CSCI 669: Mid Term Report

Hayley Song
USC ID: 5276048267
haejinso@usc.edu

In this midterm report, I will present my project's progress up till now. In summary, I have completed the pre-processing, data preparation, evaluation metrics implementation, and post-processing steps. I also established five baseline models on the SpaceNet dataset whose performances are discussed in Section 4. I will discuss how I define my geo-spatial ontology on road, building and vegetation based on the OpenStreetMap's tags(Section 5), and conclude the report by discussing the remaining steps, in particular how to build knowledge graphs (KG) from the satellite images and incorporate the KGs into the segmentation pipeline (Section 6).

1 Data preprocessing

As proposed in my project proposal, the satellite images come from the SpaceNet Challenge dataset (SpaceNet on Amazon Web Services, 2018), which contains two types of data: vector data and raster data. The vector data contains the roadnet work in geojson format, and the raster data consists of multispectral satellite images and their metadata for georeferencing. More specifically, the raster data contains panchromatic, RGB and 8-band channels. Since the dataset does not come with labelled road mask images (,and semantic segmentation works on pairs of (image, mask)), a number of preprocessing steps are required to generate segmentation masks for training and evaluation.

I experimented with three different methods. First, the tool sets by ComiQ [1] provide useful methods to generate the line strings in Geojson format to a segmentation mask. In particular, it allows the user to specify the width of the road mask. I experimented with different widths (1, 2, 3, 6 meters) and settle with the width of 2 meters as its result seem the best qualtitatively. Although this tool was useful, it has the limit that all roads are considered to be of the same width, regardless of its type (eg. Highway, Residential, etc).

Figure 1: Preprocessing methods to generate road masks from the satellite images and the vector files. Left: RGB image (Vegas image 1035). Center: 2 meter width using CosmiQ. Right: FloodFill approach

In order to address the variability in the road types, I extracted information on the road type, road condition (paved vs. unpaved) and number of lanes from the metadata in Geojson files. I then applied different road widths depending on the extracted information, which resulted in more accurate labels. Lastly, I tried the floodfill method to propagate the road labels via neighboring pixels. I used the original vector data as the "seed points" and used the OpenCV's implementation [2] to apply the floodfill method. Figure 1 shows the results of these methods.

It is clear that the floodfill approach achieves more accurate labelling: major truck roads and cul-de-sacs are more correctly labelled, while objects on the roads such as cars and vegetation are no longer incorrectly labelled. After generating the segmentation masks, I split the dataset into training, validation and test set by 70%, 15% and 15% ratio.

For the raster data, I scaled RGB channels to the same range by finding the average minimum and

[1] https://github.com/CosmiQ/apls

[2] https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html

maximum RGB values per city in the train dataset and scaled the images within this range.

2 Evaluation metrics

Two most popular datasets for aerial images before Spacenet Challenge Dataset became available are: Inria Aerial Image Labeling dataset (**?**) and Massachusetts Buildings dataset (**?**). In order to enable performance comparison with previous works, we use the same performance metrics that have been used on these datasets. Three commonly used evaluation metrics are discussed below.

2.1 Intersection over Union (IOU)

IoU, also known as Jaccard Index, is a pixel-based metric defined as:

$$IoU(GT, P) = \frac{GT \cap P}{GT \cup P)}$$

where GT is the ground truth mask and P is the predicted mask.

2.2 Pixel-Based F1

F1 score is computed by taking a harmonic mean of precision and recall in a standard way. The pixel-based precision and recall values are computed by categorizing each pixel prediction as one of true positive, false positive, true negative, and false negative.

I focused our experiments on IoU as it has become a standard for semantic segmentation [13]. Moreover, pixel-based F1 score is less desirable in road detection as large image areas are dedicated to the background (non-road) class.

2.3 Relaxed F1

For Massachusetts buildings dataset, a relaxed version of precision and recall are used to calculate the precision-recall breakeven point [17]. The relaxation assumption is to consider a positive label correct if it falls within a Q $prod Q$ neighborhood (eg.7x7 region) of any ground truth positive pixel. Then, the relaxed precision is the fraction of positive labels that has a positive ground truth label within Q pixels away. The relaxed recall is the fraction of true road pixels within Q pixels of a predicted road pixel. Since the road masks are usually not perfectly aligned to the image, this metric will provide a more realistic performance measure than the unrelaxed F1.

However, if we are interested in the network connectivity of the extracted roads (eg. route planning), none of these metrics serve as a good measure since they fail to incentivize the creation of connected road networks. For instance, the IoU metric will heavily penalize a small error in the road width, while lightly penalizing a small break in the predicted road, as illustrated in Figure

2.4 Average Path Length Score (APLS)

In order to complement this limitation, I added the third metric, Average Path Length Score (APLS) (Van Etten et al., 2018), which measures the similarity between two networks by taking into account both the **physical** and **logical** topology of their graph structures. Here, logical topology implies the connections between nodes. Given the ground truth graph G (available in the vector files) and a predicted graph G', APLS computes the shortest distances between the corresponding pairs of nodes from G and G' and uses the average as the measure of similarity. Note that G' needs to be inferred from the prediction mask P through extra post-processing steps. Details on this "Mask to Graph" generation is described in Section 3. Assuming that we have two graphs, G and G', APLS is computed as following. Refer to (Van Etten et al., 2018) for the detailed justification and our Github repository [3] for the implementation.

2.4.1 Graph Augmentation

The first step in computing APLS between two road network graphs, G and G', is to select which routes to compare. We consider the nodes of the graphs (ie, road intersections, end points and mid points) as the "control points" (as in Graphics applications) and insert extra points every 50 meters frome each node along the edges.

2.4.2 Node Snapping

In practice, the nodes in G and G' rarely align perfectly with each other, and we need to match the control points of a graph to another. We set a buffer (eg. 4 meters, equivalent to the size of buffer for computing the relaxed F1 score), and pair each node in G to a node in G' within the buffer. If there is no proposed node within the buffer, there will be no matching pair, and APLS will assign the largest penalty. Once node snapping is complete, we can compare a path between node a and b in G (path(a, b)) with the path between its "snapped" pair a' and b' in G' (path(a', b').

[3] https://github.com/cocoaaa/IE/tree/master/Project/Scripts/APLS

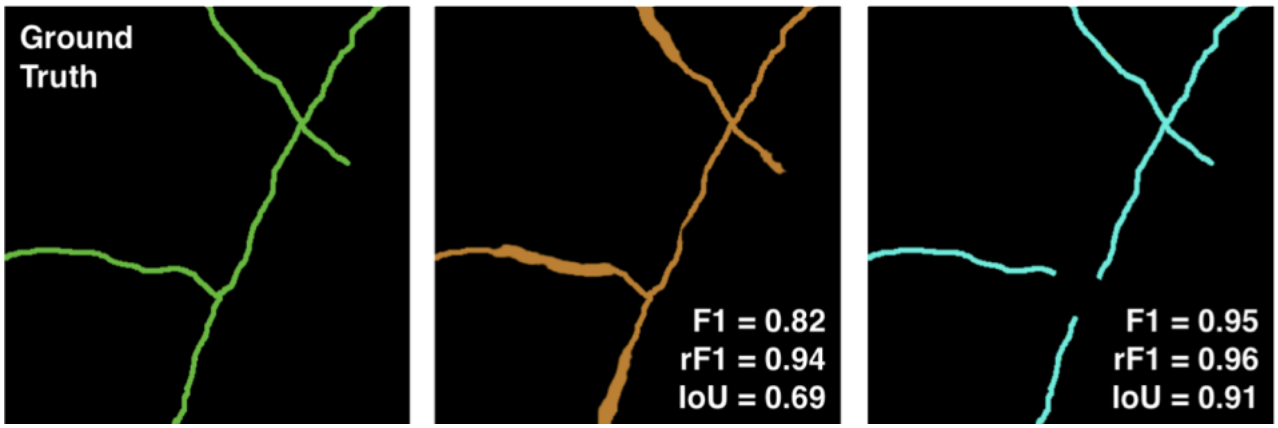

Figure 2: Limitation of IoU metric. Image Credit: Van Etten et al. (2018). Left: Ground truth road network. Center and Right: two different predicted road masks. Center mask preserves the network connectivity better than the one on the right, yet traditional metrics (F1, Relaxed F1 and IoU) favor the latter.

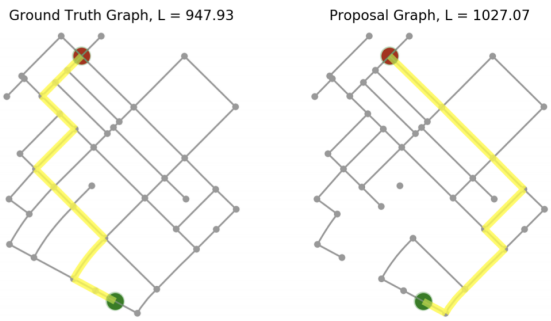

Figure 3: Shortest path in G and G'. We run Dijakstra's algorithm to compute the optimal path lengths.

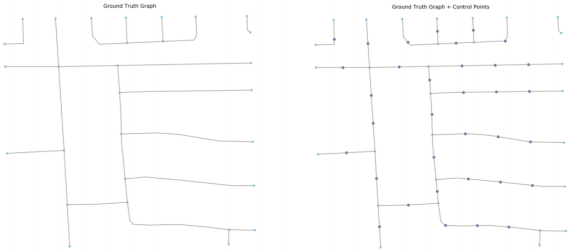

Figure 4: APLS 1: Graph augmentation. We insert new control points to G and G' at every 50 meter along the edges.

2.4.3 Symmetric comparisons

Above, we performed the node snapping from G to G'. In order to penalize spurious road proposals (Van Etten et al., 2018), we also need to perform the same procedure in the opposite direction, from G' to G.

2.4.4 Shortest path length using Dijakstra

After establishing the node correspondences between G and G', we run Dijakstra's algorithm (**?**) to compute the shortest path length of all possible paths. An example of such computation is illustrated in Figure 3.

We then normalize the difference in the two paired paths:

$$\frac{\|L(a,b) - L(a',b')\|}{L(a,b)}$$

where $L(a,b)$ is the shortest path length between node a and node b in G, and $L(a',b')$ is the equivalence in G'. APLS of (G,G') is defined as:

$$APLS(G,G') = \tag{1}$$
$$1 - \frac{1}{N}\sum \min(1, \frac{\|L(a,b) - L(a',b')\|}{L(a,b)})$$

where N is the number of unique paths, and the sum is taken over all possible source (a) and target (b) nodes. Note that APLS ranges from 0.0 (worst) to 1.0 (best). Importantly, APLS reflects the notion of "node centrality" which is the frequency of a node appearing in the graph's shortest paths. If a node with high centrality is missing in the predicted graph, Equation (1) will assign a high penalty. See Figure 4 and 5.

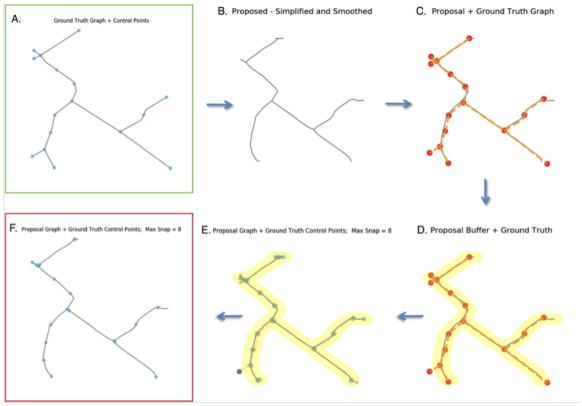

Figure 5: APLS 2: Node snapping. Image credit (Van Etten et al., 2018).

3 Post-processing: Mask to Graph

Once we have a proposal mask of the road network, we need to transform this two dimensional, boolean array into a graph (ie. a set of nodes and edges). Our overall workflow is: (1) skeletonize the mask, (2) generate a graph from the skeleton image, (3) handle the curvy parts of the graph, and (4) perform optional post-processings. See Figure 6.

Given the probability mask, we first threshold (ie. 0.3) it to get a binary image. We also remove some noise by filtering out small objects and holes (smaller than 300 pixel area). Then, we run a skeletonization algorithm on the binary image to generate a thin representation of the predicted mask.

Next, we transform the mask to graph using `sknw` [4]. The resulting graph is a multi-grpah with vertices on crossroads, which requires to be transform into straight edges. For the straightening, we run an approximation algorithm from OpenCV to represent each edge as a sequence of straight segments. In order to achieve a better graph structure, we add vertices for all of the curvy parts of the graph.

We experimented with extra post-processings such as removing small isolated subgraphs and graph correction near the edges.

[4] https://github.com/yxdragon/sknw

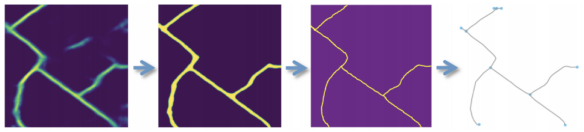

Figure 6: Transform a segmentation mask to a graph. Left to Right: probability mask, thresholded mask, skeletonized mask, graph result

4 Experiments: Baseline Results

I conducted various baseline experiments with different input training data type (eg. the image channel types (eg. all channels, RGB, Red/Near Infar-Red (R/NIR), mask generation methods(eg. floodfill methods, road width adjustment), neural network architectures, and loss functions for training. I report each experiment setting and correpsonding results in this section.

4.1 Input channels

From the multispectral dataset, I experimented with different spectral ranges.

- **RGB** I first used only the Red, Green, and Blue channels for training and evaluation.
- **R/NIR** Asphalt tends to be a poor reflector in the red and near-infrared regions of the spectrum, whereas vegetation strongely reflect energy in those regions due to chrolophyll contents. So I extracted the last 3 bands of the pan-sharpened imagery correspond to the red-edge, NIR1, and NIR2, and used them for training and evaluation.
- **8 bands** I used all channels provided in the dataset.

In all three cases above, each image was resized to 400x400 and scaled to 8-bits with a contrast stretch clips of 10% and 90%. I generated the ground-truch segmentation masks using `Cosmiq`'s methods and did not apply any additional pre-processing (i.e. no floodfill).

I used the Tiramisu architecture (described in Section 4.2) for all three scenarios. Table 1 summarizes the best performances in each case. Figure 7 and Figure 8 show the intermediate and final results of training on RGB channels; Figure 9 and Figure 10 on R/NIR channels; Figure 11 and Figure 12 on 8-bands.

For the rest of the experiments, I used all 8 bands for training and evaluation.

Figure 7: Tiramisu + RGB intermediate prediction

Figure 8: Tiramisu + RGB final prediction

Figure 9: result from Tiramisu + R/NIR imagery from Vegas AOI

Figure 10: Another result from Tiramisu + R/NIR imagery from Vegas AOI

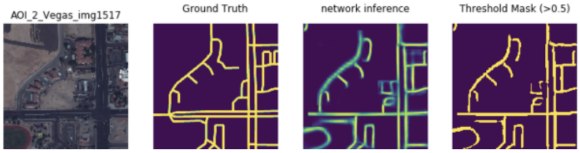

Figure 11: Result from Tiramisu + 8-band imagery from Vegas AOI

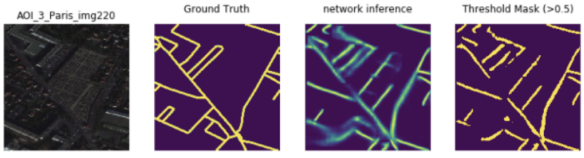

Figure 12: Another result from Tiramisu + 8-band imagery from Vegas AOI

	RGB	R/NIR	8-bands
IoU	0.61	0.79	0.81
F1	0.17	0.21	0.24
APLS	0.51	0.55	0.60

Table 1: Performances of Tiramisu model on different input channels

4.2 Model Architectures

Transfer learning I applied transfer learning techniques to various vanilla models in order to fine-tune the model for our dataset. The models I experimented with are: ResNet + LinkNet34, Inception-based encoders + LinkNet, and UNet with pre-trained VGG11 encoder VGG11 + UNet. For all these models, I used $0.8 \times \text{BCE} + 0.2 \times (1 - \text{DICE})$ as the loss function, where BCE stands for the standard Binary Cross-Entropy Loss and DICE stands for the DICE coefficient, and used Adam as the optimizer. Note that DICE coefficient is proportional to the IoU metric discussed in Section 2). The best performing model was ResNet34 as encoder and UNet-like decoder with **0.89** IoU and **0.61** APLS.

4.3 Loss Functions

Using the ResNet + UNet model, I experienced with various loss functions: average per-pixel BCE, DICE coefficient, and Iighted loss of form, $w_1\text{BCE} + w_2\text{DICE}$. The weighted loss with $w_1 = 0.8$ and $w_2 = 0.2$ achieved the best performance, yet I did not observe a significant impact of choosing one of the other loss functions.

5 Spatial Ontology and KG

My project aims to build a more generalizable segmentation classifier via using external information from the OpenStreetMap database at both training and prediction time. In this section, I present my own spatial ontology, which is designed for our specific task to extract road network and will be used to construct knowledge graphs from the input dataset (images and vector files containing geospatial metadata). I designed my ontology in consultation with the OSM's tag guideline and road-related ontologies in (Katsumi and Fox, 2018). The two main classes in our ontology are `spatial objects` and `spatial relations` (Figure 13). `spatial objects` has four subclasses: `Road`, `Building`, `Water`, and `Vegetation`; `spatial relations` has 12 subclasses that

describe the spatial relationships between two spatial objects: `surrounded by`, `adjacent to`, `intersects with`, `within 1km`, `within 1 and 5km`, `within 5 and 10km`, `farther than 10km`, `is north of`, `is south of`, `is west of`, `is east of`, and `relative size`. Each `spatial objects` instance also has attributes (aka. labels, properties) such as `area`, `length`, `usage` and `type`. See Figure 13 for the visualization of my ontology.

6 Next Steps

Lastly, I would like to summarize my next steps to complete the final project. First I will build a knowledge graph by detecting buildings, water bodies and vegetations using pretrained models. More specifically, I can detect the water bodies and vegetations with high accuracy from the multispectral channels: this is a well-studied topic in remote sensing. Then I will run a building detector (either Faster RCNN pretrained on ImageNet or Tiramisu/UNet variant model trained on other existing building datasets). Once I have these three types of object instances detected, I can incorporate them with the OSM database and the geospatial information in the vector data to construct a knowledge graph.

The second step is to use a Graph Search Neural Network as a way to walk through the KG and learn a per-node "important" score. I will investigate several networks such as Gated Graph Sequence Neural Networks [4] and Graph Partition Neural Networks [5], and similar approaches from Marino et al [6]. In addition, I will use memory networks [7] with attention [8] over the OSM knowledge base to select relevant information.

Figure 13: Spatial ontology for my segmentation task. The class of `spatial object` consists of Road, Building, Water, and Vegetation, and the class of `spatial relations` consists of 12 subclasses that describe the spatial relationships between two spatial objects.

References

Megan Katsumi and Mark Fox. 2018. Ontologies for transportation research: A survey. *Transportation Research Part C: Emerging Technologies* 89:53–82. https://doi.org/10.1016/j.trc.2018.01.023.

SpaceNet on Amazon Web Services. 2018. `https://spacenetchallenge.github.io/datasets/datasetHomePage.html`. Last modified April 30, 2018. Accessed on Oct. 30, 2018.

Adam Van Etten, Dave Lindenbaum, and Todd M. Bacastow. 2018. SpaceNet: A Remote Sensing Dataset and Challenge Series http://arxiv.org/abs/1807.01232.