Road Segmentation Classification

Lei Pang, Jingran Su, Chujun Wang

***Abstract*—In this report, our group introduces the implementation of Random Forest Classifier(RFC), Convolutional Neural Network(CNN) and a fully convolutional U-Net model(U-Net) model to classify roads on Earth satellite images. After multiple attempts, the U-Net, coupled with contracting and expansion processes, has the best performance for this project. And after a further data feature augmentation method (flip and rotation) implemented, this model could achieve a patch-wise (16x16) F1-Score of 90.0% on AICrowd evaluation system.**

I. Introduction

Satellite-based road detection has a wide range of applications in urban planning, autonomous driving, emergency command and other fields. The project is a binary segmentation problem, which requires to extract the road part from the satellite image, marking each pixel belonging to the road part as the road, and the rest as the background.

Traditional image segmentation methods may be not optimal for this project. In a typical satellite image road segmentation task, the target road only occupies a small proportion of the frame; rivers, railways are similar to the road, and the segmentation is difficult even for human eyes; the roads are complicatedly bifurcated, which has very high requirements for the recognition accuracy. Therefore, our group describes one powerful technique: fully convolutional network which has already been adopted for biomedical image segmentation in the past[1] to be applied on this project's training dataset consisting 100 satellite images and groundtruth with size of 400x400 pixels(such as in overlay image Figure 1).

Figure 1: Overlay images of training data

In this project, we used Random Forest Classifier, Classical Convolutional Neural Network and U-Net model to complete the task. For the first two models, main configurations are demonstrated in the latter II.A and II.B section. All the rest illustrations are mainly related to U-Net configuration (including section II.C and section III). In section IV, it will show the results for different methods and some discussion. section V will conclude this project.

II. Models and Methodologies

A. Random Forest

Building on the provided code snippets of scikit-learn library, we develop a Random Forest Classifier (RFC).

We select the classical values of hyperparameters in our implementation of final RFC (the number of trained trees 70 and the maximum tree depth at 10,000).

Then we create patches of 16x16 pixels and compute 7 features per patch. The most basic features correspond to the mean and variance of the patch across each of the RGB channels, yielding 6 features per patch. The seventh feature is extracted based on the Hough transform technique which could find imperfect instances of objects within a certain class of shapes by a voting procedure.

Finally, we set the threshold of 0.25 as the classification boundary. If the mean of patch is larger than the threshold, we classify it as the foreground, otherwise the background.

B. CNN

We then develop a Covolutional Neural Network (CNN) model, it constructs more abstract concepts through a series of convolutional layers.

The input of the model are cropped images of 16x16 pixel size, without overlapping. In order to better filter the edges of the input and control the size of the feature map, zero paddings of size 4 are added to the edges of each patch, leading to an input size of 24x24.

We start with the classical [32, 64, 128] filters settings first, then several architectures are tested by varying the number and the parameters of the filters, dropout layers and max-pooling layers. At the end, with an additional 16-filters layer at beginning and two more 128-filters layers stacked before an max-pooling at the end, the final model has 6 convolutional layers. Because of the small size of the input, no more than 3 maxpooling layers should be inserted and a kernel size of 3x3 is chosen.

Ended by a flatten operation and a fully connected layer with sigmoid activation function, the model gives us the binary predictions of size (256,), with every pixel in each patch being predicted.

C. U-Net (Fully Convolutional Neural Network)

The U-Net model was inspired by Ronneberger et.al [1] which mainly focused on biomedical image segmentation.

This project has similar dataset and recognition task. Hence it is better to try on this model. Our implementation for U-Net model is depicted in Figure 2. This network is used to work on a few training images and does not have any fully connected layer which will limit the input size. Therefore, this model could be fit for arbitrary size images. The algorithm requires to divide one image into several small size patches in order to improve the computation performance of GPU(here we used CUDA).

This model consists two fundamental process: contracting(downward process) and expansion(upward process). The contracting steps follow the typical convolutional network. In each layer architecture (once max pooling process counts one layer), it includes two times of 3x3 convolution (with zero padding) coupled with rectified linear unit (ReLU) as activation function, and a 2x2 max pooling calculation with 2 strides to halve the size of the input. For each contracting layer, we double the channel number. The expansion steps follow the typical convolutions which followed by ReLU as activation function and one 2x2 up-convolution operation which will halve the feature channels. Then it will concatenate with the previous information. At the final output layer, one 1x1 convolution is used to map all channels to generate the possibility of being road or background for each pixel.

For each convolution operation, we added zero-padding argument to get more information of the edge. Before each convolution, we use batch normalization[2] technique coupled with dropout in each contracting layer with dropout probability of 20% to regularize the model, which will not only increase the training speed, but also get rid of overfitting to some extent.

III. Traning Process and Validation

A. Data Patches

There are two main reasons why we choose small patches for training. Firstly, there are only 100 training images which could be used to train model; Also, smaller input size will improve the computation burden.

More training data means that the NN could extract more useful information and reduce overfitting. Hence only 100 images are much lower the requirement for training. It is better to generate more data by original data. First of all, most popular method about this is to separate one whole image into several smaller same size patches. In this project, we have tried 3, 4 and 5 layers of U-Net architecture. Therefore, different minimum requirements for patch size are applied on different architecture. Then in order to not loss much information at the patch edge, it has better use overlapping technique[1] to get more information between patches. Different overlapping steps are also applied. More results will be illustrated later.

B. Data Augmentation

In addition to previous patches generation technique, it is better to add some disturbing information to make the model more robust. In this project, we choose flip and rotation operations to create more different information. In this project, we choose horizontal and vertical flips, and 30°, 45°and 60°as rotation degrees. In the rotation operation, it will enlarge the original patch by a factor of rotation angles and it will generate some empty corners which are useless for training. Therefore, we decided to recognize these corners and delete them

C. Training Process

For one Neural Network, one important this is that there are too many neutrons in the middle. Hence there is a large amount of weighting and bias to train. It will consume much time on once training. And some techniques should be applied to fit one general and powerful network.

1) *Label Transformation* Since this project is used to classify the road and background, it is one binary classification task. Therefore, for each pixel, the ground-truth label will be transformed into 0 or 1 according to 0.5 threshold.
2) *Loss Function* The most popular loss function used for binary classification is Binary Cross=Entropy (BCE). We also used this as our loss function. BCE aims to determine how far is the prediction distribution with the true distribution in binary classification problem and back-propagate the errors to U-Net.
3) *Optimizer* For optimizer, we chose Adaptive Moment Estimation (Adam) which is one stochastic optimization method and useful in this kind of NN. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters [3].
4) *Learning Rate* The Adam optimizer uses one learning rate for each model parameter, with each of this individual rates upper bounded by the initialization learning rate. The base learning rate is chosen to be 0.001. And it is better to use a dynamic learning rate adaptation in order to refine optimization by fitting the loss stagnant.
5) *Environment* We used PyTorch for U-Net definition. Since our computer did not have GPU setting, we used Google CoLaboratory as our training platform. We trained our model on single NVIDIA Tesla P100 GPU and the CUDA version was 10.0.

D. Validation and Parameters Selection

To validate different configuration of U-Net, it is better use training data to pre-train the model. In this project, we

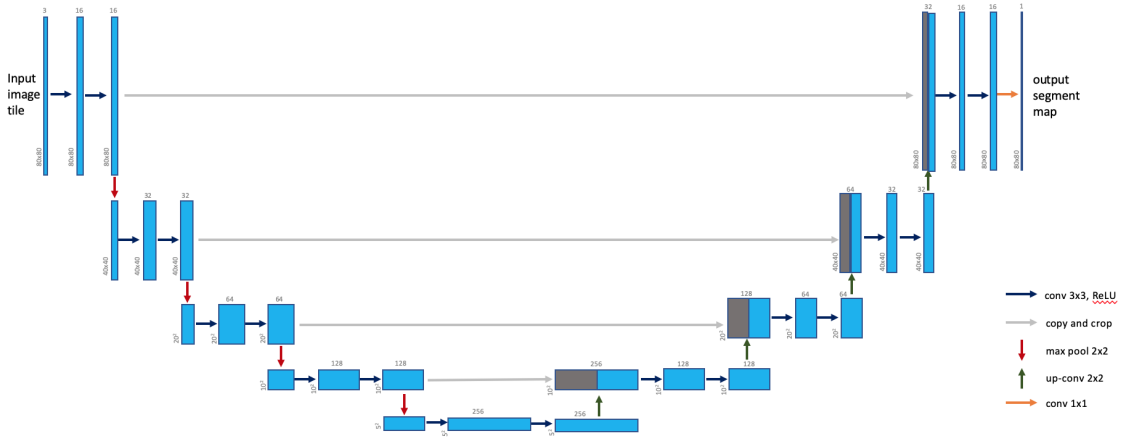

Figure 2: U-net architecture(example for 5x5 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map.The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. Grey boxes represent copied feature maps. The arrows denote the different operations.

use 80% data (80 images) as pre-training dataset and rest 20% (20 images) as validation dataset to test the accuracy for each configuration.

Firstly, it is essential to decide how many layers will be used. We chose patch size as 160 and overlapping steps as 20 as identical parameter, since 5 layers model requires the minimum 160 patch size. From results in Table I, the 4 layers model has best performance. Even 5 layers also has good performance, the more complicated the model is, the more time it will consume. Hence 4 layers model is better

configuration	Validation results
U-Net(layer = 3)	0.653202
U-Net(layer = 4)	0.647905
U-Net(layer = 5)	0.648065

Table I: Validation results for different layers

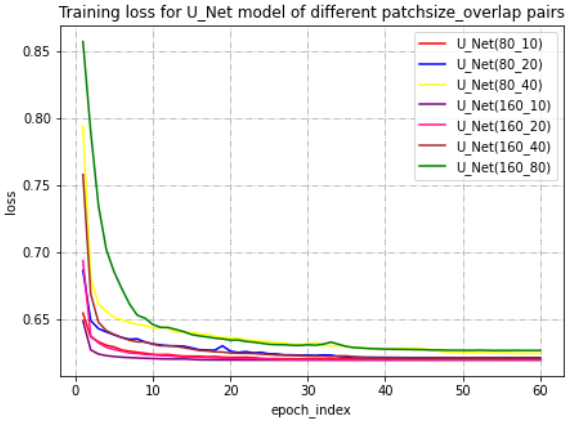

Figure 3: Validation results for different patch size and overlapping steps

After choosing the U-Net architecture, it is time to get the best patch size and overlapping steps pair for the data pre-processing. The different pairs have been test. From the results in Figure 3, the smaller patch size, the higher accuracy. And overlapping steps of 20 has the best performance. From the validation results, the more sparsely we generate patches, the higher loss we get. However, too tight steps would increase the computation burden and get no huge improvement.

After choosing the model configuration and best data pre-processing method, it is better to implement augmentation technique and find the best one. From the results, it seems the scenario only with 60°rotation has the best performance. However, in our validation process, the loss function is also used as training loss fucntion: BCE. In this project, the evaluation method is using F1-Score. Hence we compared the F1-Score between only 60°rotation and flip with 45°rotation. The latter has the higher F1-Score. Therefore, it is better to use (layers=4, patch size=80, overlapping steps=20, flip=True, rotation=45°) as our model configuration.

Augmentation	Validation results
Scenario Basic	0.638117
Scenario only Flip	0.639852
Scenario only Rotaion(30°)	0.638892
Scenario only Rotaion(45°)	0.640332
Scenario only Rotaion(60°)	0.638014
Scenario with Flip and Rotaion(30°)	0.639679
Scenario with Flip and Rotaion(45°)	0.638958
Scenario with Flip and Rotaion(60°)	0.639372

Table II: Validation results for different augmentation methods

E. Test Process

The test dataset in this project has size of 608x608 and the submission file need the prediction for small patch of 16x16 size. The test data does not need any data pre-processing and it could be fed in the trained model directly since U-Net

model. Since the road occupies about 25% in one image and the submission requires the approximation for each small patch (16x16), it is better to transform the final pixel-wise prediction into small patch prediction and the threshold is 0.25 accordingly.

IV. Results and Discussion

From Figure 3, the loss will get the plateau after 45 epoches. Hence the training epoches are set as 60. From the results in Table III, it is obvious that the U-Net models have the best performances on test dataset. And convolutional type network has higher performance compared to standard classifier. Furthermore, the U-Net with data augmentation has the best performance.

Random Forest	CNN	U-Net	U-Net 60°	U-Net best
0.73	0.81	0.879	0.885	0.9

Table III: F1-Score for different models: **U-Net**: without flip and rotation; **U-Net 60°**: U-Net only with 60°rotation; **U-Net best**: best U-Net configuration depicted in previous section

Figure 4 shows four examples of the prediction overlayed on test image of best model. From the results, it is obvious that most road could be identified correctly. From the results of three test U-Net models, we could say that the more all-round data features are provided, the more information the model could detect. The model with flip and rotation has the best score. Form Figure 4, we could see that some roads are in 45°direction, this could explain why the model with only rotaion of 60°has better performance in validation process, but is worse for test data. Therefore, due to the limitation of data quantity, only one model could not perform on all situations. For training data set, the model with only 60°rotation could fit training data set more suitably.

The high F1-Score of the best model shows that this model is not over-fitting and could be used in most situations. This is due to the overlapping patch generation technique, dropout technique, batch normalization technique and feature augmentation techniques. These methods give this model more general cases to learn. However, Figure 4 also shows that even most roads are detected correctly, some road segments shadowed by buildings, trees and cars are still hardly recognized. And if the road segment is too small in one image, it could also be difficult to identify. Hence there is no one model could predict all the information. The satisfaction depends on the acceptance level. In this project, our group thinks that this performance is acceptable.

V. Conclusion and Future Work

In this project, we used different models to predict road segments in satellite images. The U-Net model is more powerful and solid network fitting for this data set. This model has been developed before in medical field but this project shows it could also used in other segment task.

(a) Image 1 (b) Image 5

(c) Image 11 (d) Image 42

Figure 4: Overlay images on prediction data

There are also some further work we should do to improve this model. There are some post-processing techniques which could improve results after prediction. And also we could integrate dilated convolution filter in our model to improve the accuracy. However, our work is still meaningful in some segments recognition task and it could reach the accuracy with human's capacity.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," pp. 234–241, 2015.

[2] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.