In [3]:

```
```

```
---------------------------------------------------------------------
--
ModuleNotFoundError                       Traceback (most recent call las
t)
Cell In[3], line 3
      1 import os
      2 import numpy as np
----> 3 import tensorflow as tf
      4 from tensorflow.keras.layers import Conv2D, Add, Input, Lambda
      5 from tensorflow.keras.models import Model

ModuleNotFoundError: No module named 'tensorflow'
```

In [2]:

```
!pip install tensorflow
```

```
ta 0:00:01
     -------------------- ----------------- 30.7/57.5 kB 325.1 kB/s e
ta 0:00:01
     -------------------- ----------------- 30.7/57.5 kB 325.1 kB/s e
ta 0:00:01
     --------------------------------- ---- 51.2/57.5 kB 163.8 kB/s e
ta 0:00:01
     ---------------------------------------- 57.5/57.5 kB 177.6 kB/s e
ta 0:00:00
Collecting h5py>=2.9.0
  Downloading h5py-3.9.0-cp311-cp311-win_amd64.whl (2.7 MB)
     ---------------------------------------- 0.0/2.7 MB ? eta -:--:-
-
      --------------------------------------- 0.0/2.7 MB 1.9 MB/s eta
0:00:02
      --------------------------------------- 0.1/2.7 MB 812.7 kB/s e
ta 0:00:04
     - -------------------------------------- 0.1/2.7 MB 653.6 kB/s e
ta 0:00:04
     - -------------------------------------- 0.1/2.7 MB 653.6 kB/s e
```

In [3]:

```
import sys, os
import math
import tensorflow as tf
import numpy as np
import pandas as pd
import cv2
import matplotlib as mpl
import matplotlib.pyplot as plt
import skimage
```

In [4]:

```
def psnr(target, ref):
    # Assume target is RGB/BGR image
    target_data = target.astype(np.float32)
    ref_data = ref.astype(np.float32)

    diff = ref_data - target_data
    diff = diff.flatten('C')

    rmse = np.sqrt(np.mean(diff ** 2.))

    return 20 * np.log10(255. / rmse)
```

In [5]:

```
def mse(target, ref):
    target_data = target.astype(np.float32)
    ref_data = ref.astype(np.float32)
    err = np.sum((target_data - ref_data) ** 2)

    err /= float(target_data.shape[0] * target_data.shape[1])
    return err
```

In [6]:

```
from skimage.metrics import structural_similarity as ssim
```

In [7]:

```
def compare_images(target, ref):
    scores = []
    scores.append(psnr(target, ref))
    scores.append(mse(target, ref))
    scores.append(ssim(target, ref, multichannel=True))
    return scores
```

In [8]:

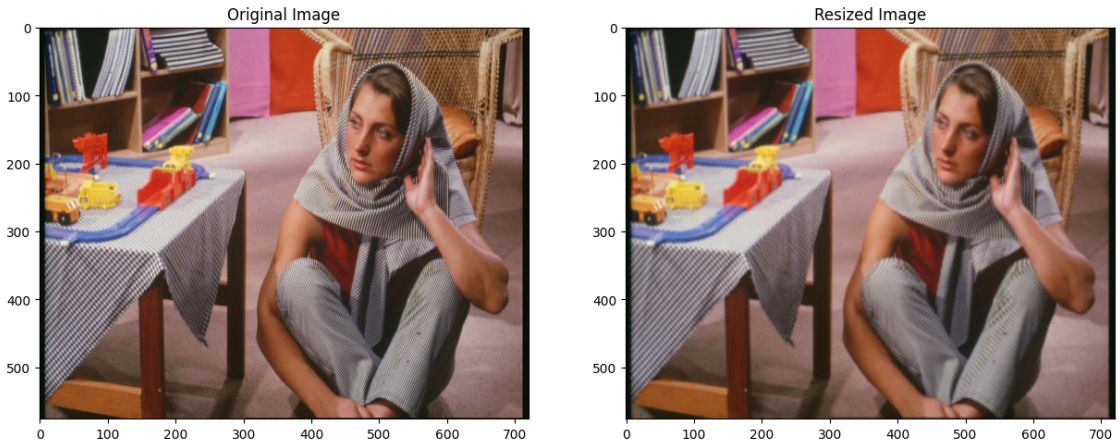
```
def prepare_images(path, factor):
    # Loop through the files in the directory
    for file in os.listdir(path):
        image = cv2.imread(path + '/' + file)

        # Find old and new image dimensions
        h, w, c = image.shape
        new_height = int(h / factor)
        new_width = int(w / factor)

        # Resize down the image
        image = cv2.resize(image, (new_width, new_height), interpolation=cv2.INTER_LINE

        # Resize up the image
        image = cv2.resize(image, (w, h), interpolation=cv2.INTER_LINEAR)

        # Save the image
        try:
            os.listdir(path + '/../../resized')
        except:
            os.mkdir(path + '/../../resized')

        cv2.imwrite(path + '/../../resized/{}'.format(file), image)
```

In [9]:

```
prepare_images (r'C:\Users\Shruti Ghoniya\Documents\SRCNN_dataset/Test/Set14', 2)
```

In [10]:

```
from PIL import Image
fig, ax = plt.subplots(1, 2, figsize=(15, 10))
ax[0].imshow(Image.open(r'C:\Users\Shruti Ghoniya\Documents\SRCNN_dataset/Test/Set14/ba
ax[0].title.set_text('Original Image')
ax[1].imshow(Image.open(r'C:\Users\Shruti Ghoniya\Documents\SRCNN_dataset/resized/barba
ax[1].title.set_text('Resized Image')
plt.show()
```



In [12]:

```
target_path = r'C:\Users\Shruti Ghoniya\Documents\SRCNN_dataset/Test/Set14/barbara.bmp'
ref_path = r'C:\Users\Shruti Ghoniya\Documents\SRCNN_dataset/resized/barbara.bmp'
target = cv2.imread(target_path)
ref = cv2.imread(ref_path)
print("Target image shape:", target.shape)
print("Reference image shape:", ref.shape)
```

```
Target image shape: (576, 720, 3)
Reference image shape: (576, 720, 3)
```

In [19]:

```
# Build train dataset
import h5py

names = sorted(os.listdir(r'C:\Users\Shruti Ghoniya\Documents\SRCNN_dataset\Train'))

data = []
label = []

for name in names:
    fpath = os.path.join(r'C:\Users\Shruti Ghoniya\Documents\SRCNN_dataset\Train', name
    hr_img = cv2.imread(fpath, cv2.IMREAD_COLOR)
    hr_img = cv2.cvtColor(hr_img, cv2.COLOR_BGR2YCrCb)
    hr_img = hr_img[:, :, 0]
    shape = hr_img.shape

    # resize operation to produce training data and labels
    shape = hr_img.shape
    lr_img = cv2.resize(hr_img, (int(shape[1] / 2), int(shape[0] / 2)))
    lr_img = cv2.resize(lr_img, (shape[1], shape[0]))

    width_range = int((shape[0] - 16 * 2) / 16)
    height_range = int((shape[1] - 16 * 2) / 16)

    for k in range(width_range):
        for j in range(height_range):
            x = k * 16
            y = j * 16

            hr_patch = hr_img[x: x + 32, y: y + 32]
            lr_patch = lr_img[x: x + 32, y: y + 32]

            hr_patch = hr_patch.astype(np.float32) / 255.
            lr_patch = lr_patch.astype(np.float32) / 255.

            hr = np.zeros((1, 20, 20), dtype=np.double)
            lr = np.zeros((1, 32, 32), dtype=np.double)

            hr[0, :, :] = hr_patch[6:-6, 6: -6]
            lr[0, :, :] = lr_patch

            label.append(hr)
            data.append(lr)

data = np.array(data, dtype=np.float32)
label = np.array(label, dtype=np.float32)
```

In [28]:

```
print("Image File Path:", fpath)
```

```
Image File Path: C:\Users\Shruti Ghoniya\Documents\SRCNN_dataset/Traint1.
bmp
```

In [20]:

```
with h5py.File('train.h5', 'w') as h:
    h.create_dataset('data', data=data, shape=data.shape)
    h.create_dataset('label', data=label, shape=label.shape)
```

In [23]:

```
names = sorted(os.listdir(r'C:\Users\Shruti Ghoniya\Documents\SRCNN_dataset\Test\Set14'
nums = len(names)

data_test = np.zeros((nums * 30, 1, 32, 32), dtype=np.double)
label_test = np.zeros((nums * 30, 1, 20, 20), dtype=np.double)

for i, name in enumerate(names):
    fpath = os.path.join(r'C:\Users\Shruti Ghoniya\Documents\SRCNN_dataset\Test\Set14',
    hr_img = cv2.imread(fpath, cv2.IMREAD_COLOR)
    hr_img = cv2.cvtColor(hr_img, cv2.COLOR_BGR2YCrCb)
    hr_img = hr_img[:, :, 0]
    shape = hr_img.shape

    # resize operation to produce training data and labels
    lr_img = cv2.resize(hr_img, (int(shape[1] / 2), int(shape[0] / 2)))
    lr_img = cv2.resize(lr_img, (shape[1], shape[0]))

    # Produce random crop
    x = np.random.randint(0, min(shape[0], shape[1]) - 32, 30)
    y = np.random.randint(0, min(shape[0], shape[1]) - 32, 30)

    for j in range(30):
        lr_patch = lr_img[x[j]:x[j] + 32, y[j]:y[j] + 32]
        hr_patch = hr_img[x[j]:x[j] + 32, y[j]:y[j] + 32]

        lr_patch = lr_patch.astype(np.float32) / 255.
        hr_patch = hr_patch.astype(np.float32) / 255.

        data_test[i * 30 + j, 0, :, :] = lr_patch
        label_test[i * 30 + j, 0, :, :] = hr_patch[6: -6, 6: -6]
```

In [24]:

```
with h5py.File('test.h5', 'w') as h:
    h.create_dataset('data', data=data_test, shape=data_test.shape)
    h.create_dataset('label', data=label_test, shape=label_test.shape)
```

In [25]:

```
def model():
    SRCNN = tf.keras.Sequential(name='SRCNN')
    SRCNN.add(tf.keras.layers.Conv2D(filters=128, kernel_size=(9, 9),
                                     padding='VALID',
                                     use_bias=True,
                                     input_shape=(None, None, 1),
                                     kernel_initializer='glorot_uniform',
                                     activation='relu'))
    SRCNN.add(tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3),
                                     padding='SAME',
                                     use_bias=True,
                                     kernel_initializer='glorot_uniform',
                                     activation='relu'))
    SRCNN.add(tf.keras.layers.Conv2D(filters=1, kernel_size=(5, 5),
                                     padding='VALID',
                                     use_bias=True,
                                     kernel_initializer='glorot_uniform',
                                     activation='linear'))
    # Optimizer
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.0003)

    # Compile model
    SRCNN.compile(optimizer=optimizer, loss='mean_squared_error', metrics=['mean_square

    return SRCNN
```

In [26]:

```
srcnn_model = model()
srcnn_model.summary()
```

```
Model: "SRCNN"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, None, None, 128)   10496

 conv2d_1 (Conv2D)           (None, None, None, 64)    73792

 conv2d_2 (Conv2D)           (None, None, None, 1)     1601

=================================================================
Total params: 85889 (335.50 KB)
Trainable params: 85889 (335.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [27]:

```
with h5py.File('./train.h5', 'r') as h:
    data = np.array(h.get('data'))
    label = np.array(h.get('label'))
    X_train = np.transpose(data, (0, 2, 3, 1))
    y_train = np.transpose(label, (0, 2, 3, 1))

with h5py.File('./test.h5', 'r') as h:
    data = np.array(h.get('data'))
    label = np.array(h.get('label'))
    X_test = np.transpose(data, (0, 2, 3, 1))
    y_test = np.transpose(label, (0, 2, 3, 1))

X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[27]:

```
((14901, 32, 32, 1), (14901, 20, 20, 1), (420, 32, 32, 1), (420, 20, 20,
1))
```

In [28]:

```
checkpoint_path = './srcnn/cp-{epoch:04d}.ckpt'
checkpoint_dir = os.path.dirname(checkpoint_path)
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_dir, save_best_only
                                                save_weights_only=True, verbose=0)
```

In [*]:

```
srcnn_model.fit(X_train, y_train, batch_size=64, validation_data=(X_test, y_test),
                callbacks=[checkpoint], shuffle=True, epochs=200, verbose=False)
```

In [ ]:

```

```