VLG Open Project

Report on

IMAGE DENOISING

Submitted by

Khushi Agarwal

21115074

EE, IV Year

June

TABLE OF CONTENTS

INRODUCTION

1.1 What is Image Denoising ?

The images that are captured in the real world come with noises. These noises can appear due to many reasons such as electric signal instabilities, malfunctioning of camera sensors, poor lighting conditions, errors in data transmission over long distances, etc. This can degrade the captured image's quality and can cause loss of information as the original pixel values are replaced by random values due to noise. So, there is a need to remove these noises from images when it comes to low-level vision tasks and image processing. The process of removing such noises from images is known as **Image Denoising**. Image denoising techniques aim to restore an image to its original quality by reducing or removing the noise, while preserving the important features of the image.

1.2 Deep Learning

A deep learning model is a neural network that is composed of multiple layers. The first layer is the input layer, which receives input data. The second layer is the hidden layer, which transforms the input data into a representation that can be used by the output layer. The output layer produces the desired output.

1.3 Performance Metrics

Peak Signal to Noise Ratio (PSNR) - The term peak signal-to-noise ratio (PSNR) is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise that affects the quality of its representation. Because many signals have a very wide dynamic range, (ratio between the largest and smallest possible values of a changeable quantity) the PSNR is usually expressed in terms of the logarithmic decibel scale. The higher the PSNR, the better the quality of the compressed image.

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE). \end{aligned}$$

2.1 Results from AutoEncoder

a) AutoEncoder Model Building

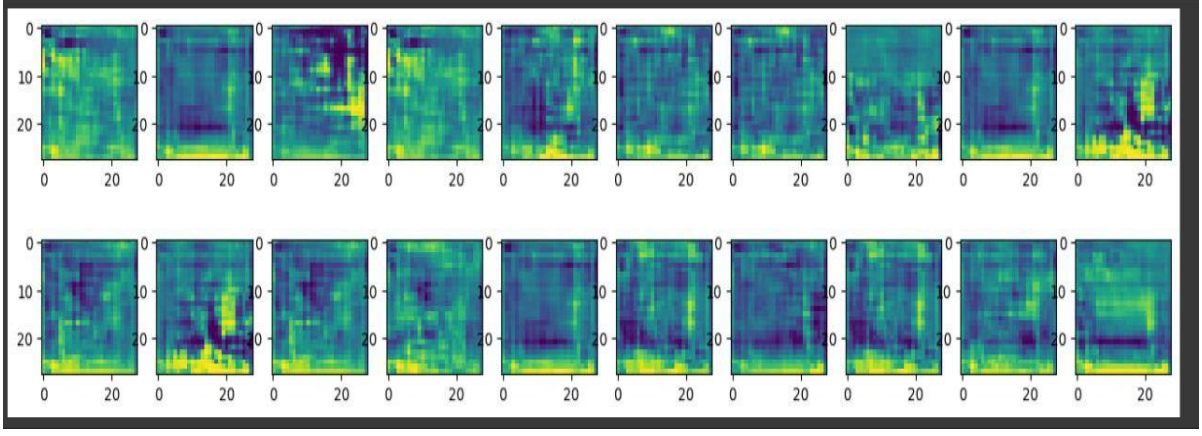
```
[234] from keras.models import Model
      from keras.layers import Input, Conv2D, MaxPool2D, Dense, UpSampling2D, BatchNormalization
      from keras.callbacks import ModelCheckpoint

      import tensorflow as tf

      encoder_input = Input(shape = train_dataset.shape[1:])
      x = Conv2D(32, (3,3), activation = 'relu', padding = 'same')(encoder_input)
      x = BatchNormalization()(x)
      x = MaxPool2D(pool_size = (2,2), padding = 'same')(x)
      x = Conv2D(32, (3,3), activation = 'relu', padding = 'same')(x)
      x = BatchNormalization()(x)
      encoded = MaxPool2D(pool_size = (2,2), padding = 'same')(x)
      x = Conv2D(32, (3,3), activation = 'relu', padding = 'same')(encoded)
      x = BatchNormalization()(x)
      x = UpSampling2D()(x)
      x = Conv2D(32, (3,3), activation = 'relu', padding = 'same')(x)
      x = BatchNormalization()(x)
      x = UpSampling2D()(x)
      decoded = Conv2D(1, (3,3), activation = 'sigmoid', padding = 'same')(x)

      autoencoder = Model(encoder_input, decoded, name = 'Denoising_Model')
      autoencoder.summary()
```

b) Model Fitting

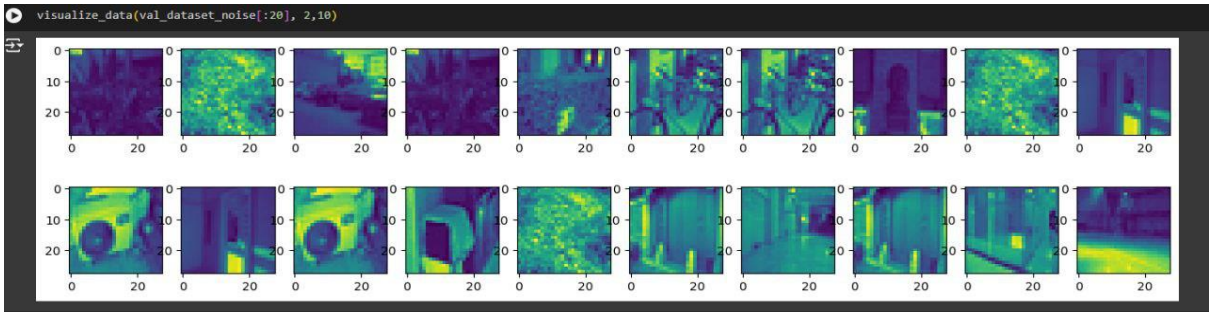
```
220] autoencoder.compile(loss = 'binary_crossentropy', optimizer = 'adam')

     checkpoint = ModelCheckpoint("denoising_model.h5", save_best_only=True, save_weights_only=False, verbose = 1)
     history = autoencoder.fit(train_dataset_noise, train_dataset, batch_size = 128, epochs = 50, callbacks = checkpoint, validation_split = 0.25, verbose = 2)
```

c) Predicted data

```
visualize_data(pred, 2,10)
```



d) Tesing data



e) Average PSNR value

```
import cv2
sum=0
for i in range(len(val_dataset_noise)):
    # print(PSNR(val_dataset_noise[i],pred[i]))
    sum+=PSNR(val_dataset_noise[i],pred[i])
print(sum/len(val_dataset_noise))
```

```
24.332416415973483
```

2.2 Zero DCE (Zero-Reference Deep Curve Estimation) –

Zero-Reference Deep Curve Estimation (Zero-DCE) formulates light enhancement as a task of image-specific curve estimation with a deep network. This method trains a lightweight deep network, DCE-Net, to estimate pixel-wise and high-order curves for dynamic range adjustment of a given image. The curve estimation is specially designed, considering pixel value range, monotonicity, and differentiability. Zero-DCE is appealing in its relaxed assumption on reference images, i.e., it does not require any paired or unpaired data during training. This is achieved through a set of carefully formulated non-reference loss functions, which implicitly measure the enhancement quality and drive the learning of the network. This method is efficient as image enhancement can be achieved by an intuitive and simple nonlinear curve mapping. A Deep Curve Estimation Network (DCE-Net) is devised to estimate a set of best-fitting Light-Enhancement curves (LE-curves) given an input image. The framework then maps all pixels of the input's RGB channels by applying the curves iteratively for obtaining the final enhanced image.

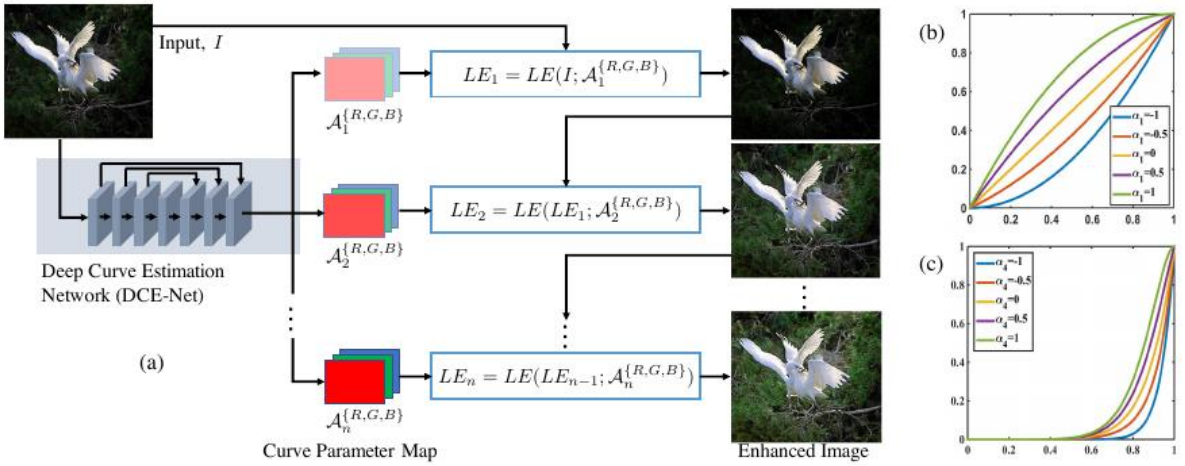
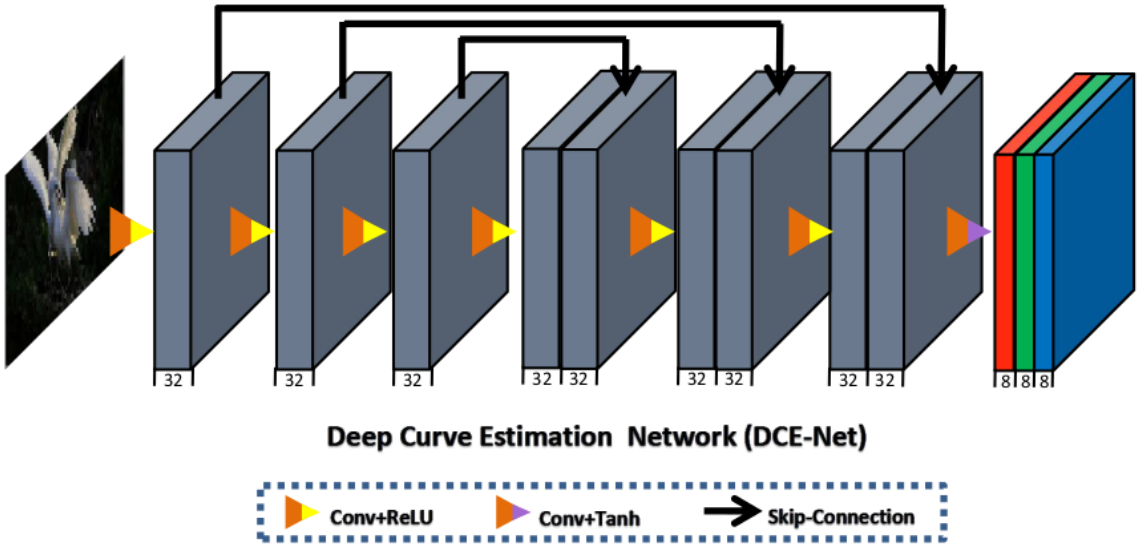


Fig. Zero DCE for low light image enhancement



a) Importing dataset-

```
from shutil import copy2

IMAGE_SIZE=256 #(256X256) size pixels
BATCH_SIZE=16 #no. of image processed during training
MAX_TRAIN_IMAGES=400 #no. of training images selected from model,rest images for testing

#importing image from drive folder
train_low_light_images=sorted(glob("/content/drive/MyDrive/Colab Notebooks/Train/low/*"))[:MAX_TRAIN_IMAGES]
val_low_light_images=sorted(glob("/content/drive/MyDrive/Colab Notebooks/Train/low/*"))[MAX_TRAIN_IMAGES:]

print("Train Images:", len(train_low_light_images))
print("Validation Images:", len(val_low_light_images))
```

```
Train Images: 400
Validation Images: 85
```

Necessary libraries have been imported like numpy, tensorflow, keras, pandas, matplotlib, os, etc. Dataset have been imported from google drive and used for training and testing.

b) Training and testing of data-

```
def load_data(image_path):
    img=tf.io.read_file(image_path)
    img=tf.image.decode_png(img,channels=3)
    img=tf.image.resize(images=img,size=[IMAGE_SIZE,IMAGE_SIZE])
    img=img/255.0
    return img


def data_generator(llimages):
    dataset=tf.data.Dataset.from_tensor_slices((llimages))
    dataset =dataset.map(load_data,num_parallel_calls=tf.data.AUTOTUNE)
    dataset=dataset.batch(BATCH_SIZE,drop_remainder=True)
    return dataset
```

```
[68]  training_dataset=data_generator(train_low_light_images)
      validation_dataset=data_generator(val_low_light_images)
```

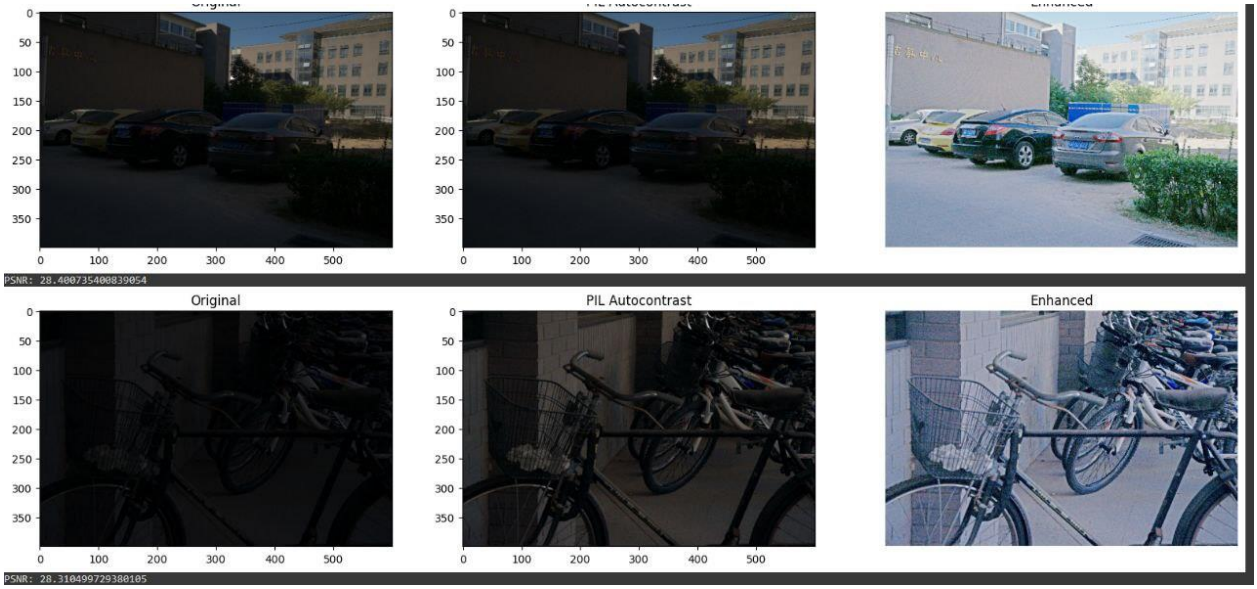
c) Building of model

```
zero_dce_model = ZeroDCE()
zero_dce_model.compile(learning_rate=1e-4)
history = zero_dce_model.fit(train_dataset, validation_data=val_dataset, epochs=100)


def plot_result(item):
    plt.plot(history.history[item], label=item)
    plt.plot(history.history["val_" + item], label="val_" + item)
    plt.xlabel("Epochs")
    plt.ylabel(item)
    plt.title("Train and Validation {} Over Epochs".format(item), fontsize=14)
    plt.legend()
    plt.grid()
    plt.show()

plot_result("total_loss")
plot_result("illumination_smoothness_loss")
plot_result("spatial_constancy_loss")
plot_result("color_constancy_loss")
plot_result("exposure_loss")
```

Various losses have been calculated like exposure loss, color constancy loss, illumination smoothness loss, spatial constancy loss.

d) PSNR value from DCE-Net Model-

PSNR value obtained is 28.40 from DCE-Net model which is improved from Autoencoder model.



REFERENCES

https://www.kaggle.com/code/soumikrakshit/tensorflow-zero-reference-deep-curve-estimation

https://medium.com/analytics-vidhya/image-denoising-using-deep-learning-dc2b19a3fd54

https://towardsai.net/p/deep-learning/image-de-noising-using-deep-learning

https://youtu.be/ggJhpgXK6E0?si=4fqO_5Kxp868xU8y

https://www.analyticsvidhya.com/blog/2021/06/autoencoders-a-gentle-introduction/