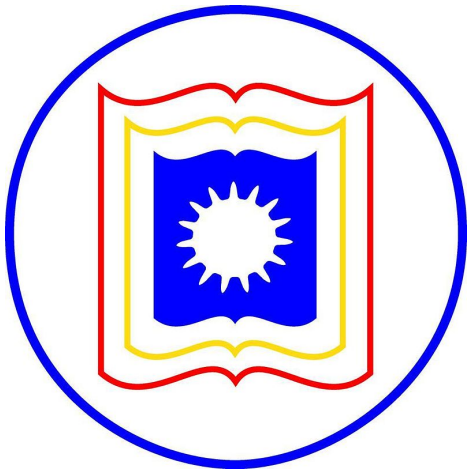
University of Rajshahi



Assignments of DIP Lab

Course Name: Digital Image Processing Lab
Course Code: CSE-4182
Date: September 12, 2022

Prepared by
Name: Md. Meem Mursalin Chowdhury
ID: 1810276103
Year: 4th, Semester: Odd
Session: 2017-18

Submitted to
Md. Khademul Islam Molla, Professor
MD. Rokanujjaman, Professor
Sangeeta Biswas, Associate Professor

Dept. of Computer Science and Engineering,
University of Rajshahi

Abstract

Digital Image Processing or DIP is a very well-known term to most of the people. It refers to the processing of different images. This processing is done images on basis of the need and demand. In this lab all the images are processed using python3 and the grayscale images or 2D images are considered.

Contents

1 Assignment-12

1.1 Introduction

Problem: Write a report on what different convolution layers of a CNN based image classifier see. You may use the attached code. You need to install Tensorflow in your computer. You can run it in Google Colab for GPU support if you do not have GPU or enough memory.

Solution: Here we have provide a report on what happens on different convolution layers of a CNN based image classifier. As the CNN model uses different layers, we can see different things happens to the images in different layers. The result and code of it given as follows.

1.2 Required Software

Here we have to install tensorflow which should be installed in a virtual environment. So at fist we have to create a virtual environment then install tensorflow. Again in the virtual environment, we have to install matplotlib, opencv and tk for generating the output. For the first execution of the program, the tensorflow model has to be loaded, so it will take some time to download. After that the program will execute fast.

1.3 Code

```
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
import cv2
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.models import Model

DIR = '/media/mursalin/New Volume/Image-Processing/code_13/'
# DIR = './'

def main():
  # Load a pre-trained model.
  baseModel = VGG16()
  baseModel.summary()

  # Prepare a new model having the desired layer as the output layer.
  for i in range(10):
    layer_number = i +1# *** Change layer number to see different convolution layer's output.
    print(layer_number)
    inputs = baseModel.input
    outputs = baseModel.layers[layer_number].output
    model = Model(inputs, outputs)

    # Prepare data.
    img = prepare_data()

    # Predict output of a specific layer.
    outputs = model.predict(img)

    # Display what different channls see.
    display_channels(outputs, layer_number)

def display_channels(chSet, layer_no):
  plt.figure(figsize = (20, 20))
  plt.suptitle('Layer-' + str(layer_no))
  for i in range(9):
    plt.subplot(3, 3, i + 1)
```

```
        plt.title('Channel-' + str(i))
        plt.imshow(chSet[0, :, :, i], cmap = 'gray')
        plt.axis('off')

      plt.savefig(DIR+'fig-'+str(layer_no)+'.jpg')
      plt.show()
      plt.close()

    def prepare_data():
      # Load an image
      imgPath = DIR + 'rose.jpg'  #'Elephant.jpg' #'Baby.jpeg' #'Rose.jpeg' #'Boat.jpeg' #
      bgrImg = cv2.imread(imgPath)
      print(bgrImg.shape)

      # Convert the image from BGR into RGB format
      rgbImg = cv2.cvtColor(bgrImg, cv2.COLOR_BGR2RGB)

      # Reshape the image so that it can fit into the model.
      #display_img(rgbImg)
      rgbImg = cv2.resize(rgbImg, (224, 224))
      display_img(rgbImg)

      # Expand dimension since the model accepts 4D data.
      print(rgbImg.shape)
      rgbImg = np.expand_dims(rgbImg, axis = 0)
      print(rgbImg.shape)

      # Preprocess image
      rgbImg = preprocess_input(rgbImg)

      return rgbImg

    def display_img(img):
      plt.imshow(img)
      plt.show()
      plt.close()

    if __name__ == '__main__':
      main()


```

Listing 1: Using pre trained model(VGG16)

1.4 Result Discussion

As we are using the VGG16 model as CNN model, there are 16 different layers are produced here. As different convolution layer are present there, so if we give input of an image, we will get different kinds of output for each kind of layers. For this experiment, we have taken output of 10 picture for each layer and found out the difference. The output picture of 10 different layers is given below. From the 1st layer, we can see that the input picture can be easily recognised. Here the edges of the object can be easily understood and here sobel filter is perhaps used. From layer-2 output picture, the picture continues to become blur and at the end of layer 10, the picture can be no more understood. After the finished layer output, the given picture can be no more recognised.

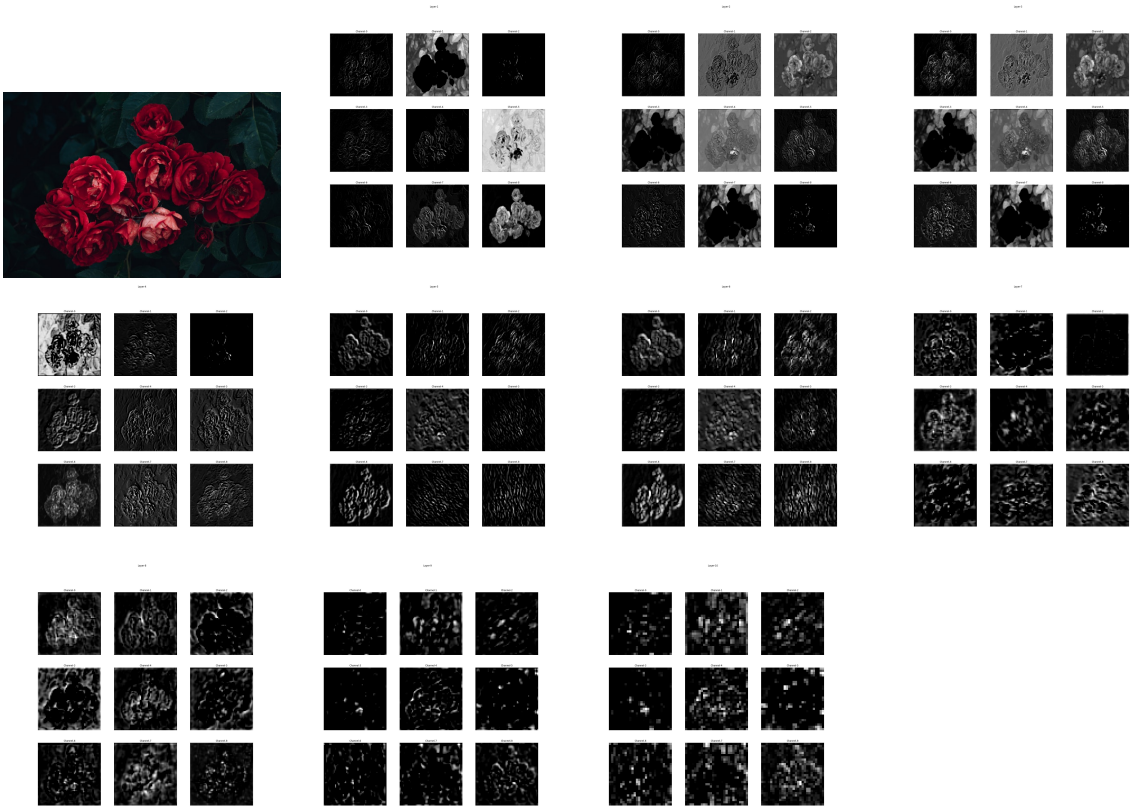


Figure 1: Input image, Output of Different Layers of VGG16 model from layer 1-10

2 Assignment-13

2.1 Introduction

Problem: Write a program to turn a JPEG image into a PNG image and vice-versa.

Solution: Here we have to load a jpeg typed image at first then we have to convert it to png and again we have to load a png image and convert it to jpeg format. At the time time of conversion we have to check the metadata of the header of the image and find the difference. So, we need two codes here for each of the conversion and showing the metadata of image header for better understanding. The codes, images and output are given below.

2.2 Required Software

Here we have to at first install Pillow for image conversion and also for getting the metadata of the image header file. We specifically import the Image from PIL and PIL.ExifTags for the metadat extraction. We can also use imghdr which is very much helpful inshowing the format of the image.

2.3 Code

```
1    from PIL import Image
2    from PIL.ExifTags import TAGS
```

```
    import imghdr

    def main():
        # path to the image or video
        img_path = './tower.jpg'
        print(img_path)
        print(imghdr.what(img_path))

        # read the image data using PIL
        image = Image.open(img_path)
        # print(image)

        # extract other basic metadata
        info_dict = {
            "Filename" : image.filename,
            "Image Size" : image.size,
            "Image Height" : image.height,
            "Image Width" : image.width,
            "Image Format" : image.format,
            "Image Mode" : image.mode,
            "Animated Image" : getattr(image, "is_animated", False),
            "Frames in Image" : getattr(image, "n_frames", 1)
        }

        # showing basic information
        for label, value in info_dict.items():
            print(f"{label:25}: {value}")

        # extract EXIF data
        exifdata = image.getexif()

        # iterating over all EXIF data fields
        for tag_id in exifdata:
            # get the tag name, instead of human unreadable tad id
            tag = TAGS.get(tag_id, tag_id)
            data = exifdata.get(tag_id)
            # decode bytes
            if isinstance(data, bytes):
                data = data.decode()
            print(f"{tag:25}: {data}")

        # jpg to png conversion
        image.save("tower.png")


    if __name__ == '__main__':
        main()
```

Listing 2: JPEG to PNG Image Conversion

```
./tower.jpg
jpeg
Filename                  : ./tower.jpg
Image Size                : (640, 480)
Image Height              : 480
Image Width               : 640
Image Format              : JPEG
Image Mode                : RGB
Animated Image            : False
Frames in Image           : 1
```

Listing 3: Output JPEG to PNG Image Conversion

```
from PIL import Image
from PIL.ExifTags import TAGS
```

```
    import imghdr

    def main():
        # path to the image or video
        img_path = './tower.png'
        print(img_path)
        print(imghdr.what(img_path))

        # read the image data using PIL
        image = Image.open(img_path)
        print(image)

        # extract other basic metadata
        info_dict = {
            "Filename" : image.filename,
            "Image Size" : image.size,
            "Image Height" : image.height,
            "Image Width" : image.width,
            "Image Format" : image.format,
            "Image Mode" : image.mode,
            "Animated Image" : getattr(image, "is_animated", False),
            "Frames in Image" : getattr(image, "n_frames", 1)
        }

        # showing basic information
        for label, value in info_dict.items():
            print(f"{label:25}: {value}")

        # extract EXIF data
        exifdata = image.getexif()

        # iterating over all EXIF data fields
        for tag_id in exifdata:
            # get the tag name, instead of human unreadable tad id
            tag = TAGS.get(tag_id, tag_id)
            data = exifdata.get(tag_id)
            # decode bytes
            if isinstance(data, bytes):
                data = data.decode()
            print(f"{tag:25}: {data}")

        # jpg to png conversion
        image2 = image.convert('RGB')
        image2.save("tower2.jpeg")

    if __name__ == '__main__':
        main()
```

Listing 4: PNG to JPEG Image Conversion

```
    ./tower.png
    png
    Filename                  : ./tower.png
    Image Size                : (640, 480)
    Image Height              : 480
    Image Width               : 640
    Image Format              : PNG
    Image Mode                : RGB
    Animated Image            : False
    Frames in Image           : 1
```

Listing 5: Output of PNG to JPEG Image Conversion

2.4 Result Discussion

Here in the output, we can see the the data is showing accordingly to the result. No inconsistency or error is observed. The input and output images are given below accordingly.

Figure 2: Input image of jpeg to png image conversion, Output image of jpeg to png image conversion and input image of png to jpeg image conversion, output image of png to jpeg image conversion