Project 4: Deep Learning

Introduction. This project will teach you how to implement a deep learning model for the reconstruction of a single-pixel camera image. ***This project is due Wednesday, November 1.***

Problem Description. The goal of this project is to implement image reconstruction for a single-pixel camera[1] using deep learning. The single-pixel camera image acquisition is simulated using the provided Matlab code. First, random natural images are taken from the STL-10 dataset. The were previously resized to be 64x64 pixels. Next, a set of binary masks are applied to each image. The masks were generated from a Russian-doll ordering of the Hadamard basis.[2] The data is compressed during acquisition. In the code provided, only the first 1/8th of the binary masks are used, resulting in an 8x compression ratio. Then, the Matlab code forms a low-quality guess of the image using the transpose of the forward model. See Figure 1 for a depiction of the steps taken by the Matlab software.

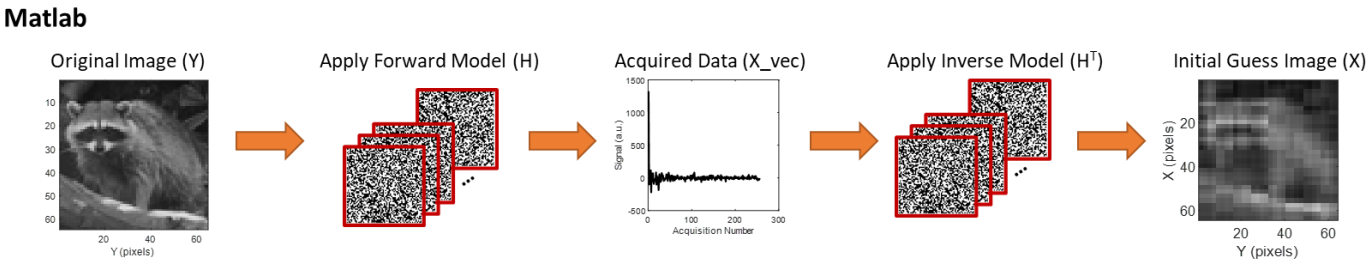

Figure 1: Flow of the Matlab software used to generate the data for deep learning

Your goal will be to create a deep neural network that takes the low-quality image, X, as an input and produces an estimate of the true image, Y.

Deep Learning Approach. You will use the Python programming language to implement your deep neural network. The industry standard is to use TensorFlow for deep learning (with PyTorch also seeing common use). However, it can be somewhat difficult to use directly. Therefore, it is typical to use an application programming interface (API) to interface with TensorFlow. Keras is a widely used interface that makes implementing deep learning much easier.

I suggest using Google Colab as a programming environment. It is a web-based implementation of Python with all the packages you need already installed. You can also make your network training run much faster by using a GPU provided by Google. This can be done by selecting "Change Runtime Type" from the "Runtime" menu.

In order to load the data into Google Colab you will have to move the .mat file generated Matlab code into your Google Drive folder. Then, you can give Colab access to your google drive using the following commands:

```
from google.colab import drive
drive.mount('/content/drive')
```

I have provided you with starter code in Appendix A that should get you to the point of loading your data. Make sure this works before proceeding with the deep network model generation.

Once your data is loaded in, you need to split it into Training, Validation, and Testing datasets. It is typical to split the overall data into 80%/10%/10% for each dataset, respectively.

Next, you need to define your deep neural network model architecture. I used a U-Net for my implementation, but feel free to try out any network you are interested in. I suggest finding existing code of a deep net instead of starting from scratch. Github can be a great source for this. Some key parameters that will drastically impact your performance are the type and size of each layer, the activation functions, and the loss function.

Once you have constructed your network, you need to train it. You will have to select a batch size and number of epochs. My network yielded pretty results with a batch size of 32 and with 10 epochs. I saw some incremental improvements if I increased the epochs to 100. But as you try to get it working, you should just use a small number of epochs for quick iterations on your parameters. Once you are done fiddling with your model, you want to train until the loss function of the plateaus on the Validation dataset.

Finally, you need to make predictions on the Testing dataset. Run the testing dataset through your model using the .predict() function.

Deliverables.

1) Describe your network in detail. Which layers did you use? Which activation functions? How many parameters did you train? What was your loss function?
2) Report the results of training. How many epochs did you train for? What was the batch size? What were the final Training and Validation loss values?
3) Present the results of your network on the Testing dataset. How close was the prediction result to the ground truth (this could be in terms of MSE or SSIM or other relevant metric). Include representative images of the results.
4) Finally, comment on the outcome and describe how you might improve your model in the future.
5) Include your well-commented code in your submission.

References Cited

1 Duarte, M. F. *et al.* Single-pixel imaging via compressive sampling. *IEEE signal processing magazine* **25**, 83-91 (2008).

2 Sun, M.-J., Meng, L.-T., Edgar, M. P., Padgett, M. J. & Radwell, N. A Russian Dolls ordering of the Hadamard basis for compressive single-pixel imaging. *Scientific reports* **7**, 3464 (2017).

Appendix A: Starting Python Code

```
# Import Statements
import scipy as sp
import numpy as np
import scipy.io as spio
import tensorflow as tf
import matplotlib.pyplot as plt
import time
import math


from keras import backend as K
from random import randint, uniform, gauss
from math import sqrt
from tqdm import tqdm_notebook, tqdm, tnrange
from pathlib import Path
from pylab import figure, cm
from tensorflow import keras
from matplotlib.colors import LogNorm

# Checking the system is up to date and GPU is enabled
import sys
print(sys.executable)
print(tf.__version__)
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())

from google.colab import drive
drive.mount('/content/drive')
```

```
mat_dict = sp.io.loadmat("/content/drive/MyDrive/Teaching/ENGS
117/2023/Projects/Project 4/Project4_Data.mat")#Replace this with your particular
file location
X = mat_dict["X"]
Y = mat_dict["Y"]
#H = mat_dict["H"]

print('X shape is ', X.shape)
print('Y shape is ', Y.shape)
#print('H shape is ', H.shape)
```

```
imageIndex = 99

im1 = plt.matshow(Y[imageIndex,:,:], cmap=cm.gray, vmin = 0.0, vmax = 1)
cbar1 = plt.colorbar(im1)
plt.gca().xaxis.tick_bottom()
plt.title(r'Ground Truth Image')

im1 = plt.matshow(X[imageIndex,:,:], cmap=cm.gray, vmin = 0.0, vmax = 1)
cbar1 = plt.colorbar(im1)
plt.gca().xaxis.tick_bottom()
plt.title(r'Initial Guess')
```