Jiamin Tan
CPLN 680
04/29/2022
Final Report

Retrieving Roads from Remote Sensing Imagery

Introduction

Road data are important in city planning. Not only they model the network that measures accessibility, but also infer potential greenhouse gas emissions and the extend of urban sprawl. However, the availability of road data is limited because creating data for roads can be a laborious work. Main sources for getting public road data include open data from governments and voluntarily contributed data from Open Street Map (OSM). The availability of road data also depends on locations and governmental form. Barrington-Leigh & Millard-Ball (2017) found that although over 80% of global road data are complete on OSM, the completeness of the road data in some countries are less than one-third.

In recent years, as artificial intelligence raises in the image recognition field, road data now can be generated by segmenting road features from remote sensing images. This will save much of human efforts for creating road data. Ronneberger et al. (2015) proposed a U-Net architecture which significantly improved neural network performance in image segmentation tasks. Zhang et al. (2017) trained the Massachusetts Roads Dataset from Kaggle on different neural network architecture including U-Net and ResUNet.

With the motivation of learning neural network, this project trained models which incorporate the U-Net architecture and a pretrained ResNet50 on WorldView-3 remote sensing images of Shanghai, China. The models are trained separately using whole images and smaller parsed images from the whole images. As a result, the model trained using whole images on the U-Net architecture without a pretrained ResNet50 has the highest F1-score amongst all models.

Data

The data used in this project is from SpaceNet, an open-source project for building machine learning algorithms on remote sensing imageries. Specifically, the data is from its third challenge which features road network detection. The original dataset contains satellite images from four cities: Las Vegas, Paris, Shanghai, and Khartoum. This project focuses on the Shanghai dataset which 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. Attributes such as type and number of lanes are included in the *.geojson* files. The dataset provides four types of images: multi-spectral images, panchromatic images, pan-sharpened multi-spectral images, and pan-sharpened RGB imagery. This project uses the 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, but labels of the test set are not provided since the dataset was used for a challenge. Therefore, this project only uses the data provided in the original training set and split new validation and test sets from it.

The dimension of each *.tif* image is 1300 x 1300 pixels. 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

Data Preparation

The first step is to generate a binary mask for each image from the *.geojson* files. This step is done by modifying the given scripts from SpaceNet, which originally buffers all the road strings by 2 meters regardless of their types or number of lanes. The modified script then buffers the road string by 3 meters times the number of lanes of each road string. Although the type of road is also provided, this attribute is not used because there are more than half of the roads are labeled as “unclassified” for their types while their width varied without a fix pattern. In addition, the given script also convert all the 48 bit-depth images to 24 bit-depth images, which mean all the value of each pixel in a band is converted to fit the range from 0 to 255. Some examples of the original image, their *.geojson* labels (overlaying on the original image), and their binary masks generated (overlaying on the original image) are shown in Figure 1.

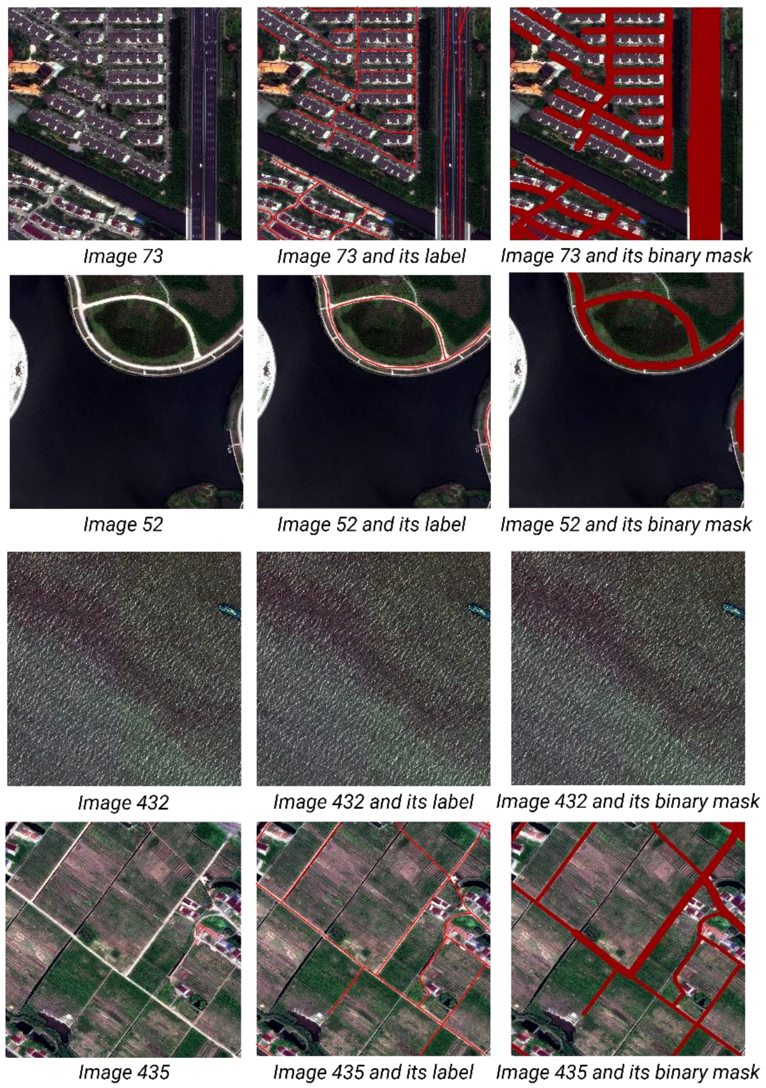
Image 73
Image 73 and its label
Image 73 and its binary mask
Image 52
Image 52 and its label
Image 52 and its binary mask
Image 432
Image 432 and its label
Image 432 and its binary mask
Image 435
Image 435 and its label
Image 435 and its binary mask

Figure 1 – Examples of original images, *.geojson* labels, binary masks

Creating Training, Validation, and Test Sets

Whole Images

The original training set provided by SpaceNet is split into a new training set of 1070 images and their masks, a validation set of 59 images and their masks, and a test set of 61 images and their masks. The images and masks in all sets above are randomly selected.

Split Images

To get a larger training size, another strategy is to parse each whole image into smaller pieces. Therefore, 120 whole images are randomly selected from the training set containing 1070 images. Each of the selected image is parsed into 25 (256, 256) smaller pieces, so 3000 smaller images are generated. From the 3000 smaller pieces, a new training set is created by randomly selected 2850 of them, and a new validation set is created by the rest 150. To compare the results in the same context, whole images in the test set mentioned above are parsed in the same way. Therefore, 1525 (256, 256) test images and their mask are created from the 61 whole image test set. Figure 2 demonstrates how the images are parse into smaller pieces.

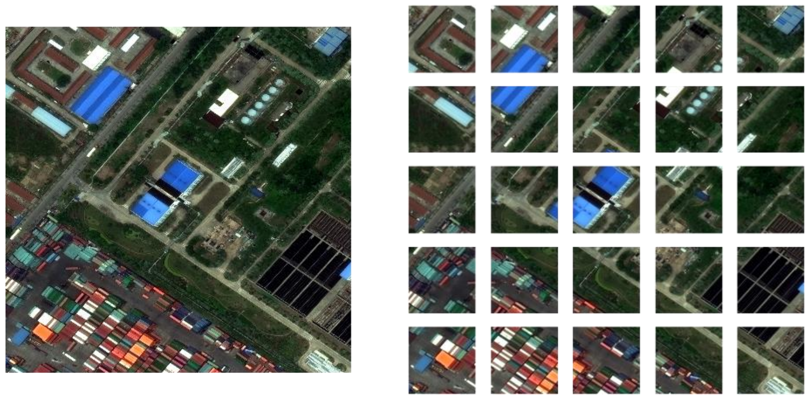
Figure 2 – Illustration of split images

"Vanilla" U-Net Model

The first model created is the "vanilla" U-Net model, which means the model basically following the original U-Net architecture proposed in the 2015 Ronneberger et al. paper. Figure 3 shows the original U-Net architecture. The U-Net have 4 blocks in the encoding path and 4 blocks in the decoding path. With in the encoding path, the input images are convolved twice and then maxpooled. A copy of the result from each block will be concatenated to the corresponding block in the decoding path. By doing so the model know "where" the interested feature is when generating the final segmentation.

The input size of this U-Net model will be (1024, 1024, 3) for the whole image sets and (256, 256, 3) for the split image sets. The original (1300, 1300, 3) whole images are resized before feeding to the model. The loss function used for this model is the dice loss which was propose by Milletari et al. in 2016. In the following equation, p stands for prediction and g stands for ground truth.

$$Dice\ Coefficient = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2}$$

$$Dice\ Loss = 1 - Dice\ Coefficient$$

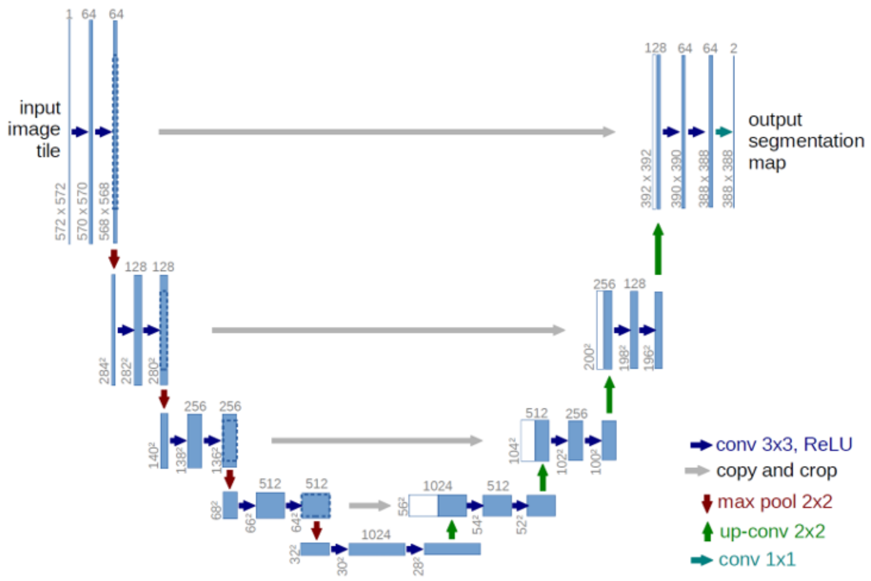

Figure 3 – U-Net architecture (Ronneberger et al. 2015)

Different from the binary cross entropy loss which calculates the average loss of each pixel, the dice loss is a regional-based loss calculating the similarity between two images and works well when the amount of different labels are imbalanced (Jadon 2020). The dice loss is chosen because the amount of non-road pixels are much more than the road pixels. The model is fed with data in batches of 2. The model runs through 12 epoches. The learning rate is set to 0.0001, and the optimizer is adam.

U-Net with Pre-trained ResNet50 Model

Besides the “vanilla” U-net, a U-Net with a pre-trained ResNet50 model is built. In deep learning practices, transfer learning using pre-trained architectures is common for performance improvement. Figure 4 shows a similar structure using ResNet34 in a U-Net architecture. The difference between ResNet34 and ResNet50 is that the prior one has 34 layers and the later has 50 layers.

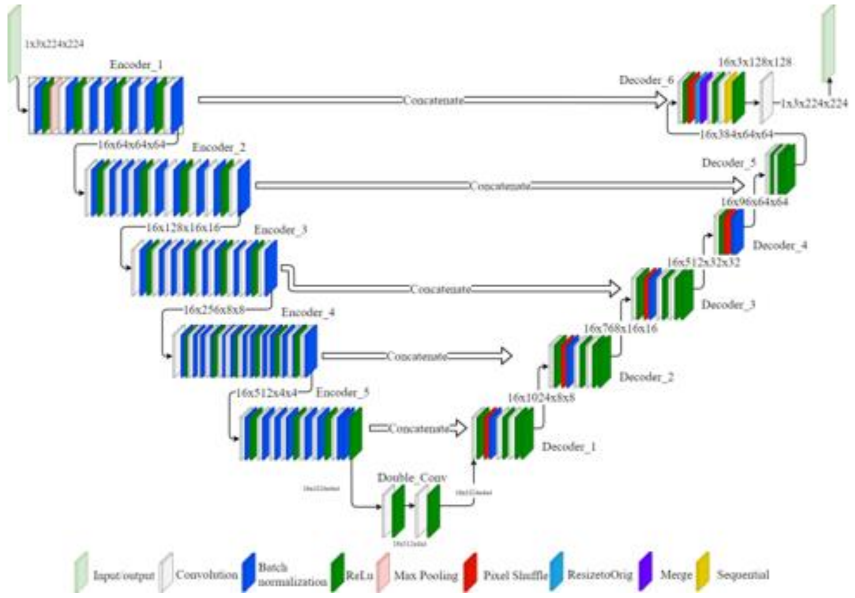

Figure 4 – Example of a U-Net using ResNet34 (Adleman, forums.fast.ai)

The input of this model is (512, 512, 3) for the whole image sets and (256, 256, 3) for the split image sets. The reason for the whole images being resized to (512, 512, 3) but not (1024, 1024, 3) this time is that the GPU used could not afford the computation for (1024, 1024, 3) which will generate more parameters with the ResNet50 comparing to the previous case. The model is fed with data in batches of 2. The model runs through 12 epoches. The learning rate is set to 0.0001, and the optimizer is adam.

F1-Score

To measure the prediction, F1-score is calculated for outcomes from each model using the same test set. F1 score is a harmonic mean of precision and recall. It is also a metric used for imbalanced data. The F1-score is defined as

$$F_1\ Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

where

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Results

The full results of predicted image can be found in the Jupyter notebooks saved in the notebook folder on the project GitHub repo. Table 1 below shows the result of the F1-scores using different image set and models.

Table 1 – F1-score of prediction using different models and image sizes

	Whole Images (1024, 1024, 3) or (512, 512, 3)	Split Images (256, 256, 3)
“Vanilla” U-Net	0.6685	0.6269
U-Net with ResNet50	0.6569	0.6292

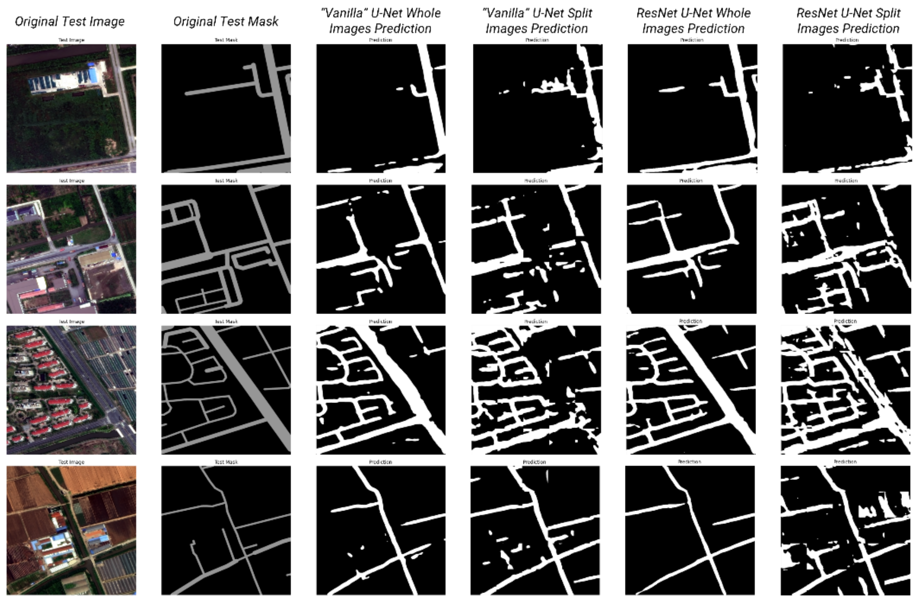
Original Test Image
Original Test Mask
"Vanilla" U-Net Whole Images Prediction
"Vanilla" U-Net Split Images Prediction
ResNet U-Net Whole Images Prediction
ResNet U-Net Split Images Prediction

Figure 5 – Random prediction examples from the test set with predictions from each model.

Figure 5 above shows some random examples from the test set prediction. From left to right, the images are the original images from the test set, original masks of the test image, predictions using the "vanilla" U-Net trained on whole images, predictions using the "vanilla" U-Net trained on split images, predictions using the U-Net with pre-trained ResNet50 trained on the whole images, predictions using the U-Net with pre-trained ResNet50 trained on the split images.

Discussion

Data Quality

The binary mask of the training does not have very high accuracy. This is caused by two reasons: 1) people who made the original *.geojson* labels might not have enough ground knowledge. In some images, one can tell some contiguous road segments have uniformed widths while the number of lanes documented says they are different. For instance, in the last binary mask in Figure 1 above, the road in the middle of the image expanded with one more lane in the middle a road which should always have one lane according to the image. 2) the missing type of road made the width less accurate because width of a lane in residential areas can be different from the width of a lane on highways. Therefore, to derive better results, instead of using the *.geojson* vector strings, one should manually label draw the masks for training images to ensure the masks are accurate. However, since the use case of such application is to gather available road data in vector format (likely from OSM), make prediction, and convert the raster result back to vector data to contribute to open data sources such as the OSM, there will be a tradeoff between human efforts of labeling and prediction accuracy for this kind of task.

Comparison Amongst Models

Larger training set but smaller image size does not necessarily lead to better predictions. The fourth and sixth columns in Figure 5 are the prediction results from models trained on split images. These are not raw results from the prediction, but concatenations of the split image individually predicted. Comparing to the predictions made from whole images, there are a lot of noises in those split image prediction. Since the spatial resolution of the training images is very high, small objects which are not aggregated into road pixels can interfere with the training process. The split images create a larger training set but sacrifice the area covered in each image which could provide more context, such as the continuation of a road.

The impact of different resolution still needs to be found out. When comparing the third and fifth column in Figure 5, which are predictions from models trained on the whole images, the fifth column shows less noises than the third column. To understand whether the difference is caused by using the pre-trained network or by different resolutions selected (i.e. 1024 x 1024 with 0.4 meters resolution vs 512 x 512 with 0.8 meters resolution), the first model with images resized to (512, 512) should be explored in the future. This also can become a research question for future: what is the optimal resolution for getting contiguous road segment predicted from remote sensing images?

Conclusion: ***what I learnt from this project***

I learnt a lot from making this project. I had zero knowledge of neural networks at the beginning of the semester, but I finally got reasonable results at the end of it. Although I do not fully understand every detail in deep learning based on what I learnt so far, I think I gained a lot through the process. My models did not work until three days before the final presentation of the class, and I could not figure out where the problems were. One thing I learnt from this project is that the process of loading data into

Python and making them ready for training is not easier than coding an existing architecture itself. In the future, I will continue to learn about neural network, and I believe the knowledge will start to accumulate when I explore further.

Acknowledgement

I want to thank Jonathan Tanne and Guray Erus who are the instructors for the MUSA Capstone class and the Remote Sensing class for all their advice. I also want to thank all my classmates for helping me and encouraging me to complete this project.

References

Barrington-Leigh, C., & Millard-Ball, A. (2017). The world's user-generated road map is more than 80% complete. *PLOS ONE*, *12*(8). https://doi.org/10.1371/journal.pone.0180698

Jadon, S. (2020). A survey of loss functions for semantic segmentation. *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. https://doi.org/10.1109/cibcb48159.2020.9277638

Milletari, F., Navab, N., & Ahmadi, S.-A. (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation. *2016 Fourth International Conference on 3D Vision (3DV)*. https://doi.org/10.1109/3dv.2016.79

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Lecture Notes in Computer Science*, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

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

Zhang, Z., Liu, Q., & Wang, Y. (2018). Road extraction by deep residual U-Net. *IEEE Geoscience and Remote Sensing Letters*, *15*(5), 749–753. https://doi.org/10.1109/lgrs.2018.2802944