Fall 2024 ECE 364 Bonus Project Option # 1: Background-Foreground Separation

Due Date: **December 21, 11:59pm on Gradescope**

1 Project Background

We have provided a dataset that captures vehicle traffic camera images in the suburbs of Chicago, IL. The dataset is partitioned into four separate scenes of a single camera location, i.e. traffic intersection. Each scene is the East, North, South, or West direction of the intersection, respectively, and each scene has 50 (image, annotation) pairs. The annotations for this dataset outline pixels that contain any foreground objects that interact with the roadway, i.e. any vehicles or pedestrians. This task is referred to as **background foreground separation**.

The dataset is split into a `Images` and `Annotations` folder. For a given `<scene name>`, `<date>`, and `<timestamp>`, we can find the corresponding input image and annotation image in the path "`Images/<scene name>/<date>/<timestamp>.jpg`" and "`Annotations/<scene name>/<date>/<timestamp>.png`". For example, we may find one (image, annotation) pair with:

- `<scene name>`: "Buffalo Grove at Deerfield East"
- `<date>`: "2018-08-26"
- `<timestamp>`: "2018-08-26-09-20"

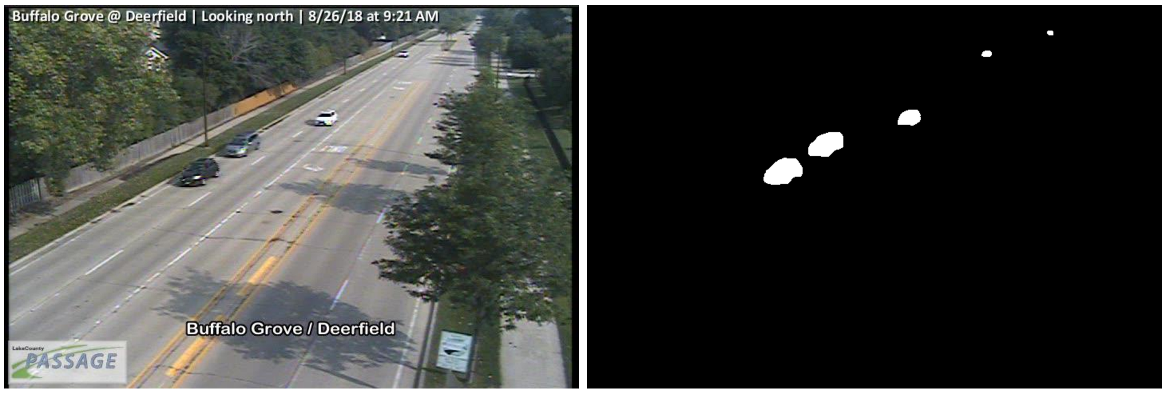
The resulting pair is shown in Fig. 1.

The task of background foreground separation (BFS) may be seen as a two-class semantic segmentation problem where we have a background and foreground class. Thus, we may either have each pixel give the probability of background and foreground or simply the probability of foreground since the two probabilities are complementary. Either approach works equally well, but the two approaches require different choices of loss function. The primary metric we will use to evaluate models for this project is intersection over union (IoU). We have discussed IoU in previous lectures and homeworks, but will briefly review here as well. For two binary images, $\mathbf{Y}$ and $\hat{\mathbf{Y}}$, that represent the annotation and prediction images, we may compute the IoU between them as

$$\text{Intersection Over Union} = \frac{|\mathbf{Y} \cap \hat{\mathbf{Y}}|}{|\mathbf{Y} \cup \hat{\mathbf{Y}}|} \in [0, 1]. \tag{1}$$

Above, $|\mathbf{Y}|$ gives the number of non-zero elements on $\mathbf{Y}$, $\cap$ denotes the intersection symbol, and $\cup$ denotes union. We have provided an evaluation function that computes the IoU across a provided dataset for a given model (contained in the `iou` function of `iou.py`). The provided function has comments and docstring explaining the expected shape of inputs and resulting outputs of the function.

Helpful Suggestion: You may consider resizing the images and annotations by reducing the size, e.g. by a factor of two in both dimensions, to speed up training and evaluation whether you use CPU or GPU device for training. Reducing size by a factor of two, three, or four is okay for these images. We only ask that you explain any resizing chosen for your implementation.


Buffalo Grove @ Deerfield | Looking north | 8/26/18 at 9:21 AM
Buffalo Grove / Deerfield
PASSAGE

(a) Image (b) Annotation

Figure 1: Example (image, annotation) pair.

2 Deliverables

We have provided a file titled `deliverables.py` as a placeholder for which classes/functions we expect you to implement, i.e. dataset, deep net class, training loop. You do not need to use this file, but it may be a helpful reminder or suggestion for how to structure the different necessary parts of your implementation.

1. Create a PyTorch `Dataset` class for the semantic segmentation problem. **Your dataset class must return an (image, annotation) pair in order to work with the provided evaluation code**. (Note: we recommend your dataset class represent the data for a single camera view/scene; although, you may create your dataset however you see fit.)
2. Create a deep neural network class through the PyTorch `nn.Module` base class to implement your semantic segmentation model.
3. Write the necessary training loop code to train your semantic segmentation model. (Note: we have provided the necessary evaluation code to compute the intersection-over-union (IoU) across a dataset of images and a given model, please refer to explanation/documentation with the function.)
4. Train your model in **one** of the following setups:
   (a) Train your model on all four scenes with half the images ($\approx$ 100 images) used for training and half the images for validation ($\approx$ 100 images). Provide the training and validation IoU values of your model on each of the four scenes. You may plot these metrics against training epoch number or simply give the final values.
   (b) Train your model on one of the four scenes with half the images used for training ($\approx$ 25 images) and half the images for validation ($\approx$ 25 images). Provide the training and validation IoU values of your model on the training scene, and the validation IoU values on the remaining three validation scenes. You may plot these metrics against training epoch number or simply give the final values.

3 Submission

1. Share all of your codes that implement your dataset, model, training loop, and evaluation in a .zip file.
2. Submit a 2 page PDF report (1" margins, 12pt font) that:
   (a) Describes the implementation of your dataset.
   (b) Describes the implementation and design of your deep neural network.
   (c) Describes your training algorithm, chosen parameters, any other design choices.
   (d) Discusses your training and validation results for your chosen setup of Item #4 above, i.e. which scenes are easiest/hardest, how do models transfer to scenes/images they were not trained on?