ene-image-classification-using-cnn

April 27, 2024

[ ]:

Objective: Classifify images in three categories

Step1: Import All dependencies

[4]:
```
! pip install -q tensorflow
! pip install -q gradio
```

[ ]:

[5]:
```
# import the necessary packages
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Activation
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras import backend as K
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.utils import to_categorical

from sklearn.model_selection import train_test_split

from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import random
import cv2
import os

import warnings
from tqdm import tqdm_notebook as tqdm
import itertools
```

```
import tensorflow as tf
print(tf.__version__)
warnings.filterwarnings("ignore")
SEED = 42   # set random seed
```

```
2.15.0
```

Step 2: Load Dataset

[6]:
```
from google.colab import drive
drive.mount('/content/drive')

import os
os.chdir("/content/drive/MyDrive/ Deep Learning Projects/ Image classification␣
↪using CNN")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:
```

```

[7]:
```
!dir
```

```
cnn_model_50.h5  Multi_Class_Image_Classification_Using_CNN.ipynb
dataset.rar      MultiClass\ _Image_Classification_Using_CNN.ipynb.ipynb
FINAL_CNN.html   test_examples.rar
```

[9]:
```
!pip install patool
```

```
Requirement already satisfied: patool in /usr/local/lib/python3.10/dist-packages
(2.2.0)
```

[10]:
```
import patoolib
patoolib.extract_archive("dataset.rar")
patoolib.extract_archive("test_examples.rar")
```

```
INFO patool: Extracting dataset.rar …
INFO:patool:Extracting dataset.rar …
INFO patool: running /usr/bin/unrar x -- "/content/drive/MyDrive/ Deep Learning
Projects/ Image classification using CNN/dataset.rar"
INFO:patool:running /usr/bin/unrar x -- "/content/drive/MyDrive/ Deep Learning
Projects/ Image classification using CNN/dataset.rar"
INFO patool:     with cwd='./Unpack_c9dim25w', input=''
INFO:patool:    with cwd='./Unpack_c9dim25w', input=''
INFO patool: … dataset.rar extracted to `dataset'.
INFO:patool:… dataset.rar extracted to `dataset'.
INFO patool: Extracting test_examples.rar …
INFO:patool:Extracting test_examples.rar …
```

```
INFO patool: running /usr/bin/unrar x -- "/content/drive/MyDrive/ Deep Learning
Projects/ Image classification using CNN/test_examples.rar"
INFO:patool:running /usr/bin/unrar x -- "/content/drive/MyDrive/ Deep Learning
Projects/ Image classification using CNN/test_examples.rar"
INFO patool:     with cwd='./Unpack_cqaz4jx3', input=''
INFO:patool:    with cwd='./Unpack_cqaz4jx3', input=''
INFO patool: … test_examples.rar extracted to `test_examples'.
INFO:patool:… test_examples.rar extracted to `test_examples'.
```

[10]:
```
'test_examples'
```

Step 3: Build CNN Model

[11]:
```
# create CNN model
class LeNet:
    @staticmethod
    def build(width, height, depth, classes):

        # initialize the model
        model = Sequential()
        inputShape = (height, width, depth) # (h, h, chanel)

        # if we are using "channels first", update the input shape
        if K.image_data_format() == "channels_first":
            inputShape = (depth, height, width)

        # first set of CONV => RELU => POOL layers
        model.add(Conv2D(50, (5, 5), padding="same", input_shape=inputShape))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

        # second set of CONV => RELU => POOL layers
        model.add(Conv2D(150, (5, 5), padding="same"))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

        # third set of CONV => RELU => POOL layers
        model.add(Conv2D(200, (5, 5), padding="same"))
        model.add(Activation("relu"))
        model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

        # ------------------------ANN-----------------------
        # first (and only) set of FC => RELU layers
        model.add(Flatten())
        model.add(Dense(2000))
        model.add(Activation("relu"))

        # softmax classifier
```

```
        model.add(Dense(classes))
        model.add(Activation("softmax"))

        # return the constructed network architecture
        return model
```

[ ]:

[ ]:

Step 4: Data Reading

[12]:
```
DATASET= "dataset" # this folder must contain three subfolder with images
MODEL="Scene.model"# name to store the model on dis
PLOT="plot.png" # plot name
```

[13]:
```
# initialize the data and labels
print("[INFO] loading images...")
data = []  # x
labels = [] # y

# grab the image paths and randomly shuffle them
imagePaths = sorted(list(paths.list_images("dataset")))
random.seed(42)
random.shuffle(imagePaths)

# progress bar
with tqdm(total=len(imagePaths)) as pbar:

    # loop over the input images
    for idx, imagePath in enumerate(imagePaths):
        # load the image, pre-process it, and store it in the data list
        image = cv2.imread(imagePath)
        image = cv2.resize(image, (28, 28))
        image = img_to_array(image)
        data.append(image)

        # extract the class label from the image path and update the
        # labels list
        label = imagePath.split(os.path.sep)[-2]

        if label == "Buildings":
            label = 0
        elif label == "Forest":
            label = 1
        elif label == "Sea":
            label = 2
```

```
        labels.append(label)

        # update the progressbar
        pbar.update(1)
```

```
[INFO] loading images…
  0%|          | 0/1326 [00:00<?, ?it/s]
```

[14]:
```
# check shape of single image
data[0].shape
```

[14]:
```
(28, 28, 3)
```

[15]:
```
data[0]
```

[15]:
```
array([[[237., 209., 198.],
        [240., 215., 205.],
        [237., 215., 204.],
        …,
        [188., 143., 116.],
        [200., 156., 125.],
        [195., 138., 106.]],

       [[235., 207., 192.],
        [232., 210., 198.],
        [236., 217., 209.],
        …,
        [214., 169., 145.],
        [182., 117.,  80.],
        [174., 113.,  77.]],

       [[228., 199., 184.],
        [243., 219., 206.],
        [245., 225., 214.],
        …,
        [206., 154., 123.],
        [209., 156., 120.],
        [181., 118.,  81.]],

       …,

       [[155., 164., 172.],
        [143., 152., 165.],
        [137., 149., 155.],
        …,
        [ 66.,  57.,  54.],
```

```
        [ 17.,   9.,   7.],
        [ 14.,   6.,   6.]],

       [[ 97., 113., 129.],
        [113., 129., 141.],
        [119., 130., 140.],
        …,
        [ 19.,  15.,  16.],
        [ 27.,  25.,  23.],
        [ 15.,  12.,  12.]],

       [[147., 159., 169.],
        [138., 149., 158.],
        [127., 136., 145.],
        …,
        [108., 114., 112.],
        [100., 100., 100.],
        [ 96.,  98.,  98.]]], dtype=float32)
```

Step 5: Data Spliting

[16]:
```
# perform data normalisation
data=np.array(data,dtype="float")/255.0
labels = np.array(labels)
```

[17]:
```
data[0]
```

[17]:
```
array([[[0.92941176, 0.81960784, 0.77647059],
        [0.94117647, 0.84313725, 0.80392157],
        [0.92941176, 0.84313725, 0.8       ],
        …,
        [0.7372549 , 0.56078431, 0.45490196],
        [0.78431373, 0.61176471, 0.49019608],
        [0.76470588, 0.54117647, 0.41568627]],

       [[0.92156863, 0.81176471, 0.75294118],
        [0.90980392, 0.82352941, 0.77647059],
        [0.9254902 , 0.85098039, 0.81960784],
        …,
        [0.83921569, 0.6627451 , 0.56862745],
        [0.71372549, 0.45882353, 0.31372549],
        [0.68235294, 0.44313725, 0.30196078]],

       [[0.89411765, 0.78039216, 0.72156863],
        [0.95294118, 0.85882353, 0.80784314],
        [0.96078431, 0.88235294, 0.83921569],
        …,
```

```
        [0.80784314, 0.60392157, 0.48235294],
        [0.81960784, 0.61176471, 0.47058824],
        [0.70980392, 0.4627451 , 0.31764706]],

       …,

       [[0.60784314, 0.64313725, 0.6745098 ],
        [0.56078431, 0.59607843, 0.64705882],
        [0.5372549 , 0.58431373, 0.60784314],
        …,
        [0.25882353, 0.22352941, 0.21176471],
        [0.06666667, 0.03529412, 0.02745098],
        [0.05490196, 0.02352941, 0.02352941]],

       [[0.38039216, 0.44313725, 0.50588235],
        [0.44313725, 0.50588235, 0.55294118],
        [0.46666667, 0.50980392, 0.54901961],
        …,
        [0.0745098 , 0.05882353, 0.0627451 ],
        [0.10588235, 0.09803922, 0.09019608],
        [0.05882353, 0.04705882, 0.04705882]],

       [[0.57647059, 0.62352941, 0.6627451 ],
        [0.54117647, 0.58431373, 0.61960784],
        [0.49803922, 0.53333333, 0.56862745],
        …,
        [0.42352941, 0.44705882, 0.43921569],
        [0.39215686, 0.39215686, 0.39215686],
        [0.37647059, 0.38431373, 0.38431373]]])
```

[18]:
```
#Split the data into train test
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.25,␣
 ↪random_state=42)
```

[19]:
```
# verify shapes
print(trainX.shape)
print(testX.shape)
print(trainY.shape)
print(testY.shape)
```

```
(994, 28, 28, 3)
(332, 28, 28, 3)
(994,)
(332,)
```

[20]:
```
testY
```

[20]:
```
array([0, 1, 1, 2, 0, 2, 2, 2, 0, 0, 2, 2, 2, 1, 2, 1, 0, 1, 2, 2, 0, 0,
       1, 0, 1, 0, 0, 2, 2, 2, 2, 1, 2, 1, 1, 0, 0, 0, 1, 1, 2, 1, 1, 0,
       2, 1, 1, 0, 0, 1, 2, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 2, 2, 2,
       1, 2, 1, 1, 1, 2, 2, 0, 0, 0, 2, 2, 2, 2, 0, 1, 0, 0, 2, 0, 2, 2,
       0, 0, 2, 2, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       2, 1, 0, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 0, 0, 2, 2, 2, 1, 0, 2, 0,
       0, 2, 2, 0, 1, 0, 1, 1, 0, 0, 2, 2, 1, 1, 2, 0, 1, 2, 0, 0, 2, 2,
       2, 0, 2, 1, 0, 0, 2, 2, 0, 1, 1, 0, 0, 1, 2, 2, 2, 0, 1, 2, 1, 1,
       0, 1, 0, 0, 0, 1, 0, 2, 1, 2, 2, 0, 0, 1, 1, 0, 0, 0, 2, 2, 0, 2,
       0, 0, 0, 1, 2, 1, 0, 2, 1, 1, 2, 2, 0, 1, 0, 0, 2, 1, 2, 1, 0, 0,
       0, 0, 1, 1, 1, 2, 2, 0, 2, 2, 1, 1, 1, 0, 2, 0, 0, 2, 1, 1, 0, 0,
       0, 1, 0, 0, 2, 0, 1, 0, 0, 1, 0, 0, 2, 2, 2, 0, 2, 0, 0, 2, 2, 2,
       2, 2, 2, 2, 0, 2, 0, 0, 2, 0, 0, 2, 2, 2, 0, 1, 2, 2, 0, 2, 2, 2,
       2, 0, 1, 2, 0, 1, 0, 1, 0, 0, 0, 1, 0, 2, 1, 1, 1, 2, 1, 0, 0, 2,
       1, 0, 1, 1, 0, 2, 1, 0, 0, 1, 0, 0, 2, 1, 2, 1, 1, 0, 0, 0, 2, 1,
       0, 0])
```

[21]:
```
# Perform one hot label encoding
trainY = to_categorical(trainY, num_classes=3)
testY = to_categorical(testY, num_classes=3)
```

[22]:
```
trainY[0]
```

[22]:
```
array([1., 0., 0.], dtype=float32)
```

Step 6: Data Preprocessing (Augumentation)

[23]:
```
# construct the image generator for data augmentation
aug= ImageDataGenerator(rotation_range=30,
                        width_shift_range=0.1,
                        height_shift_range=0.1,
                        shear_range=0.2,
                        zoom_range=0.2,
                        horizontal_flip=True,
                        fill_mode="nearest")
```

Step 7: Compile and Training

[24]:
```
INIT_LR = 1e-3
BS = 32

# initialize the model
print("[INFO] compiling model...")
model=LeNet.build(width=28,height=28,depth=3,classes=3)
opt= Adam(learning_rate=INIT_LR)

model.compile(loss="categorical_crossentropy", optimizer=opt,␣
 ↪metrics=["accuracy"])
```

```
print("[INFO] model complied...")
```

```
[INFO] compiling model…
[INFO] model complied…
```

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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 50)        3800

 activation (Activation)     (None, 28, 28, 50)        0

 max_pooling2d (MaxPooling2  (None, 14, 14, 50)        0
 D)

 conv2d_1 (Conv2D)           (None, 14, 14, 150)       187650

 activation_1 (Activation)   (None, 14, 14, 150)       0

 max_pooling2d_1 (MaxPoolin  (None, 7, 7, 150)         0
 g2D)

 conv2d_2 (Conv2D)           (None, 7, 7, 200)         750200

 activation_2 (Activation)   (None, 7, 7, 200)         0

 max_pooling2d_2 (MaxPoolin  (None, 3, 3, 200)         0
 g2D)

 flatten (Flatten)           (None, 1800)              0

 dense (Dense)               (None, 2000)              3602000

 activation_3 (Activation)   (None, 2000)              0

 dense_1 (Dense)             (None, 3)                 6003

 activation_4 (Activation)   (None, 3)                 0

=================================================================
Total params: 4549653 (17.36 MB)
Trainable params: 4549653 (17.36 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

[26]:
```
# train the network
EPOCH = 50
print("[INFO] training network....")
H= model.fit(x=aug.flow(trainX,trainY,batch_size=BS),
             validation_data=(testX, testY),
              steps_per_epoch=len(trainX) // BS,
              epochs=EPOCH,
              verbose=1)

# save the model and label binarizer to disk
print("[INFO] serializing network ...")
model.save("cnn_model_{}.h5".format(EPOCH))
```

```
[INFO] training network…
Epoch 1/50
31/31 [==============================] - 8s 71ms/step - loss: 1.0094 - accuracy:
0.5010 - val_loss: 0.7147 - val_accuracy: 0.6928
Epoch 2/50
31/31 [==============================] - 1s 22ms/step - loss: 0.8036 - accuracy:
0.6227 - val_loss: 0.7188 - val_accuracy: 0.6687
Epoch 3/50
31/31 [==============================] - 1s 24ms/step - loss: 0.7293 - accuracy:
0.6632 - val_loss: 0.7046 - val_accuracy: 0.7078
Epoch 4/50
31/31 [==============================] - 1s 23ms/step - loss: 0.7196 - accuracy:
0.6809 - val_loss: 0.6865 - val_accuracy: 0.7229
Epoch 5/50
31/31 [==============================] - 1s 23ms/step - loss: 0.6241 - accuracy:
0.7349 - val_loss: 0.4606 - val_accuracy: 0.8042
Epoch 6/50
31/31 [==============================] - 1s 22ms/step - loss: 0.6049 - accuracy:
0.7464 - val_loss: 0.5156 - val_accuracy: 0.7982
Epoch 7/50
31/31 [==============================] - 1s 21ms/step - loss: 0.5461 - accuracy:
0.7817 - val_loss: 0.5220 - val_accuracy: 0.7771
Epoch 8/50
31/31 [==============================] - 1s 24ms/step - loss: 0.5413 - accuracy:
0.7786 - val_loss: 0.4220 - val_accuracy: 0.8313
Epoch 9/50
31/31 [==============================] - 1s 29ms/step - loss: 0.5223 - accuracy:
0.7983 - val_loss: 0.5797 - val_accuracy: 0.7380
Epoch 10/50
31/31 [==============================] - 1s 36ms/step - loss: 0.5530 - accuracy:
0.7890 - val_loss: 0.8799 - val_accuracy: 0.6265
Epoch 11/50
31/31 [==============================] - 1s 34ms/step - loss: 0.5451 - accuracy:
0.7900 - val_loss: 0.4629 - val_accuracy: 0.8283
```

```
Epoch 12/50
31/31 [==============================] - 1s 29ms/step - loss: 0.4686 - accuracy:
0.8306 - val_loss: 0.3914 - val_accuracy: 0.8434
Epoch 13/50
31/31 [==============================] - 1s 22ms/step - loss: 0.4498 - accuracy:
0.8222 - val_loss: 0.4089 - val_accuracy: 0.8464
Epoch 14/50
31/31 [==============================] - 1s 22ms/step - loss: 0.4128 - accuracy:
0.8493 - val_loss: 0.3758 - val_accuracy: 0.8645
Epoch 15/50
31/31 [==============================] - 1s 22ms/step - loss: 0.4439 - accuracy:
0.8389 - val_loss: 0.5861 - val_accuracy: 0.7319
Epoch 16/50
31/31 [==============================] - 1s 23ms/step - loss: 0.4831 - accuracy:
0.7952 - val_loss: 0.4036 - val_accuracy: 0.8494
Epoch 17/50
31/31 [==============================] - 1s 22ms/step - loss: 0.3971 - accuracy:
0.8565 - val_loss: 0.4125 - val_accuracy: 0.8223
Epoch 18/50
31/31 [==============================] - 1s 22ms/step - loss: 0.4107 - accuracy:
0.8326 - val_loss: 0.3645 - val_accuracy: 0.8464
Epoch 19/50
31/31 [==============================] - 1s 23ms/step - loss: 0.4011 - accuracy:
0.8378 - val_loss: 0.4157 - val_accuracy: 0.8193
Epoch 20/50
31/31 [==============================] - 1s 23ms/step - loss: 0.3538 - accuracy:
0.8617 - val_loss: 0.4030 - val_accuracy: 0.8313
Epoch 21/50
31/31 [==============================] - 1s 22ms/step - loss: 0.3569 - accuracy:
0.8524 - val_loss: 0.5603 - val_accuracy: 0.8042
Epoch 22/50
31/31 [==============================] - 1s 23ms/step - loss: 0.4241 - accuracy:
0.8254 - val_loss: 0.3247 - val_accuracy: 0.8855
Epoch 23/50
31/31 [==============================] - 1s 24ms/step - loss: 0.3368 - accuracy:
0.8753 - val_loss: 0.3486 - val_accuracy: 0.8584
Epoch 24/50
31/31 [==============================] - 1s 24ms/step - loss: 0.4066 - accuracy:
0.8430 - val_loss: 0.3557 - val_accuracy: 0.8464
Epoch 25/50
31/31 [==============================] - 1s 39ms/step - loss: 0.4360 - accuracy:
0.8306 - val_loss: 0.3811 - val_accuracy: 0.8584
Epoch 26/50
31/31 [==============================] - 1s 36ms/step - loss: 0.4114 - accuracy:
0.8337 - val_loss: 0.3576 - val_accuracy: 0.8524
Epoch 27/50
31/31 [==============================] - 1s 35ms/step - loss: 0.3383 - accuracy:
0.8701 - val_loss: 0.4380 - val_accuracy: 0.8253
```

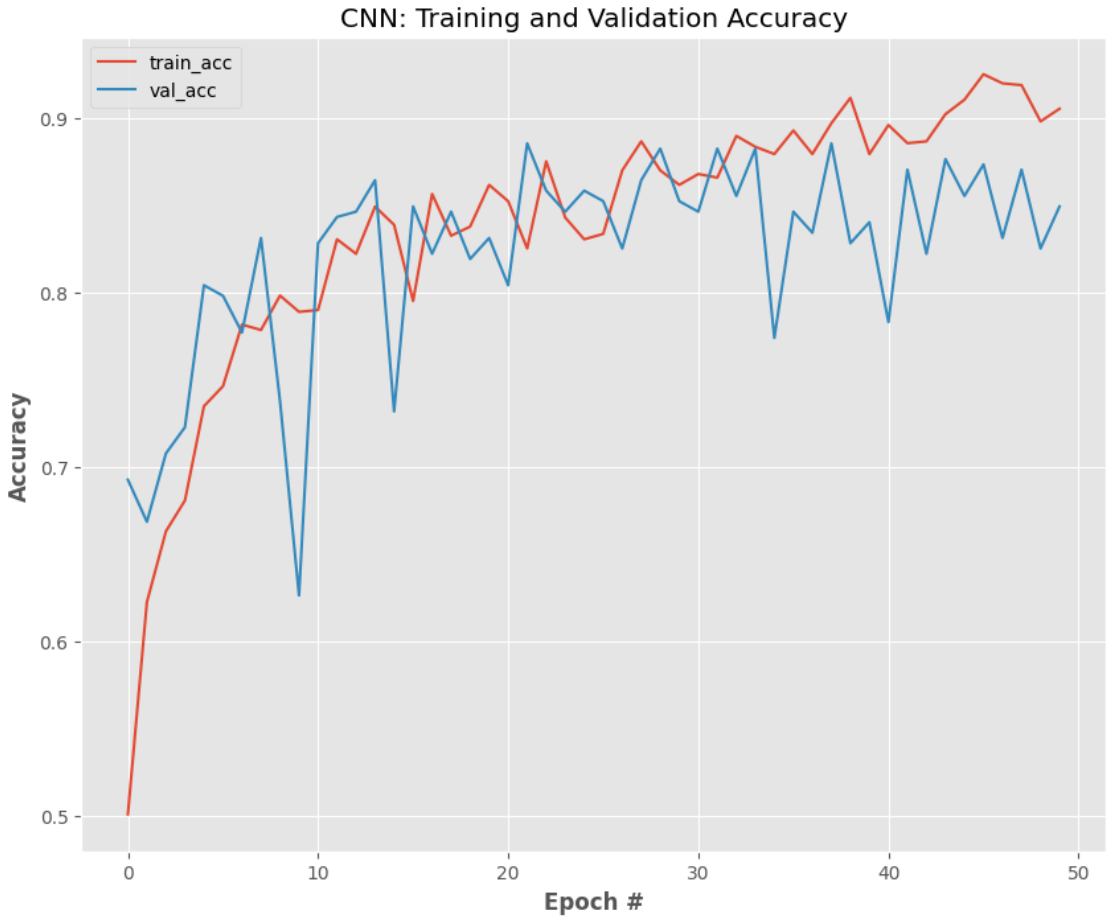
```
Epoch 28/50
31/31 [==============================] - 1s 22ms/step - loss: 0.3180 - accuracy:
0.8867 - val_loss: 0.3848 - val_accuracy: 0.8645
Epoch 29/50
31/31 [==============================] - 1s 24ms/step - loss: 0.3577 - accuracy:
0.8701 - val_loss: 0.3449 - val_accuracy: 0.8825
Epoch 30/50
31/31 [==============================] - 1s 24ms/step - loss: 0.3695 - accuracy:
0.8617 - val_loss: 0.3581 - val_accuracy: 0.8524
Epoch 31/50
31/31 [==============================] - 1s 23ms/step - loss: 0.3368 - accuracy:
0.8680 - val_loss: 0.4390 - val_accuracy: 0.8464
Epoch 32/50
31/31 [==============================] - 1s 22ms/step - loss: 0.3441 - accuracy:
0.8659 - val_loss: 0.3206 - val_accuracy: 0.8825
Epoch 33/50
31/31 [==============================] - 1s 22ms/step - loss: 0.2915 - accuracy:
0.8898 - val_loss: 0.3468 - val_accuracy: 0.8554
Epoch 34/50
31/31 [==============================] - 1s 23ms/step - loss: 0.2940 - accuracy:
0.8836 - val_loss: 0.3061 - val_accuracy: 0.8825
Epoch 35/50
31/31 [==============================] - 1s 22ms/step - loss: 0.2955 - accuracy:
0.8794 - val_loss: 0.4837 - val_accuracy: 0.7741
Epoch 36/50
31/31 [==============================] - 1s 22ms/step - loss: 0.2790 - accuracy:
0.8929 - val_loss: 0.3582 - val_accuracy: 0.8464
Epoch 37/50
31/31 [==============================] - 1s 21ms/step - loss: 0.2919 - accuracy:
0.8794 - val_loss: 0.3925 - val_accuracy: 0.8343
Epoch 38/50
31/31 [==============================] - 1s 23ms/step - loss: 0.2557 - accuracy:
0.8971 - val_loss: 0.2823 - val_accuracy: 0.8855
Epoch 39/50
31/31 [==============================] - 1s 36ms/step - loss: 0.2294 - accuracy:
0.9116 - val_loss: 0.4344 - val_accuracy: 0.8283
Epoch 40/50
31/31 [==============================] - 1s 36ms/step - loss: 0.2759 - accuracy:
0.8794 - val_loss: 0.4171 - val_accuracy: 0.8404
Epoch 41/50
31/31 [==============================] - 1s 24ms/step - loss: 0.2691 - accuracy:
0.8960 - val_loss: 0.4991 - val_accuracy: 0.7831
Epoch 42/50
31/31 [==============================] - 1s 23ms/step - loss: 0.2961 - accuracy:
0.8857 - val_loss: 0.3350 - val_accuracy: 0.8705
Epoch 43/50
31/31 [==============================] - 1s 22ms/step - loss: 0.2908 - accuracy:
0.8867 - val_loss: 0.3897 - val_accuracy: 0.8223
```

```
Epoch 44/50
31/31 [==============================] - 1s 23ms/step - loss: 0.2437 - accuracy:
0.9023 - val_loss: 0.3014 - val_accuracy: 0.8765
Epoch 45/50
31/31 [==============================] - 1s 22ms/step - loss: 0.2278 - accuracy:
0.9106 - val_loss: 0.4170 - val_accuracy: 0.8554
Epoch 46/50
31/31 [==============================] - 1s 22ms/step - loss: 0.2063 - accuracy:
0.9252 - val_loss: 0.3368 - val_accuracy: 0.8735
Epoch 47/50
31/31 [==============================] - 1s 23ms/step - loss: 0.2000 - accuracy:
0.9200 - val_loss: 0.4769 - val_accuracy: 0.8313
Epoch 48/50
31/31 [==============================] - 1s 23ms/step - loss: 0.1934 - accuracy:
0.9189 - val_loss: 0.3498 - val_accuracy: 0.8705
Epoch 49/50
31/31 [==============================] - 1s 22ms/step - loss: 0.2626 - accuracy:
0.8981 - val_loss: 0.4769 - val_accuracy: 0.8253
Epoch 50/50
31/31 [==============================] - 1s 22ms/step - loss: 0.2780 - accuracy:
0.9054 - val_loss: 0.4578 - val_accuracy: 0.8494
[INFO] serializing network …
```

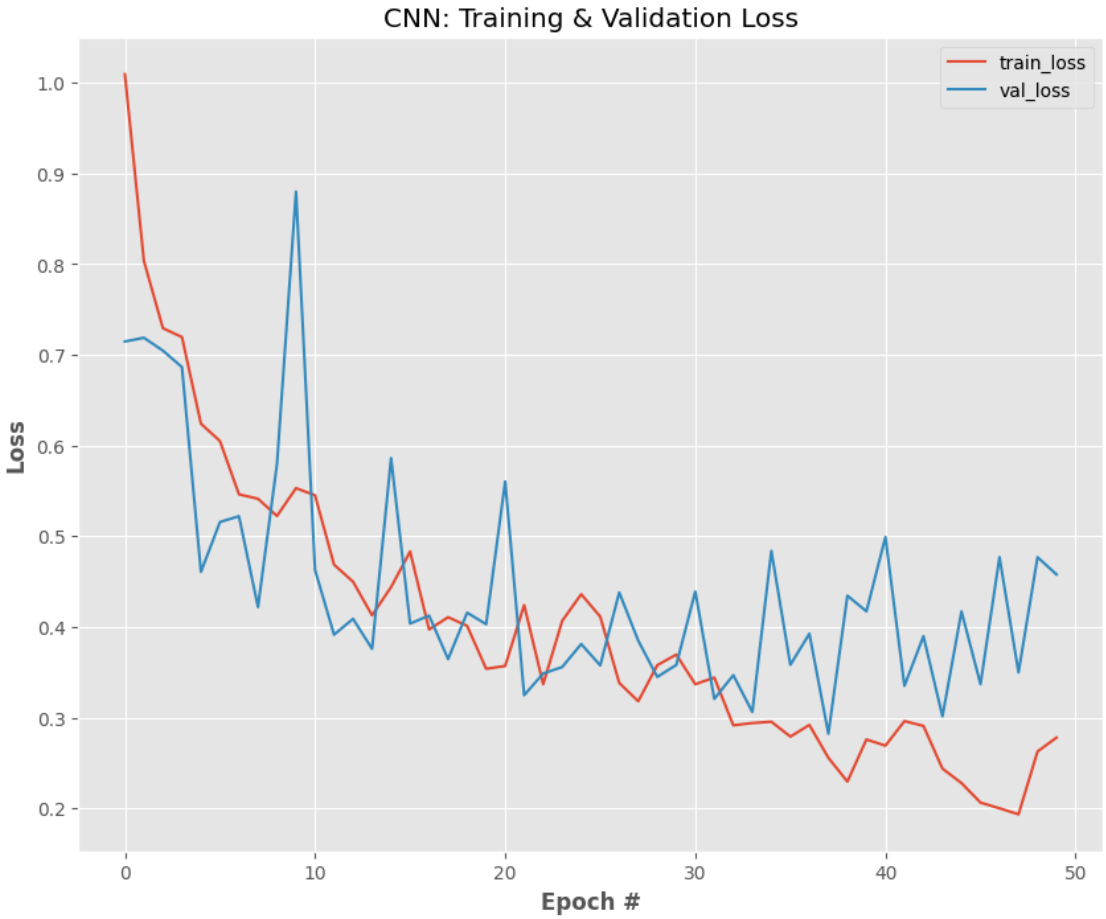
[ ]:

Step 8: Model Evaluation

[27]:
```
# plot the training and validation accuracy
N = np.arange(0, EPOCH )
plt.style.use("ggplot")
plt.figure(figsize = [10,8])
plt.plot(N, H.history["accuracy"], label="train_acc")
plt.plot(N, H.history["val_accuracy"], label="val_acc")
plt.title("CNN: Training and Validation Accuracy")
plt.xlabel("Epoch #", weight="bold")
plt.ylabel("Accuracy", weight="bold")
plt.legend()
plt.show()
```



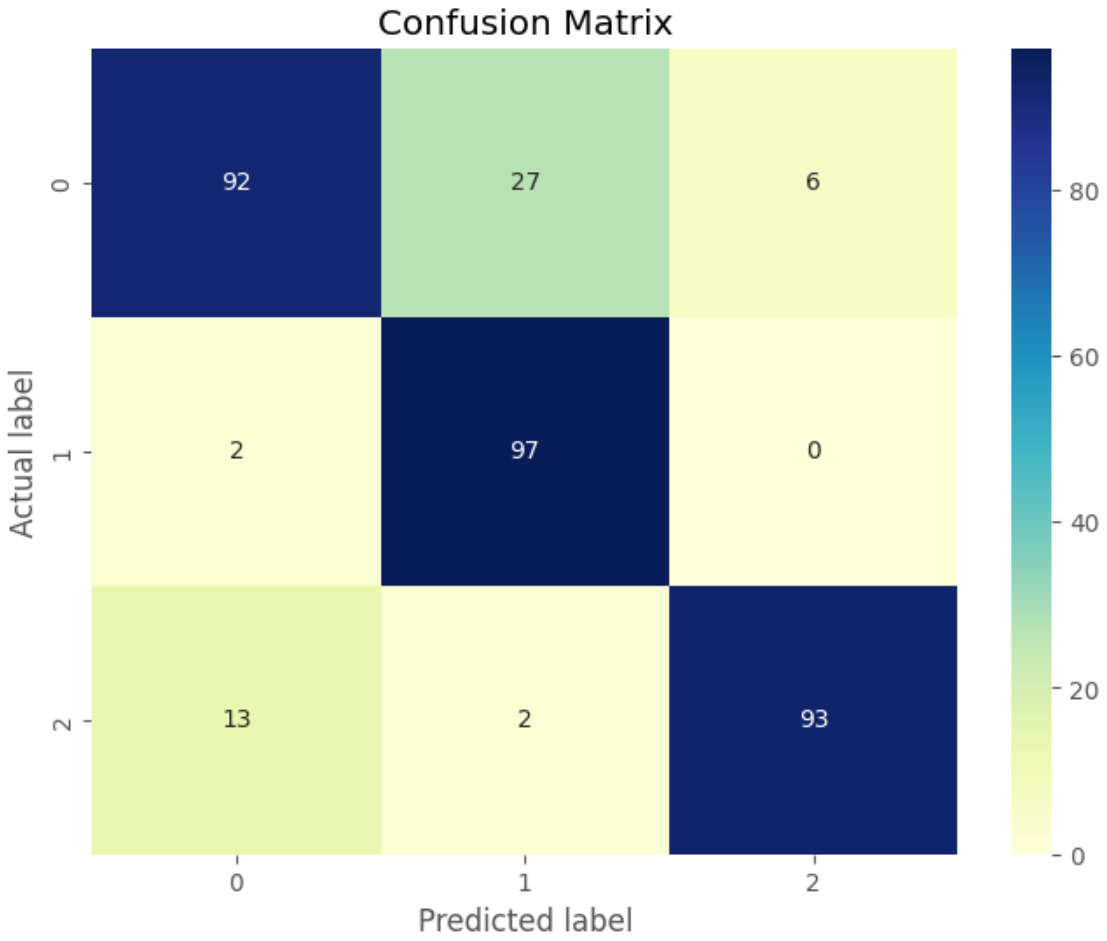
[28]:
```
# plot the training and validation loss
N = np.arange(0, EPOCH)
plt.style.use("ggplot")
plt.figure(figsize = [10,8])
plt.plot(N, H.history["loss"], label="train_loss")
plt.plot(N, H.history["val_loss"], label="val_loss")
plt.title("CNN: Training & Validation Loss")
plt.xlabel("Epoch #", weight="bold")
plt.ylabel("Loss", weight="bold")
plt.legend()
plt.show()
```



Confusion Matrix

[32]:
```
# Assuming you have defined y_test and y_pred earlier or from your model␣
↪predictions
# Example:
y_test = actual_classes
y_pred = predicted_classes

# Make the Confusion Matrix
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap="YlGnBu", fmt='g')
plt.title('Confusion Matrix')
```

```
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

# Calculate and print the Classification Report
from sklearn.metrics import classification_report

class_report = classification_report(y_test, y_pred)
print("\nClassification Report:")
print(class_report)
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.74      0.79       125
           1       0.77      0.98      0.86        99
           2       0.94      0.86      0.90       108
```

```
    accuracy                           0.85       332
   macro avg       0.86      0.86      0.85       332
weighted avg       0.86      0.85      0.85       332
```

Step 9: Prediction on Unseen Data

[33]:
```
def display_img(img):
    fig = plt.figure(figsize=(12,10))
    ax = fig.add_subplot(111)
    ax.imshow(img)
```

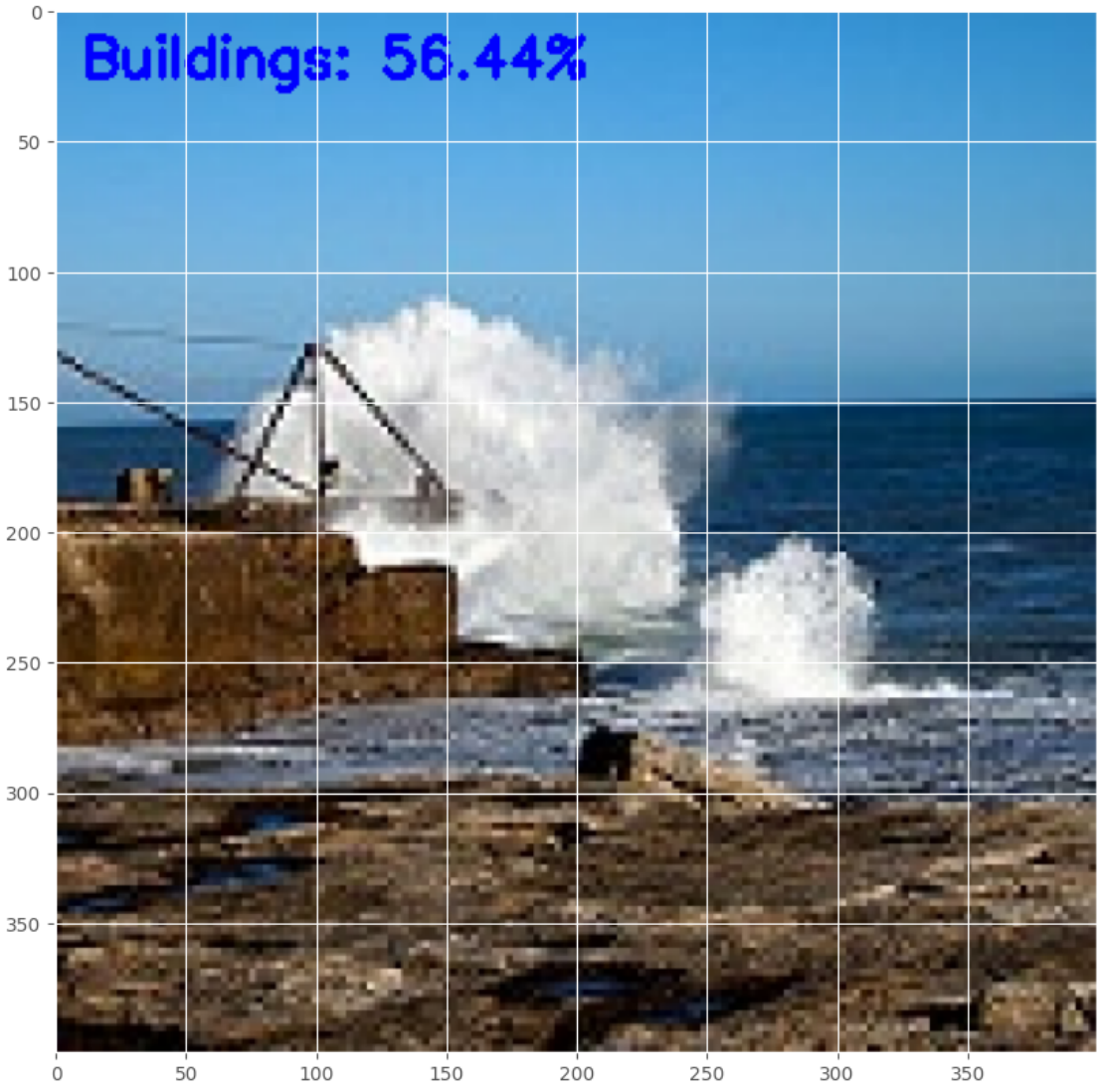
[ ]:
```
import cv2
import numpy as np
import imutils
from keras.preprocessing.image import img_to_array
from tqdm import tqdm

from imutils import paths  # Add this line to import paths from imutils



# grab the image paths and randomly shuffle theme

testImagePaths= sorted(list(paths.list_images('test_examples')))   # data␣
 ↪folder with 2 categorical folders

all_class= ["Buildings", "Forest", "Sea"]

#progressbar

with tqdm(total=len(testImagePaths)) as pbar:

    for imagePath in testImagePaths:

        # load the image
        image = cv2.imread(imagePath)
        orig = image.copy()

        # pre-process the image for classification
        image = cv2.resize(image, (28, 28))
        image = image.astype("float") / 255.0
        image = img_to_array(image)
        image = np.expand_dims(image, axis=0)

        # classify the input image
        prd_conf= model.predict(image)[0] # [[0.1 , 0.8 , 0.1]]
```

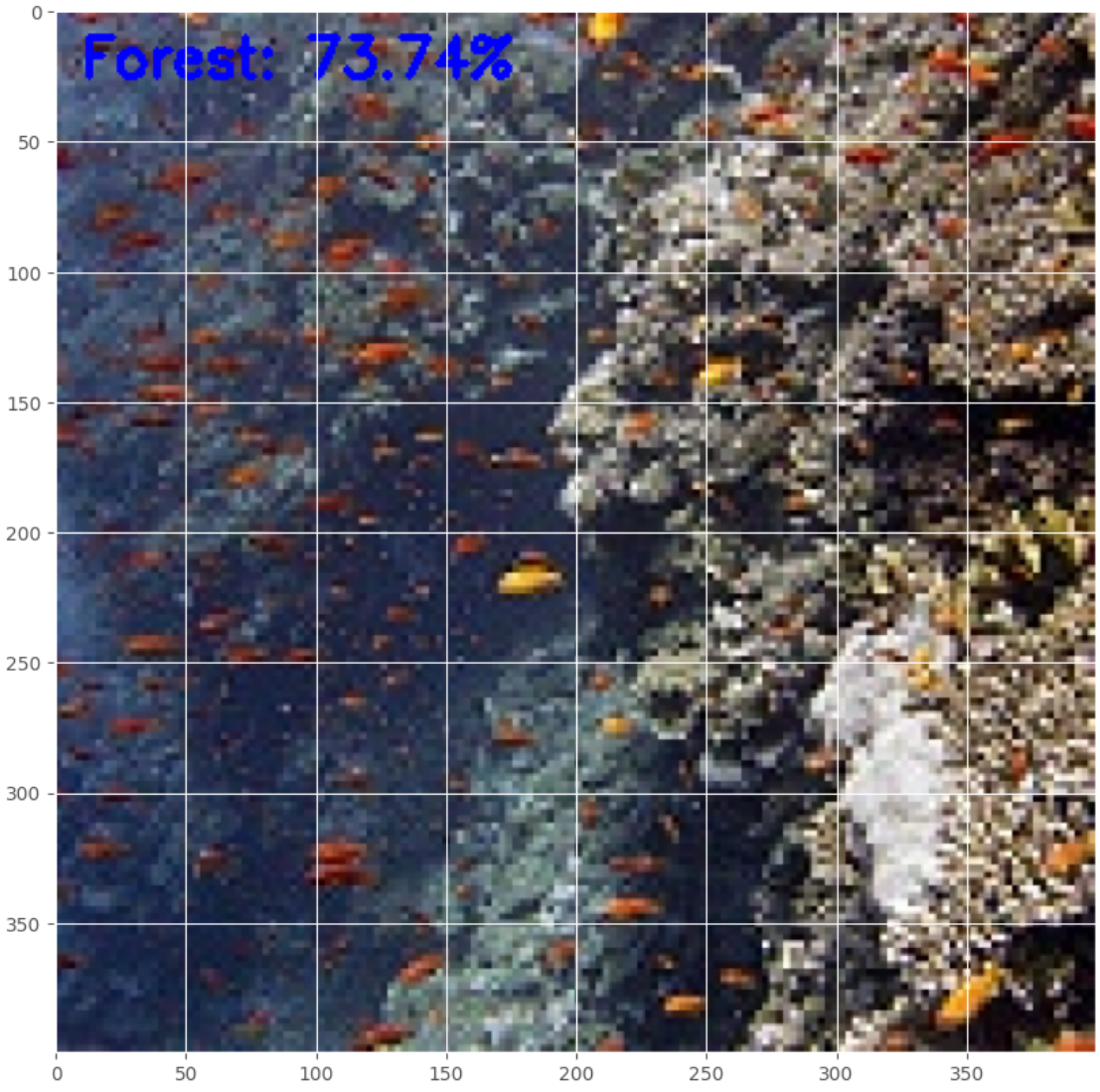
```
        print(prd_conf)

        # build the label
        label = all_class[np.argmax(prd_conf)] #[b f s]-> f
        proba = prd_conf[np.argmax(prd_conf)] # [0.1 , 0.8 , 0.1]  -> 0.8

        label = "{}: {:.2f}%".format(label, proba * 100)

        # draw the label on the image
        output = imutils.resize(orig, width=400)
        cv2.putText(output, label, (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,
            0.7, (255, 0, 0), 2)

        # convert img to rgb format and display in notebook
        img = cv2.cvtColor(output, cv2.COLOR_BGR2RGB)
        display_img(img)

        pbar.update(1)
```

```
  0%|          | 0/28 [00:00<?, ?it/s]
1/1 [==============================] - 0s 72ms/step
  4%|          | 1/28 [00:00<00:03,  6.95it/s]
[0.56444573 0.06998564 0.3655686 ]
1/1 [==============================] - 0s 17ms/step
[0.257111   0.73741686 0.00547208]
1/1 [==============================] - 0s 17ms/step
 11%|          | 3/28 [00:00<00:02,  9.83it/s]
[0.03927499 0.02638899 0.934336  ]
1/1 [==============================] - 0s 20ms/step
[0.00679145 0.16123104 0.83197755]
1/1 [==============================] - 0s 19ms/step
[0.18470265 0.00706683 0.80823046]
 18%|          | 5/28 [00:00<00:02, 10.50it/s]
1/1 [==============================] - 0s 18ms/step
[1.0017570e-01 1.6020221e-04 8.9966404e-01]
1/1 [==============================] - 0s 18ms/step
[0.01302703 0.01032175 0.97665125]
 25%|         | 7/28 [00:00<00:01, 10.60it/s]
1/1 [==============================] - 0s 17ms/step
[0.01056753 0.01082154 0.97861093]
1/1 [==============================] - 0s 16ms/step
[3.8310650e-04 3.8503404e-03 9.9576652e-01]
```

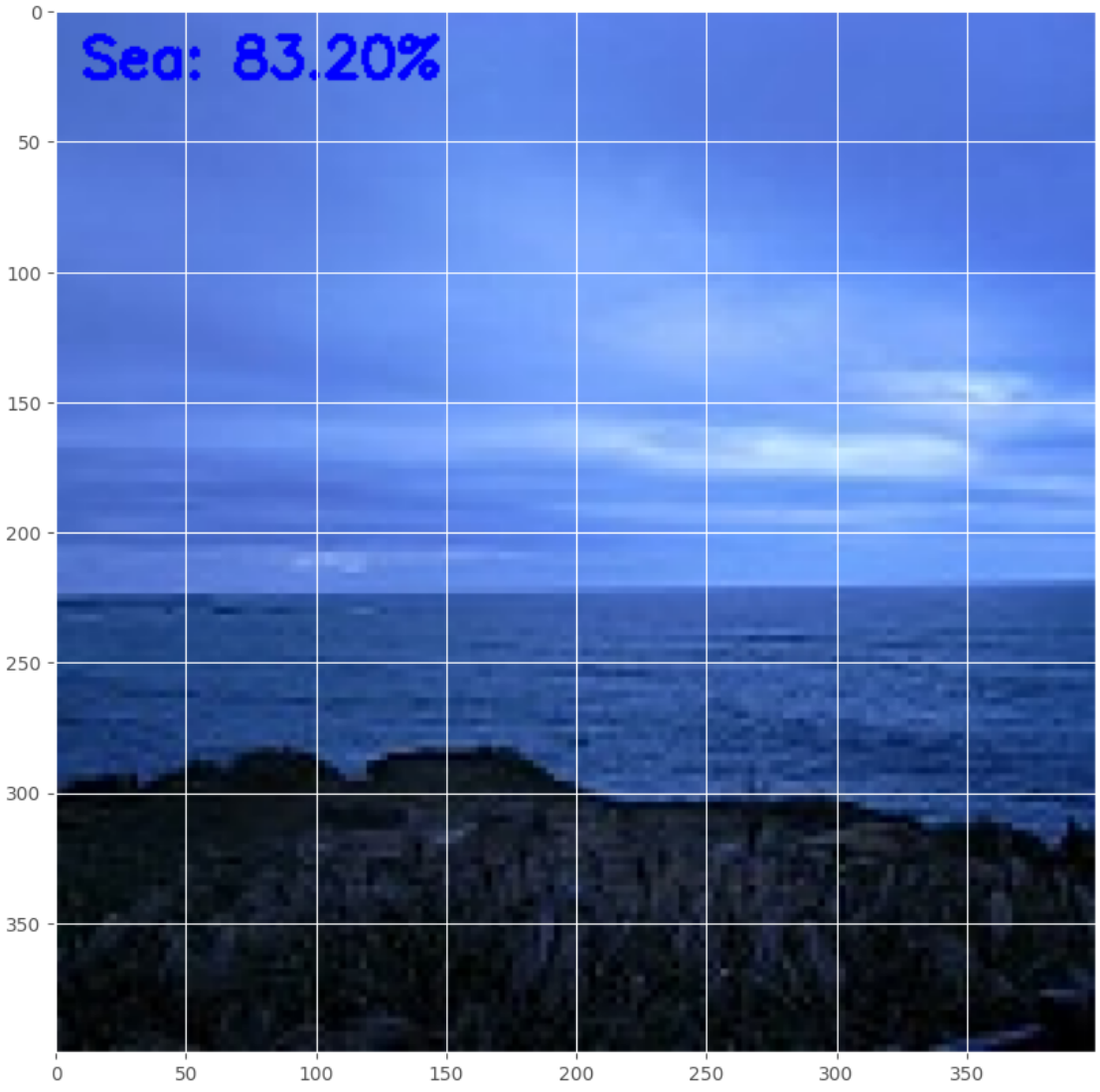
```
 32%|        | 9/28 [00:00<00:01, 10.79it/s]

1/1 [==============================] - 0s 17ms/step
[0.0747823  0.16433965 0.760878  ]
1/1 [==============================] - 0s 19ms/step
[0.87234414 0.00369883 0.12395702]

 39%|        | 11/28 [00:01<00:01, 10.92it/s]

1/1 [==============================] - 0s 20ms/step
[0.6086292  0.3869466  0.00442412]
1/1 [==============================] - 0s 18ms/step
[0.16493545 0.83091927 0.0041453 ]

 46%|        | 13/28 [00:01<00:01, 10.93it/s]

1/1 [==============================] - 0s 17ms/step
[0.9261783  0.03949699 0.03432473]
1/1 [==============================] - 0s 17ms/step
[4.3187261e-02 9.5680916e-01 3.5604155e-06]

 54%|       | 15/28 [00:01<00:01, 10.84it/s]

1/1 [==============================] - 0s 17ms/step
[9.9163437e-01 8.2040401e-03 1.6163773e-04]
1/1 [==============================] - 0s 17ms/step
[0.9842074  0.00380755 0.01198498]

 61%|       | 17/28 [00:01<00:01, 10.90it/s]

1/1 [==============================] - 0s 18ms/step
[0.36250576 0.0510759  0.5864184 ]
1/1 [==============================] - 0s 17ms/step

 68%|       | 19/28 [00:01<00:00, 11.06it/s]

[6.6102161e-03 9.9338949e-01 2.6704956e-07]
1/1 [==============================] - 0s 17ms/step
[3.2080643e-04 9.9967897e-01 2.1804702e-07]
1/1 [==============================] - 0s 18ms/step
[3.5435132e-05 9.9996448e-01 1.2230321e-07]

 75%|      | 21/28 [00:01<00:00, 11.03it/s]

1/1 [==============================] - 0s 17ms/step
[6.9879028e-03 9.9286872e-01 1.4334361e-04]
1/1 [==============================] - 0s 19ms/step
[1.4067056e-07 9.9999988e-01 5.6797650e-11]

 82%|      | 23/28 [00:02<00:00, 10.92it/s]

1/1 [==============================] - 0s 18ms/step
[7.2909740e-04 9.9925560e-01 1.5272024e-05]
1/1 [==============================] - 0s 18ms/step
[9.6631242e-04 9.9902046e-01 1.3164218e-05]
```

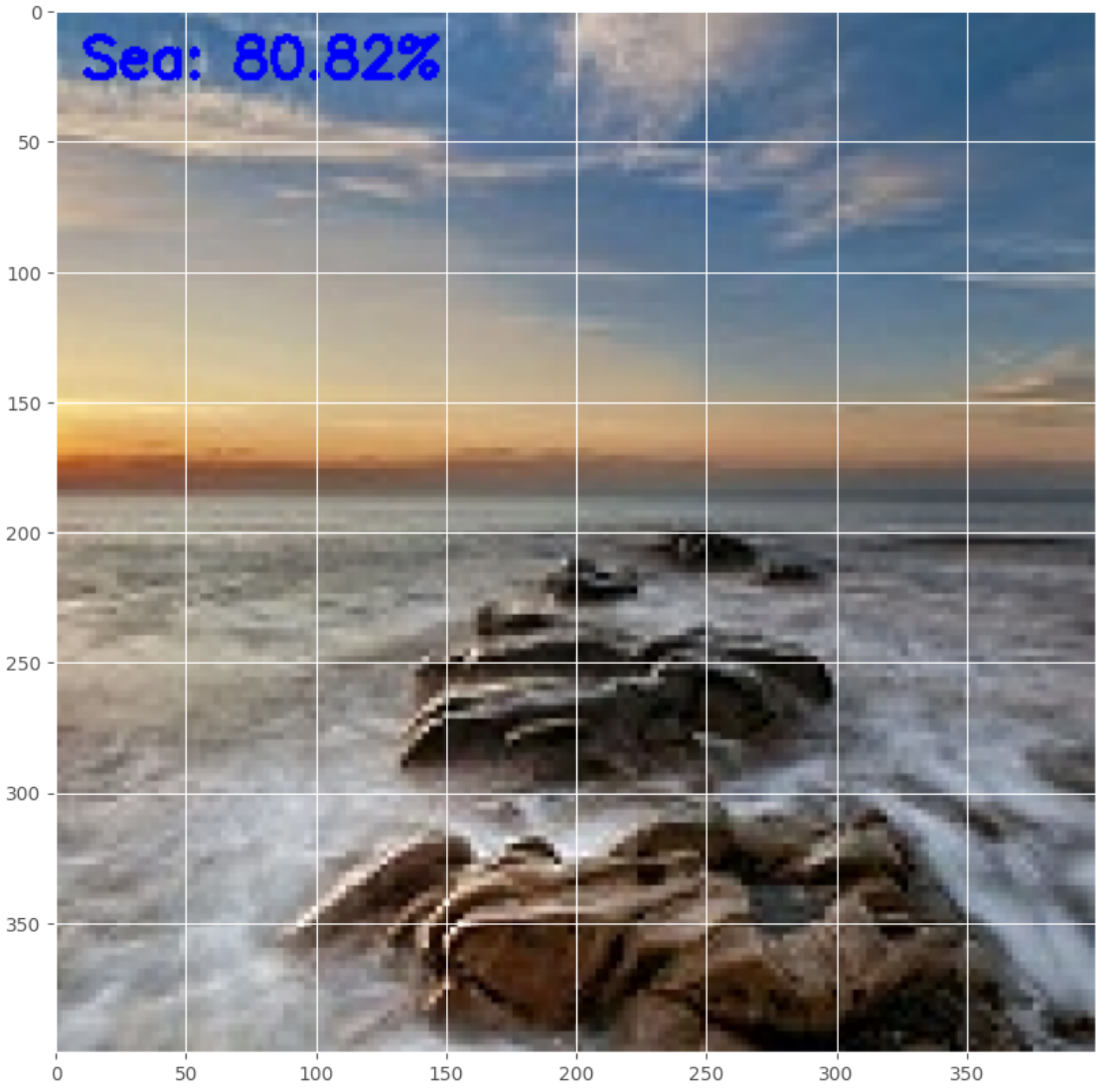
```
 89%|      | 25/28 [00:02<00:00, 10.77it/s]

1/1 [==============================] - 0s 18ms/step
[0.00464251 0.9942367  0.00112079]
1/1 [==============================] - 0s 23ms/step
[1.3744735e-06 9.9999857e-01 6.2528325e-12]

 96%|      | 27/28 [00:02<00:00, 10.74it/s]

1/1 [==============================] - 0s 18ms/step
[1.0953154e-05 9.9998903e-01 9.0804564e-10]

100%|      | 28/28 [00:02<00:00, 10.72it/s]
```

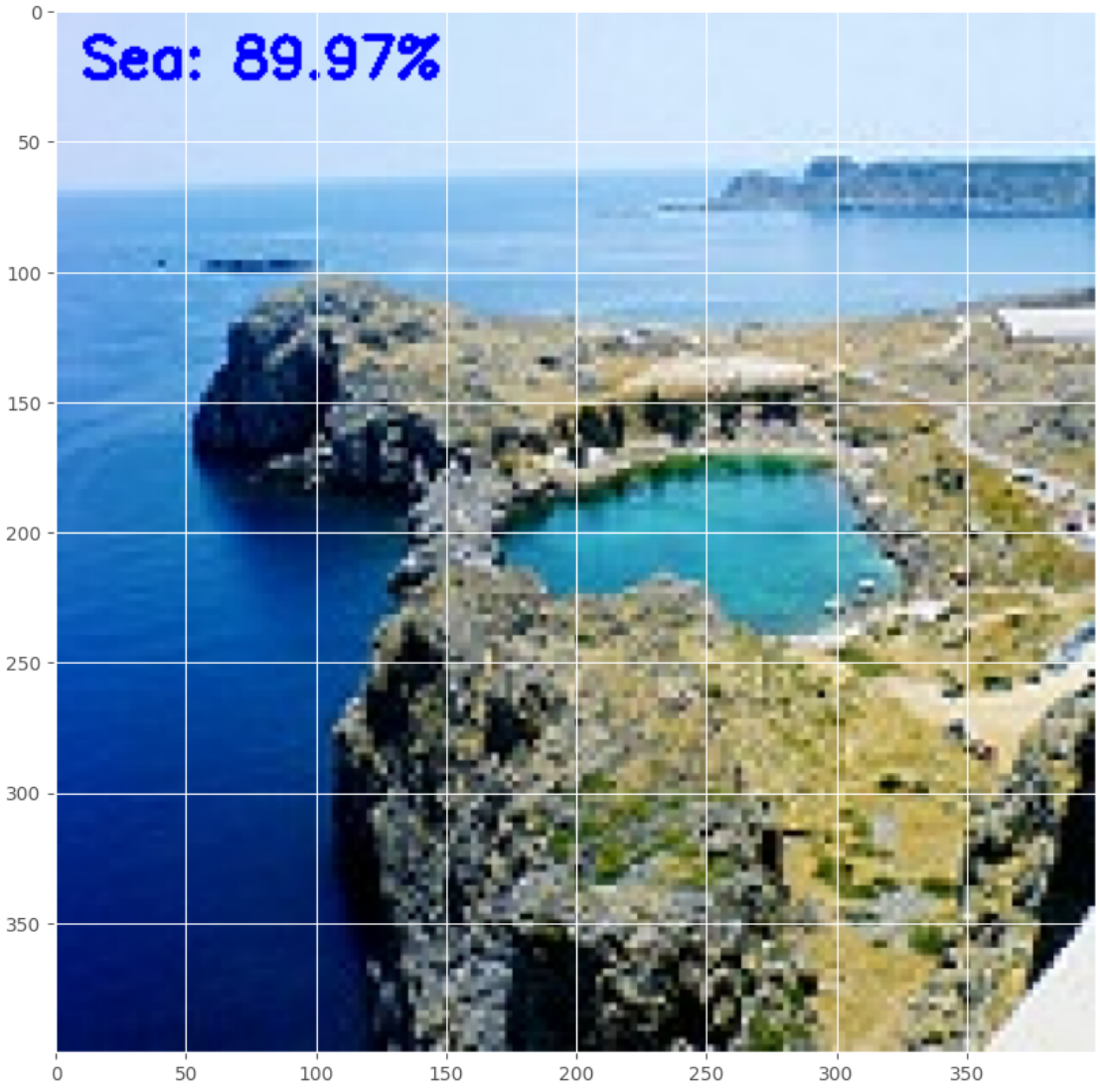


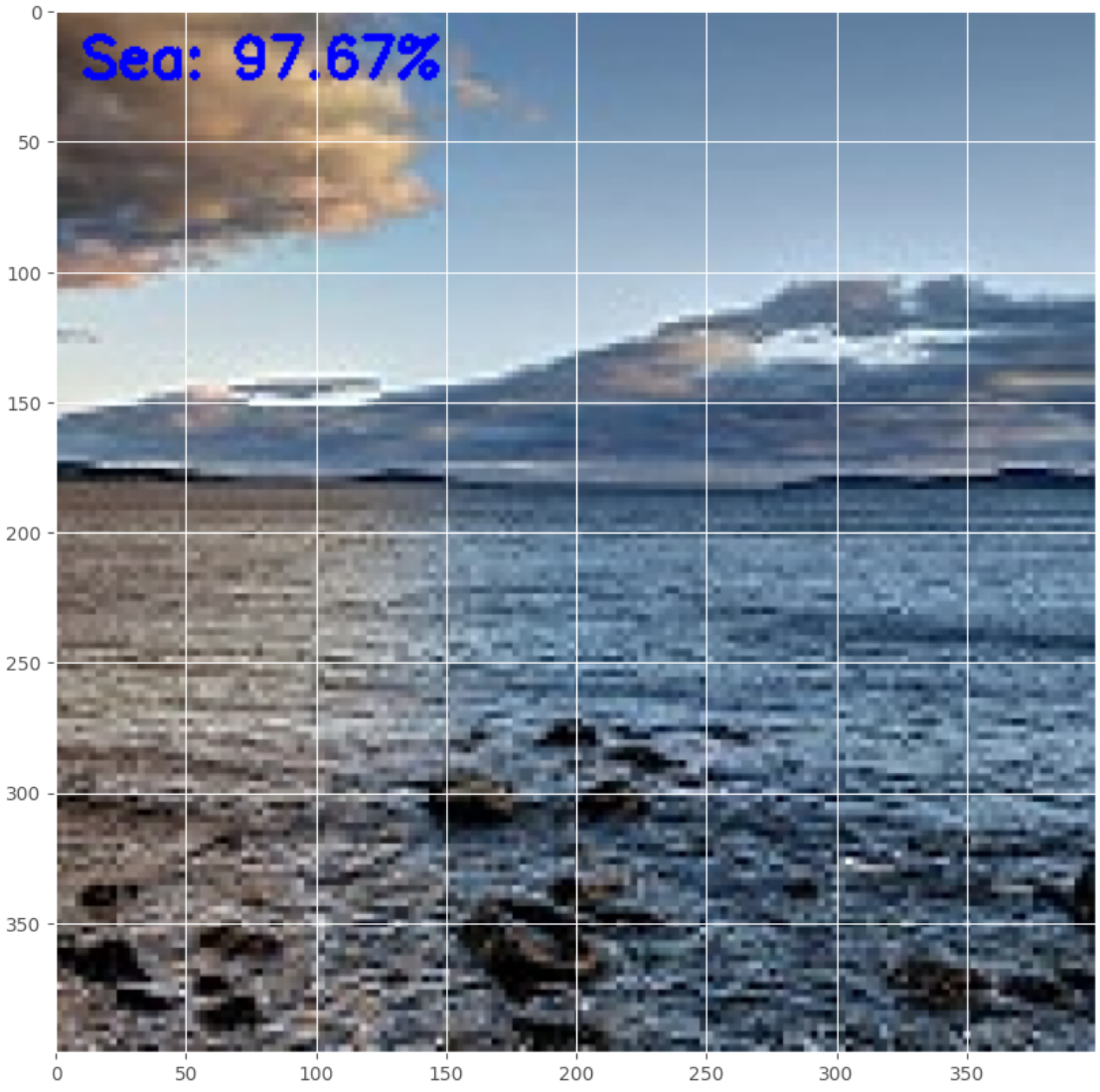

Forest: 73.74%
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350

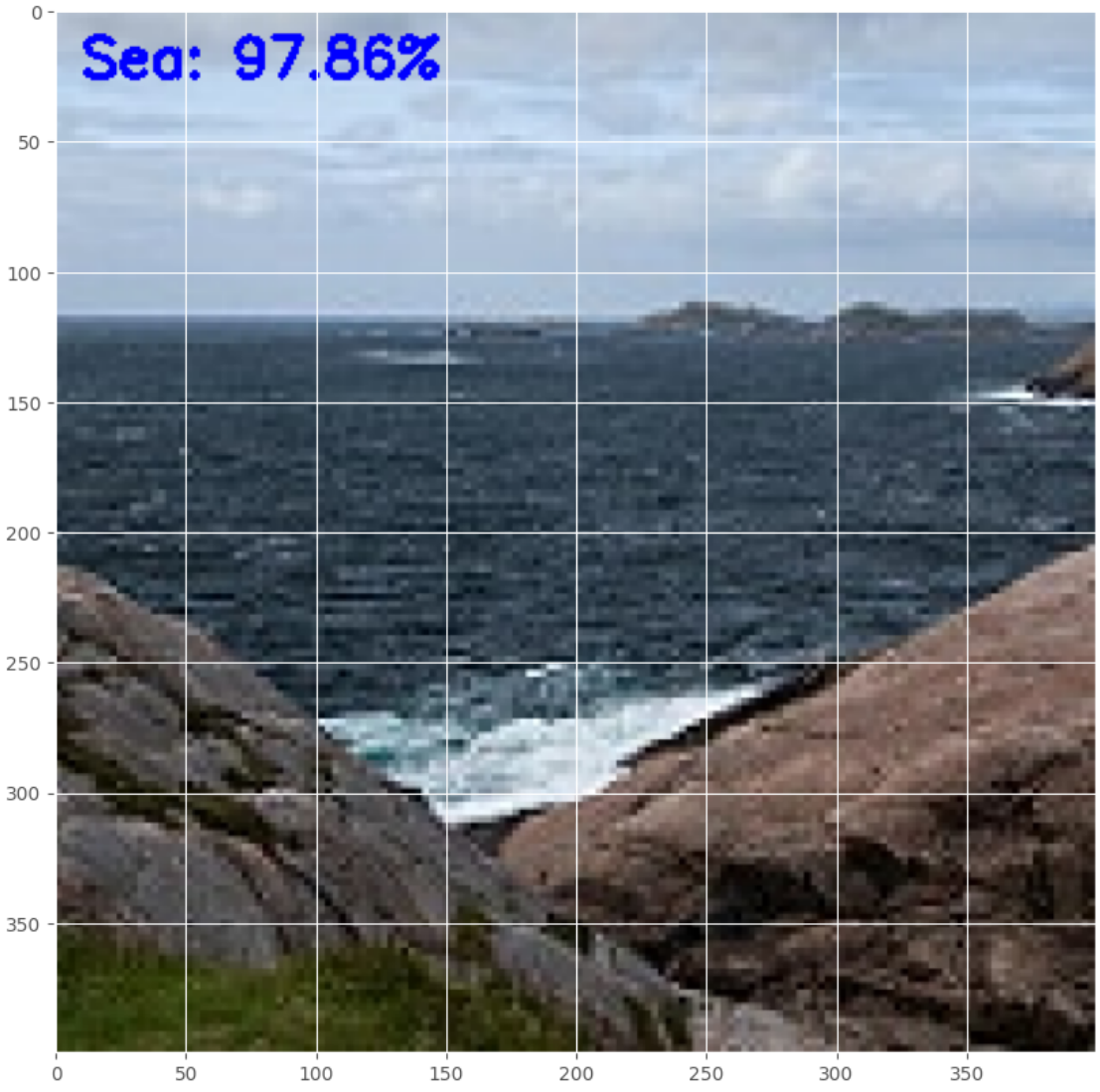

Sea: 93.43%
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350

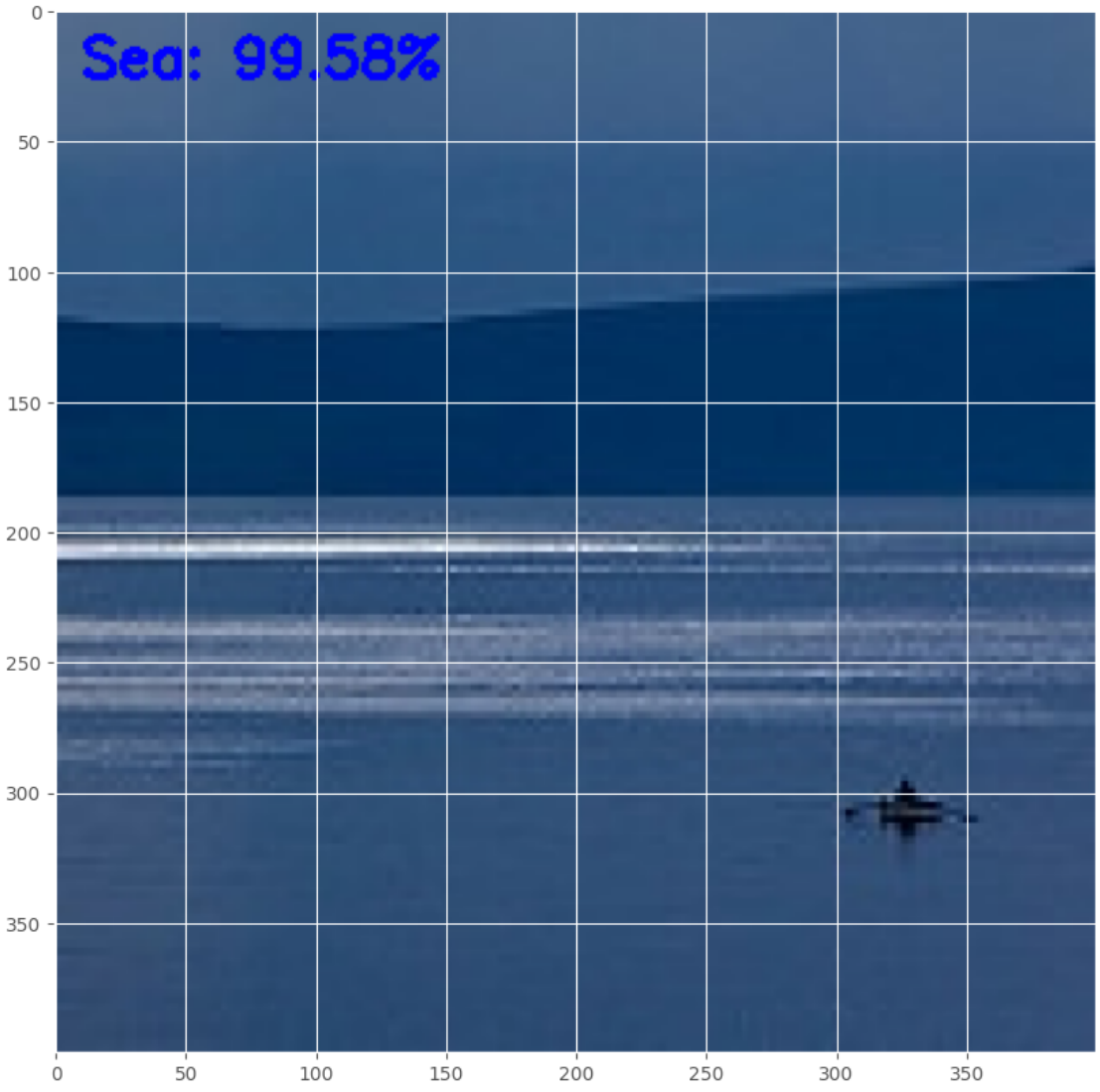

Sea: 83.20%
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350

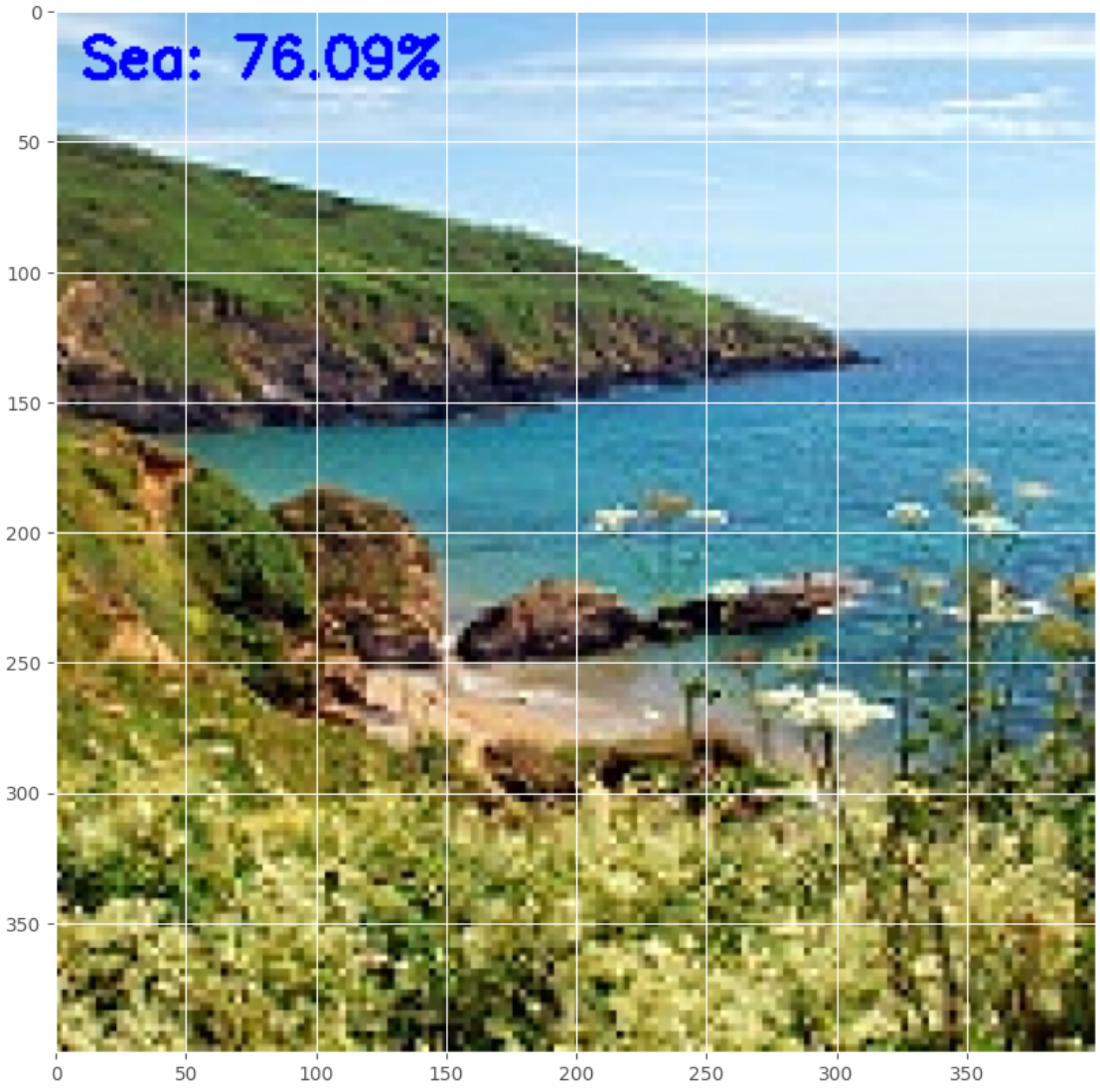

Sea: 80.82%
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350

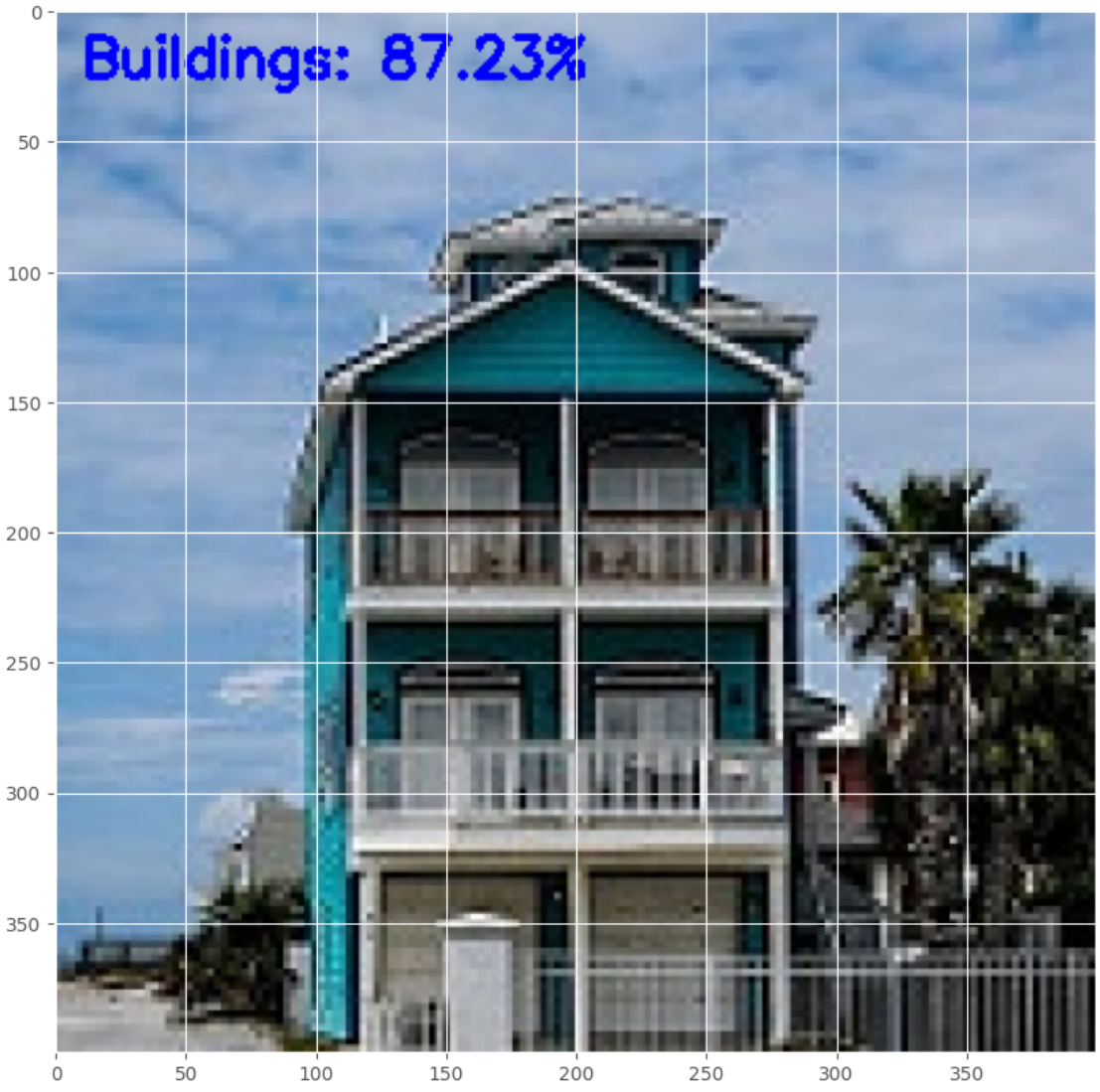

Sea: 89.97%
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350


Sea: 97.67%
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350


Sea: 97.86%
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350


Sea: 99.58%
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350


Sea: 76.09%
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350


Buildings: 87.23%
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350

[ ]:

Step 10: Deployment on Gradio

[35]:
```
import gradio as gr
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
import argparse
import imutils
import cv2
import matplotlib.pyplot as plt
from imutils import paths
```

```
%matplotlib inline
```

[36]:
```
from google.colab import drive
drive.mount('/content/drive')

import os
os.chdir("/content/drive/MyDrive/ Deep Learning Projects/ Image classification␣
 ↪using CNN")

# # load the model
print("[INFO] loading network and...")
model = load_model("cnn_model_{}.h5".format(EPOCH))
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
[INFO] loading network and…
```

[38]:
```
def predict_image(image):


    # pre-process the image for classification
    image = cv2.resize(image, (28, 28))
    image = image.astype("float") / 255.0
    image = img_to_array(image)
    image = np.expand_dims(image, axis=0)

    preds = model.predict(image)[0]
    result = dict()
    result["Buildings"] = round(float(list(preds)[0]), 3)
    result["Forest"] = round(float(list(preds)[1]), 3)
    result["Sea"] = round(float(list(preds)[2]), 3)

    print(result)

    return result
```

[41]:
```
!pip install gradio==3.50
```

```
Collecting gradio==3.50
  Downloading gradio-3.50.0-py3-none-any.whl (20.3 MB)
                           20.3/20.3 MB
23.9 MB/s eta 0:00:00
Requirement already satisfied: aiofiles<24.0,>=22.0 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (23.2.1)
Requirement already satisfied: altair<6.0,>=4.2.0 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (4.2.2)
Requirement already satisfied: fastapi in /usr/local/lib/python3.10/dist-
```

```
packages (from gradio==3.50) (0.110.2)
Requirement already satisfied: ffmpy in /usr/local/lib/python3.10/dist-packages
(from gradio==3.50) (0.3.2)
Collecting gradio-client==0.6.1 (from gradio==3.50)
  Downloading gradio_client-0.6.1-py3-none-any.whl (299 kB)
                           299.2/299.2
kB 28.1 MB/s eta 0:00:00
Requirement already satisfied: httpx in /usr/local/lib/python3.10/dist-
packages (from gradio==3.50) (0.27.0)
Requirement already satisfied: huggingface-hub>=0.14.0 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (0.20.3)
Requirement already satisfied: importlib-resources<7.0,>=1.3 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (6.4.0)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.10/dist-
packages (from gradio==3.50) (3.1.3)
Requirement already satisfied: markupsafe~=2.0 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (2.1.5)
Requirement already satisfied: matplotlib~=3.0 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (3.7.1)
Requirement already satisfied: numpy~=1.0 in /usr/local/lib/python3.10/dist-
packages (from gradio==3.50) (1.25.2)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.10/dist-
packages (from gradio==3.50) (3.10.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from gradio==3.50) (24.0)
Requirement already satisfied: pandas<3.0,>=1.0 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (2.0.3)
Requirement already satisfied: pillow<11.0,>=8.0 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (9.4.0)
Requirement already satisfied:
pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (2.7.0)
Requirement already satisfied: pydub in /usr/local/lib/python3.10/dist-packages
(from gradio==3.50) (0.25.1)
Requirement already satisfied: python-multipart in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (0.0.9)
Requirement already satisfied: pyyaml<7.0,>=5.0 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (6.0.1)
Requirement already satisfied: requests~=2.0 in /usr/local/lib/python3.10/dist-
packages (from gradio==3.50) (2.31.0)
Requirement already satisfied: semantic-version~=2.0 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (2.10.0)
Requirement already satisfied: typing-extensions~=4.0 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (4.11.0)
Requirement already satisfied: uvicorn>=0.14.0 in
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (0.29.0)
Requirement already satisfied: websockets<12.0,>=10.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from gradio==3.50) (11.0.3)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from gradio-client==0.6.1->gradio==3.50) (2023.6.0)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-
packages (from altair<6.0,>=4.2.0->gradio==3.50) (0.4)
Requirement already satisfied: jsonschema>=3.0 in
/usr/local/lib/python3.10/dist-packages (from altair<6.0,>=4.2.0->gradio==3.50)
(4.19.2)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages
(from altair<6.0,>=4.2.0->gradio==3.50) (0.12.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.14.0->gradio==3.50) (3.13.4)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub>=0.14.0->gradio==3.50) (4.66.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio==3.50)
(1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib~=3.0->gradio==3.50) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio==3.50)
(4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio==3.50)
(1.4.5)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio==3.50)
(3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio==3.50)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas<3.0,>=1.0->gradio==3.50) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
packages (from pandas<3.0,>=1.0->gradio==3.50) (2024.1)
Requirement already satisfied: annotated-types>=0.4.0 in
/usr/local/lib/python3.10/dist-packages (from
pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4->gradio==3.50) (0.6.0)
Requirement already satisfied: pydantic-core==2.18.1 in
/usr/local/lib/python3.10/dist-packages (from
pydantic!=1.8,!=1.8.1,!=2.0.0,!=2.0.1,<3.0.0,>=1.7.4->gradio==3.50) (2.18.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests~=2.0->gradio==3.50)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests~=2.0->gradio==3.50) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests~=2.0->gradio==3.50)
```

```
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests~=2.0->gradio==3.50)
(2024.2.2)
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.10/dist-
packages (from uvicorn>=0.14.0->gradio==3.50) (8.1.7)
Requirement already satisfied: h11>=0.8 in /usr/local/lib/python3.10/dist-
packages (from uvicorn>=0.14.0->gradio==3.50) (0.14.0)
Requirement already satisfied: starlette<0.38.0,>=0.37.2 in
/usr/local/lib/python3.10/dist-packages (from fastapi->gradio==3.50) (0.37.2)
Requirement already satisfied: anyio in /usr/local/lib/python3.10/dist-packages
(from httpx->gradio==3.50) (3.7.1)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-
packages (from httpx->gradio==3.50) (1.0.5)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-
packages (from httpx->gradio==3.50) (1.3.1)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio==3.50) (23.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
/usr/local/lib/python3.10/dist-packages (from
jsonschema>=3.0->altair<6.0,>=4.2.0->gradio==3.50) (2023.12.1)
Requirement already satisfied: referencing>=0.28.4 in
/usr/local/lib/python3.10/dist-packages (from
jsonschema>=3.0->altair<6.0,>=4.2.0->gradio==3.50) (0.34.0)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=3.0->altair<6.0,>=4.2.0->gradio==3.50) (0.18.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib~=3.0->gradio==3.50) (1.16.0)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-
packages (from anyio->httpx->gradio==3.50) (1.2.1)
Installing collected packages: gradio-client, gradio
  Attempting uninstall: gradio-client
    Found existing installation: gradio_client 0.16.0
    Uninstalling gradio_client-0.16.0:
      Successfully uninstalled gradio_client-0.16.0
  Attempting uninstall: gradio
    Found existing installation: gradio 4.28.3
    Uninstalling gradio-4.28.3:
      Successfully uninstalled gradio-4.28.3
Successfully installed gradio-3.50.0 gradio-client-0.6.1
```

[49]:
```
im = gr.Image()
label = gr.Label(num_top_classes=3)

gr.Interface(fn=predict_image, inputs=im, outputs=label, title="CNN Demo").
  ↪launch(share=True, debug=True)
```

```
Colab notebook detected. This cell will run indefinitely so that you can see
```

```
errors and logs. To turn off, set debug=False in launch().

Could not create share link. Missing file: /usr/local/lib/python3.10/dist-
packages/gradio/frpc_linux_amd64_v0.2.

Please check your internet connection. This can happen if your antivirus
software blocks the download of this file. You can install manually by following
these steps:

1. Download this file: https://cdn-media.huggingface.co/frpc-
gradio-0.2/frpc_linux_amd64
2. Rename the downloaded file to: frpc_linux_amd64_v0.2
3. Move the file to this location: /usr/local/lib/python3.10/dist-
packages/gradio

<IPython.core.display.Javascript object>

Keyboard interruption in main thread… closing server.
```

[49]:

[ ]:

[ ]:

[ ]:

[ ]: