Jiamin Tan

CPLN 680

03/04/2022

Mid-Point Draft

**Object**

Use neural networks to identify roads from remote sensing images.

**Motivation**

I want to learn deep learning techniques and apply them on data related urban issues. In this project, I would like to build a neural network to identify roads from remote sensing images. The outcome of this project can contribute to open-source dataset of urban roads through automation rather than manual mapping. I had experiences of classifying different land uses using other machine learning techniques in R, so I want to learn about deep learning using Python.

**Datasets Identified**

[SpaceNet](https://spacenet.ai/), an open-source project for building machine learning algorithms on remote sensing imageries, provided several challenges with labeled training and testing dataset. Specifically, their third challenge features [road network detection](https://spacenet.ai/spacenet-roads-dataset/).

**Dataset Summary**

The dataset mentioned above contains satellite images from four cities: Las Vegas, Paris, Shanghai, and Khartoum. This project will focus on the Shanghai dataset first, but if there is extra time, other cities would also be considered. The Shanghai dataset contains WorldView-3 images that cover 1000 square kilometers of the city with 3537 km road center lines labeled (SpaceNet and Van Etten et al., 2018). Road label are provided in .geojson format. Types of roads are also differentiated in the labels. The dataset provides four types of images: multi-spectral images, panchromatic images, pan-sharpened multi-spectral images, and pan-sharpened RGB imagery. This project will focus on using pan-sharpened RGB images because those are the most intuitive ones for humans. The training set has 1198 images, and the test set has 399 images. However, the labels of the test set are not provided since the dataset was used for a challenge. Thus, for training the neural network in the next step, I will split the original training set into a new training set and a new testing set. The dimension of each image is 1300 x 1300 pixels, and each pixel has a 0.31m x 0.31m spatial resolution. In other words, each image covers a 400m x 400m area in Shanghai. The color depth is 48-bit, so each band (red, blue, and green) has a 16-bit color depth. That means a pixel value in each band ranges from 0 to 65535. Not all images come with perfect condition because some of the images were damaged as they were, so no labels were provided for those images.

**Methods/What I am currently working on**

*Mask Generation*

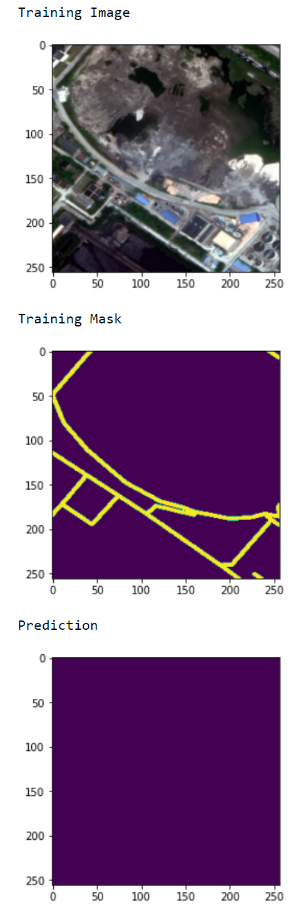
My first step is convert the .geojson files into binary masks – road pixels will be buffered from the vector strings and labeled with 1 while non-road pixels will be labeled with 0. Using some Python scripts provided by the challenge organizer in in this [GitHub repo](https://github.com/CosmiQ/apls), I have generated masks for all 1198 images in the training set. The given scripts also generate an 8-bit image for each 16-bit pan-sharpened RGB image in the training set. For future steps, the 8-bit images will be used since they are smaller in size and easier to manipulate in Python.

*A Vanilla U-Net CNN*

I’ve been focused on learning convolutional neural networks. I learned that the U-Net architecture published in a [Ronneberger et al. (2015)](https://arxiv.org/pdf/1505.04597.pdf) paper is popular and useful for solving image segmentation problems. To learn how to implement the U-Net architecture in Python, I watched a series of tutorials by [Sreenivas Bhattiprolu (DigitalSreeni)](https://www.youtube.com/channel/UC34rW-HtPJulxr5wp2Xa04w) on YouTube. Based on his tutorials and codes, I wrote a vanilla U-net model using Tensor Flow and Keras in Python. Since the dataset I am using is too big to be uploaded to my GitHub repo, I saved a .html version of the Jupyter Notebook with codes and results by running it on Google Colab. The file can be found [here](https://drive.google.com/file/d/1K2kJQy5WJ1AD8c0fg1CMaSd1bTmal6Z5/view?usp=sharing) or in my GitHub repo - it is called “Vanilla\_U-Net\_Example\_on\_RS.html”.

Graphical user interface

Description automatically generated with medium confidenceI used the same U-Net architecture published in Ronneberger et al.’s paper, except the size of the input image. Currently, it is hard coded, but this help me to understand how the layers are generated and used. I will make better engineering later in the project. I resized all my training images/masks from 1300 x 1300 pixel to 256 x 256 pixel. Given the length and width of each image is 400 meters on the ground, the training image now has a spatial resolution of 1.56 x 1.56 meter. The masks were 2 meters, but as they are getting resized, neighboring pixels might be also aggregated as roads. I tried to resize the training images and masks into 512 x 512 pixel, but Google Colab gave me an out-of-memory (OOM) error when training the model.

A picture containing graphical user interface

Description automatically generated Currently, the result of the model is not ideal at all. I am not sure whether it is I coded something wrong (more likely), or the vanilla U-Net naturally doesn’t work well with my data. I randomly tried to visualize multiple prediction results using training images, but no roads were shown.

Graphical user interface, chart

Description automatically generatedOne guess I have is that there are too many non-roads pixels which can be very similar to the road pixels after resizing the image. I still need to investigate in that. In addition, TensorFlow did provide some tools for analyzing the model. For example, the epoch accuracy and loss over the number of epochs were provided. I am learning to understand how the information can be used for improving the model.

**Next Steps**

1. investigate if there is something wrong with my hard coded U-Net architecture.
2. Read more articles on road segmentation using U-Net and experiment some of those architectures.
3. Transform the current hard coded scripts with better engineering.
4. Create masks which take consideration of types of roads.
5. Figure out a way for adding spatial information (such as adding rasterized(?) OSM building data).

**Deliverables**

The final deliverable will be a report documenting the methods and result analysis. A GitHub repo containing all the data and codes will be created and maintained.

**References**

Bhattiprolu, Sreenivas. “U-Net for image segmentation”. YouTube. March 2022, from

https://www.youtube.com/playlist?list=PLZsOBAyNTZwbR08R959iCvYT3qzhxvGOE

Ronneberger, Olaf, Fischer, Philipp, & Brox, Thomas (2015). U-Net: Convolutional Networks for

Biomedical Image Segmentation

SpaceNet 3: Road Network Detection. https://spacenet.ai/spacenetroads dataset/

Van Etten, A., Lindenbaum , D., & Bacastow , T.M. (2018). SpaceNet : A Remote Sensing Dataset and

Challenge ArXiv , abs/1807.01232