

Deep Learning

EXERCISE 1 - DATA READERS

1 Introduction

The first and most tedious part of training a Deep Neural Network is creating the data flow. In this exercise you will create your data reader, which you may later use for the course project. We will be working with Tensorflow version 2.0.0 and we recommend working with Google Colaboratory. Please fill in the required parts. For any questions you can email: arbellea@post.bgu.ac.il

2 Setup

2.1 Tensorflow Version

Install Tensorflow version 2.0.0: `pip install tensorflow==2.0.0` Import tensorflow and check version:

```
»import matplotlib.pyplot as plt »import os »import tensorflow as tf »print(tf.__version__)
```

2.2 Access Data

First, make sure that you added the folder to your drive:

Go to your google drive (sign in through your post.bgu.ac.il account) Upload the folder containing this file to your google drive. Open Exercise1.ipynb file from Colab: <https://colab.research.google.com> Run the following code block. You will get a link to authorize the access to your drive. Click on the link and follow the steps. Copy the authorization code and paste it here. This procedure should only be required once.

```
»from google.colab import drive »drive.mount('/content/drive')
Sanity check: Make sure you have access to the files
»root_dir = '/content/drive/My Drive/Exercise1/Data' »os.listdir(root_dir)
```

3 Exercise

3.1 Create Dataset

Write a function, that given a path to the csv file, creates a `tf.data.Dataset`, parses the file and loads the image: The main functions you should use are:

```
tf.data.experimental.CsvDataset tf.io.read_file tf.image.decode_image (Note
that the raw images are of type uint16 and the segmentations are of type uint8)
»def create_dataset(csv_filename): » pass # Do things here, return the
dataset » # return dataset »train_csv = os.path.join(root_dir, 'train.csv')
```

```

»val_csv = os.path.join(root_dir, 'val.csv') »train_dataset = create_dataset(train_csv)
»val_dataset = create_dataset(val_csv) »train_dataset=train_dataset.repeat(None)
# Repeat dataset indefinitely »val_dataset=val_dataset.repeat(None) # Repeat dataset indefinitely

```

If everything works correctly, the following code snippet should display an image and corresponding segmentation

```

»for image, seg in train_dataset: » I = image.numpy() » S = seg.numpy()
» plt.figure() » plt.imshow(I.squeeze()) » plt.figure() » plt.imshow(S.squeeze())
» break

```

```

Batch the dataset »# Write your code here »# train_dataset = .... »# val_dataset = ....

```

If everything works correctly, the following code snippet should display the tensor size including (batch, image height, image width, channel)

```

»for image_batch, seg_batch in train_dataset: » print('Image Shape: ', Segmentation Shape: '.format(image_batch.shape, seg_batch.shape)) » break

```

3.2 Add Augmentations (Optional but recommended)

Add random augmentations such as random flips, rotates, crop, image intensity and what ever comes to mind. (Make sure to apply the relevant augmentation to the segmentation too)

```

»# Write your code here: »for image_batch, seg_batch in train_dataset:
» I = image_batch[0].numpy() » S = seg_batch[0].numpy() » plt.figure() » plt.imshow(I.squeeze()) » plt.figure() » plt.imshow(S.squeeze()) » break

```