -
lr: 0.0010
Epoch 11/25
495/495 [==============================] - 188s 379ms/step - loss:
0.4636 - accuracy: 0.7844 - val_loss: 0.4348 - val_accuracy: 0.7918 -
lr: 0.0010
Epoch 12/25
495/495 [==============================] - 194s 393ms/step - loss:
0.4512 - accuracy: 0.7884 - val_loss: 0.4523 - val_accuracy: 0.7948 -
lr: 0.0010
Epoch 13/25
495/495 [==============================] - 208s 421ms/step - loss:
0.4363 - accuracy: 0.7988 - val_loss: 0.5342 - val_accuracy: 0.7276 -
lr: 0.0010
Epoch 14/25
495/495 [==============================] - 191s 386ms/step - loss:
0.4313 - accuracy: 0.8024 - val_loss: 0.3934 - val_accuracy: 0.8231 -
lr: 0.0010
Epoch 15/25
495/495 [==============================] - 166s 334ms/step - loss:
```

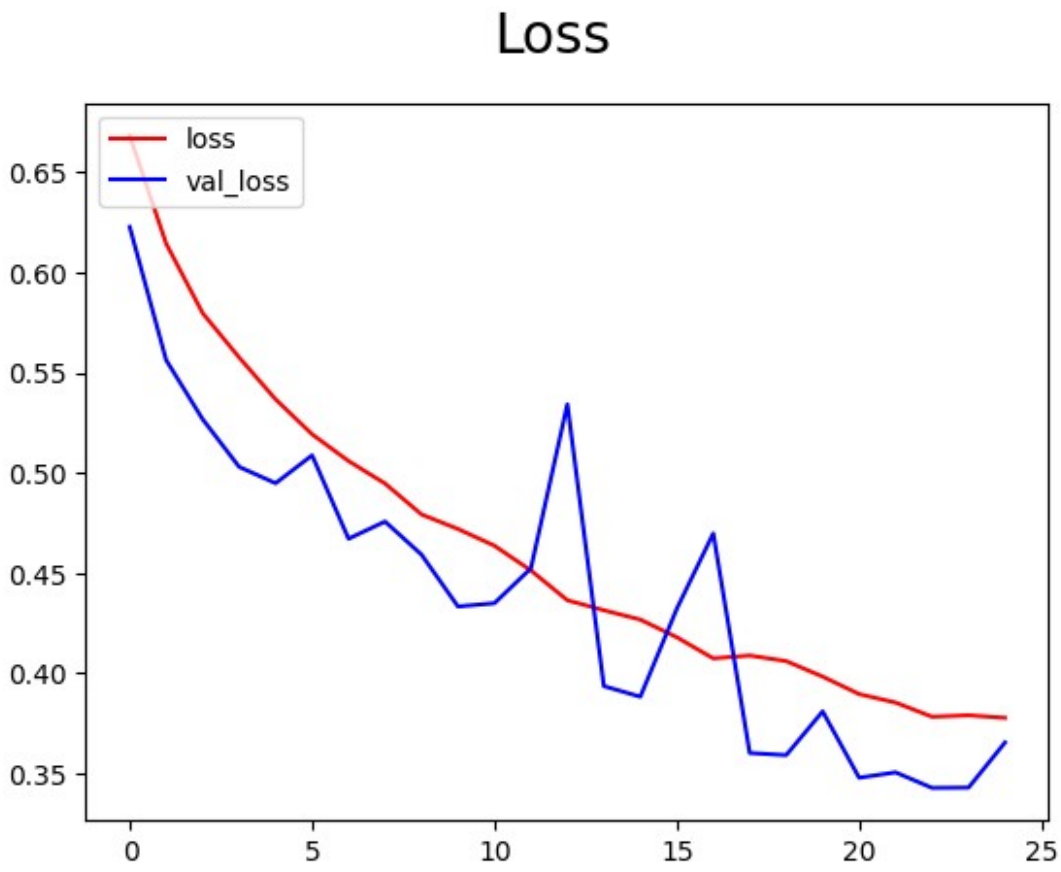
```
0.4267 - accuracy: 0.8050 - val_loss: 0.3882 - val_accuracy: 0.8236 -
lr: 0.0010
Epoch 16/25
495/495 [==============================] - 151s 305ms/step - loss:
0.4178 - accuracy: 0.8095 - val_loss: 0.4321 - val_accuracy: 0.7893 -
lr: 0.0010
Epoch 17/25
495/495 [==============================] - 159s 320ms/step - loss:
0.4073 - accuracy: 0.8108 - val_loss: 0.4697 - val_accuracy: 0.7739 -
lr: 0.0010
Epoch 18/25
495/495 [==============================] - 153s 309ms/step - loss:
0.4087 - accuracy: 0.8155 - val_loss: 0.3600 - val_accuracy: 0.8393 -
lr: 0.0010
Epoch 19/25
495/495 [==============================] - 158s 318ms/step - loss:
0.4059 - accuracy: 0.8118 - val_loss: 0.3590 - val_accuracy: 0.8380 -
lr: 0.0010
Epoch 20/25
495/495 [==============================] - 158s 318ms/step - loss:
0.3982 - accuracy: 0.8214 - val_loss: 0.3808 - val_accuracy: 0.8282 -
lr: 0.0010
Epoch 21/25
495/495 [==============================] - 159s 321ms/step - loss:
0.3895 - accuracy: 0.8259 - val_loss: 0.3477 - val_accuracy: 0.8459 -
lr: 0.0010
Epoch 22/25
495/495 [==============================] - 353s 713ms/step - loss:
0.3852 - accuracy: 0.8250 - val_loss: 0.3504 - val_accuracy: 0.8446 -
lr: 0.0010
Epoch 23/25
495/495 [==============================] - 384s 777ms/step - loss:
0.3781 - accuracy: 0.8313 - val_loss: 0.3426 - val_accuracy: 0.8459 -
lr: 0.0010
Epoch 24/25
495/495 [==============================] - 395s 797ms/step - loss:
0.3789 - accuracy: 0.8316 - val_loss: 0.3428 - val_accuracy: 0.8509 -
lr: 0.0010
Epoch 25/25
495/495 [==============================] - 396s 799ms/step - loss:
0.3776 - accuracy: 0.8306 - val_loss: 0.3653 - val_accuracy: 0.8360 -
lr: 0.0010

WARNING:absl:Found untraced functions such as
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _update_step_xla while saving (showing 4
of 4). These functions will not be directly callable after loading.
```

```
=================================================================


model = load_model('Trials/100x100_3layers_stride1')

model.summary()

Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resizing_6 (Resizing)       (None, 100, 100, 3)       0

 rescaling_5 (Rescaling)     (None, 100, 100, 3)       0

 random_flip_2 (RandomFlip)  (None, 100, 100, 3)       0

 random_rotation_2 (RandomRo  (None, 100, 100, 3)      0
 tation)

 random_zoom_2 (RandomZoom)  (None, 100, 100, 3)       0

 conv2d_18 (Conv2D)          (None, 98, 98, 32)        896

 max_pooling2d_18 (MaxPoolin  (None, 49, 49, 32)       0
 g2D)

 conv2d_19 (Conv2D)          (None, 47, 47, 64)        18496

 max_pooling2d_19 (MaxPoolin  (None, 23, 23, 64)       0
 g2D)

 conv2d_20 (Conv2D)          (None, 21, 21, 128)       73856

 max_pooling2d_20 (MaxPoolin  (None, 10, 10, 128)      0
 g2D)

 dropout_8 (Dropout)         (None, 10, 10, 128)       0

 flatten_6 (Flatten)         (None, 12800)             0

 dense_12 (Dense)            (None, 128)               1638528

 dropout_9 (Dropout)         (None, 128)               0

 dense_13 (Dense)            (None, 1)                 129

=================================================================
Total params: 1,731,905
Trainable params: 1,731,905
```

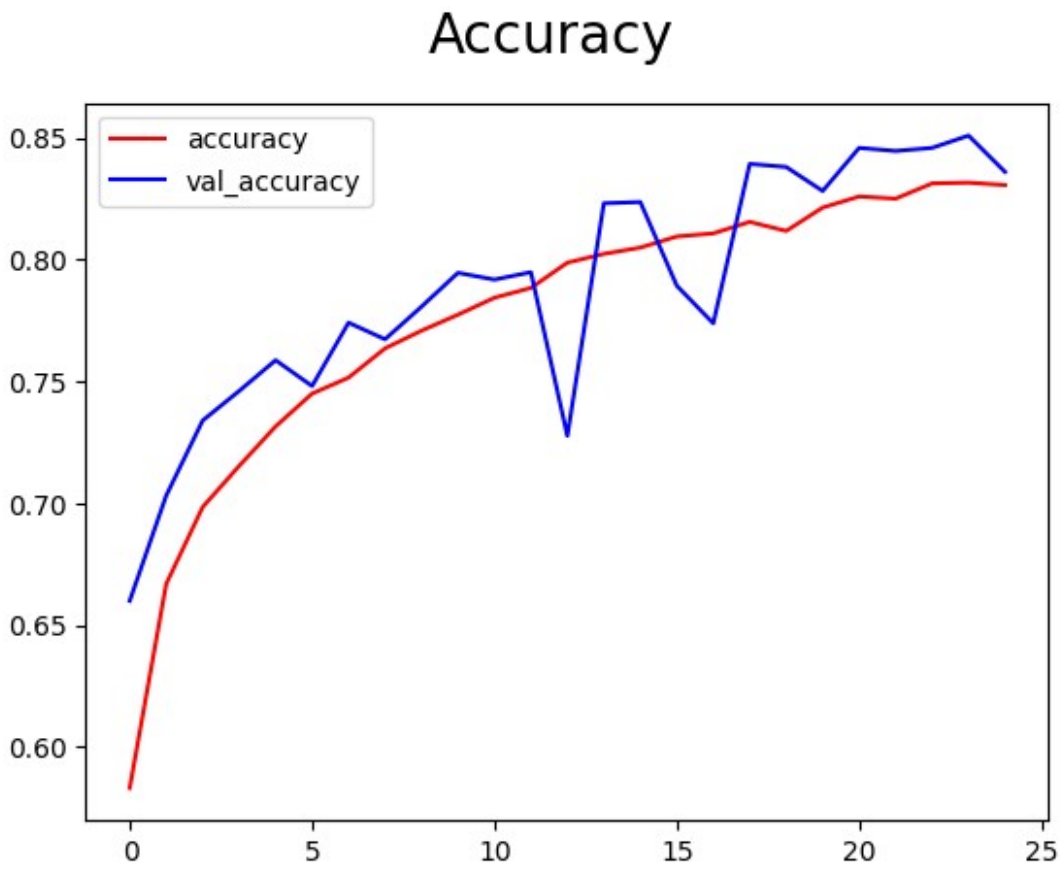
```
Non-trainable params: 0
_________________________________________________________________
```

2.5 Plotting Model's Performance:

```
fig = plt.figure()
plt.plot(hist.history['loss'], color = 'red', label = 'loss')
plt.plot(hist.history['val_loss'], color = 'blue', label = 'val_loss')
fig.suptitle('Loss', fontsize = 20)
plt.legend(loc='upper left')
plt.show()
```




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



3. Evaluating Model's Performance

3.1 Evaluate The Model:

```
model = load_model('Trials/100x100_3layers_stride1')

#Visualizing model's summary:
model.summary()

test =
tf.keras.utils.image_dataset_from_directory('CatsDogs_Split/test',
image_size=(100,100))

Found 4947 files belonging to 2 classes.

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 112ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 67ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 111ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 68ms/step
1/1 [==============================] - 0s 71ms/step
1/1 [==============================] - 0s 67ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 56ms/step
```

```
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 67ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 55ms/step
```

```
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 63ms/step
```

```
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 102ms/step
```

```
print(f'Precision:{pre.result().numpy()}, Recall:
{re.result().numpy()}, Accuracy:{acc.result().numpy()}')
```

```
Precision:0.8587548732757568, Recall:0.8935222625732422,
Accuracy:0.8734586834907532
```