Final Project Instructions - Microscopy Cell Segmentation

Assaf Arbelle, Tammy Riklin Raviv

Deep Learning and Applications Course 2018

1 Introduction

This project is intended to give the student practical experience in designing, coding, and training a Deep Neural Net (DNN). This project is suitable for all levels of expertise, requiring minimum knowledge of python programming and Tensroflow. We propose a code package that requires minimum coding of a network architecture. Although it is not necessary, students that feel they are capable, are highly encouraged to change any and all of the suggested code.

2 Project Objective

The objective of this project is to preform segmentation of cell nuclei in microscopy images. The data is decribed in Section 2.2. The students are requires to design and train a DNN than preforms pixel-wise segmentation into three classes i.e, background, cell nucleus and cell contour. The students are free to design and choose any architecture to preform this task.

2.1 Notations and Loss function

Let Ω denote the image domain of height H and width W and let $I : \Omega \rightarrow \mathbb{R}$ denote the input image. Let $G : \Omega \rightarrow \{0, 1, 2\}$ denote the ground truth segmentation of each pixel into one of three classes corresponding to background, cell nucleus and cell contour. We denote the output of the DNN with parameters Θ as $P_\Theta \in \mathbb{R}^{H,W,3}$ which holds the class probabilities for each pixel. The loss for training, as is currently implemented in the code, is the weighted cross entropy loss:

$$L = -\sum_{x=0}^{W}\sum_{y=0}^{H}\sum_{c=1,2,3} w_c \cdot \delta(c, G(x, y)) \cdot log(P_\Theta(x, y, c)) \tag{1}$$

2.2 Data

The data for this project is Microscopy Live Cell images. The raw images gray level images and saved in uint16 format .The segmentation images hold one of three values, (0,1,2) for each pixel corresponding to the pixel's class: 0-background, 1-cell nucleus, 2-cell contour. The segmentation images are saved

in uint8 format. Note, If you try to view the images and only see black, it is probably because of the dynamic range of the image. Try to open the image with Python/MATLAB and display a normalised image. Example of a cropped image is available in Fig 1

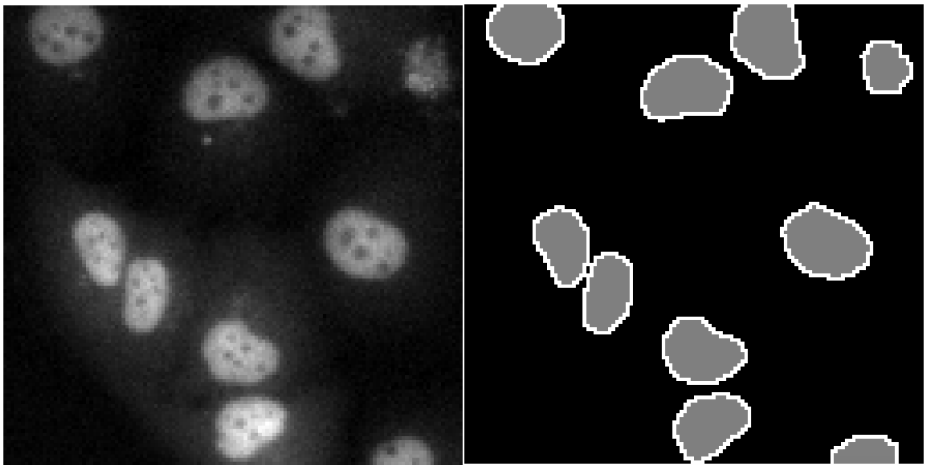

Image Segmentation

Fig. 1: An example of the image and the corresponding ground truth segmentation

2.3 Evaluation Metric

In addition to the loss function, the code measures the Jaccard index to evaluate the quality of the segmentation. For a given index the foreground is calculated as the pixels where the maximum probability is of cell nucleus:

$$F_{net} = (x, y) : \arg\max_{c}(P_{\Theta}(x, y, c) = 1 \tag{2}$$

$$F_{gt} = (x, y) : \arg\max_{c}(G_{(}x, y) = 1 \tag{3}$$

The Jaccard index is defined as:

$$J = |\frac{F_{net} \cap F_{gt}}{F_{net} \cup F_{gt}}| \tag{4}$$

3 Running The Code

3.1 Designing Your Network

In order to design your network, write your code in the supplied function "Network.py" under the build() method of the class Network(). (line 8) Make sure to follow the following instructions:

1. The first input to the function is a tensor of size (B,H,W,1) where B is the batch size. Do not change the first input!
2. You may add any additional inputs to the function.b(see "*example_parameter*"). All Additional parameters should be passed through the Param.py file under "*net_build_params*".
3. All of the network definition should be done inside the build function.
4. The output of the network should be of size (B,H,W,3) corresponding the three UNNORMALISED class probabilities. (Do not add a softmax layer at the end!)
5. The build function should return ONLY the net output as defined above.

3.2 Setting Parameters

Open the "Params.py "file and edit the parameters under the parameter class Params() (line 56). DON NOT CHANGE ParamsBase() !! The following is a list of all the parameters:

1. *net_build_params* optional paramaters for you network, use key-value pairs.
2. *use_gpu* IF NO GPU AVAILABE, SET TO FALSE
3. *gpu_id* IF MORE THAN 1 GPU IS AVAILABLE, SELECT WHICH ONE
4. *dry_run* SET TO TRUE IF YOU DO NOT WANT TO SAVE ANY OUTPUTS (GOOD WHEN DEBUGGING)
5. *profile* SET TO TRUE FOR THROUGHPUT PROFILING
6. *batch_size* number of images per batch
7. *num_iterations* total number of itterations
8. *learning_rate* learning rate
9. *crops_per_image* number of random crops per image, used for faster training
10. *crop_size* crop size of the input image
11. *class_weights* Weight for loss on Foreground, Background and Edge
12. *validation_interval* number of train iterations between each validation step
13. *load_checkpoint* set to True if you want to load model weights from checkpoint
14. $load\_checkpoint_patg$ Path to checkpoint
15. *experiment_name* Name of your experiment. Will be used for directory names
16. *save_checkpoint_dir* Path to save files. set at top of this file
17. *save_checkpoint_iteration* number of iteration between each checkpoint save
18. *save_checkpoint_every_N_hours* Keep checkpoints at intervals of time
19. *save_checkpoint_max_to_keep* Maximum recent checkpoints to keep
20. *write_to_tb_interval* number of iteration between each print to tensorboard

3.3 Running the Code

Run the function "*train_SegNet.py*"

4 Submition Instructions

When submitting the project please submit the training code, along with a trained checkpoint. Please also submit a document clearly explaining your project. The document should include at least:

1. Names and IDs.
2. Detailed explanation of the project (i.e Network Architecture, paramterts ets)
3. Detailed explanation of the training process (Losses, regularizations, batch size, crop size, etc)
4. figures of the loss curve and examples of results (from tensorboard).
5. A summary of your work.
   Please submit all the files in a zip file. ONLY ONE SUBMISSION PER GROUP.

5 Python Dependencies

Before running the code, please make sure that you have all following dependencies installed on your computer:

1. Python 3
2. Tensorflow (1.3.0 or greater)
3. Numpy
4. open-cv (cv2)
5. csv
6. glob

You can use the requirements.txt file to install all requirements. using "pip3 install -r requirements.txt"