COMP540 Term Project Report

Yuxuan Cui(yc110), Bidan Zhu(bz24)

April 21, 2019

Contents

1 Introduction of Project

In this competition, we are asked to create a model that can segment satellite images to identify roads. The model could detect and label the roads. We are given 10897 training images and their corresponding masks and use the train data to construct the model. In the training dataset, we have 10897 pairs of satellite images and Boolean mask images. The mask images Boolean images only contains black and white pixels. The road pixel is marked white in the mask image and the non-road pixel is marked black. Some mask pictures are all black, which means no road can be detected thorough the image. We need to take the satellite images as the input and use the model trained by the training set to create the mask of validation dataset and fill the .csv file with the RLE form of mask images. We also have 2170 test data without masks given and we can submit our results to Kaggle to get the scores of our model.

We tried two models to solve the problem. Mask RCNN and U-Net. U-Net gained a better performance on the problem with higher scores by Kaggle.

We mainly train our models and use the model to do predictions on test data on Amazon AWS platform using the p2.xlarge EC2 instances and we also use our own laptops to do some prediction which is slow.

2 Data Preparation

2.1 data analysis

This dataset contains thousands of satellite images that cover a wide range of landscapes (forest, farms, country side, cities, arid landscapes, etc...). Each image covers roughly 16 acres of land in a 256 m by 256 m cube. Each image is of size 512 by 512 resulting in a pixel resolution of 50cm. And the images are in RGB color mode with 3 channels. The size of the image is 512*512*3.

This dataset also contains a Boolean mask for each satellite image that, when applied to the respective satellite image, reveals only "road" pixels. However, each mask has been segmented by hand which has many limitations. Therefore, some masks may label "road" pixels that are actually not "road" pixels or vice versa. Moreover, small farm roads and trails are not labeled as "road" pixels.

Whats more, we find that only wide road can be marked in mask pictures. If some roads are in the field or its too narrow, it will not appear in the corresponding mask picture.

2.2 data operation

We tried some operations on the dataset to augment the performance.

By using OpenCV package, we tried several augmentation operations, including image gradients (Laplacian, Sobel weight, Sobel X, Sobel Y), edge detect. Eventually, we compute the average picture of all images, but the result seems no sense, because every image is so dim and there are so many images that the result just like nothing happens.

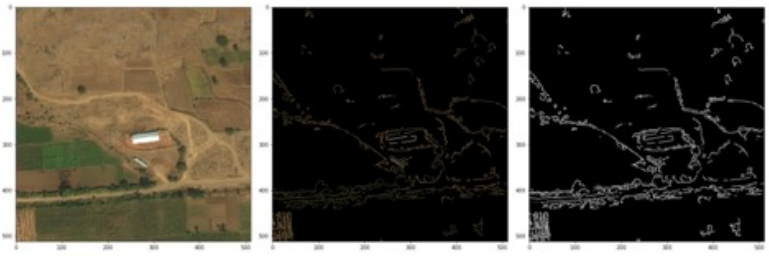


Figure 1. Edge detection of image

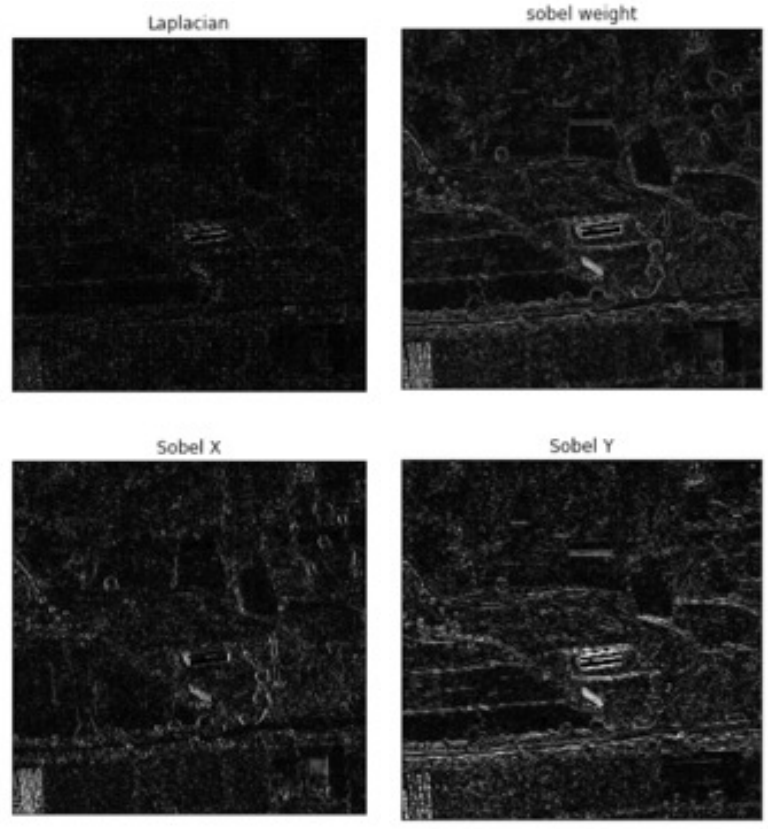


Figure 2. Gradient of image

Figure 3. Average of images

3 Model

3.1 U-Net

3.1.1 Introduction of unet

U-Net is from a paper named U-Net: Convolutional Networks for Biomedical Image Segmentation. The advantage of this model is that it can get precise segmentation base on few images. The point of this project is to train a model which can segment the satellite picture into roads and non-rods part. So, we tried U-Net. The architecture is as below.

The left part of the picture is convolutional layers and the right part is up-sampling layers. The activation value of convolutional layer will concatenate to the corresponding up sampling layer. Every 3*3 convolutions have two ReLU, then a max pool of 2*2 will applied in order to have down sample. And the size of the picture will decrease after up sampling. Then will use up sample on the symmetry part to double the row and column.

We need to have three down convolution layers and one down convolution layer includes two down convolution and one pooling. There are also three up convolution Layers, includes transpose convolutions layers. One transpose convolution layer contains one concatenate and two up sampling operation.

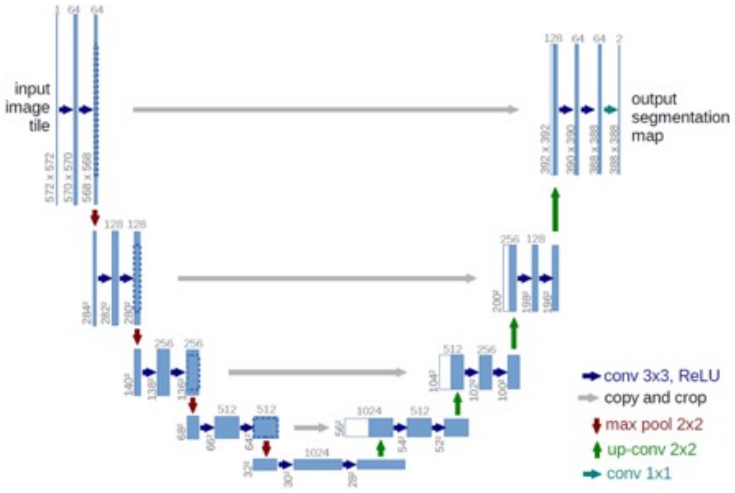


Figure 4. architecture of UNet

3.1.2 Experiments

We read the papers kernel and one of the codes on GitHub and made some modify. The implementation is based on TensorFlow backend and Keras.

Firstly, we change the picture into matrix to fit in the model and do data augmentation since U-Net is developed for small dataset, but we also think we can use data augmentation in our model to train more data and get a more accurate result. Then, the process of up-sampling and down-sampling can be implemented by using functions in Keras package. We tried different activation functions, including ReLU, SoftMax, Sigmoid and Exponential Linear. The Loss function we used is binary cross-entropy loss function, because we only have two class: road and non-road. We finally chose ReLU as the activation function and in the last layer we use sigmoid.

3.1.3 Optimization

1. In order to avoid overfitting, we add two dropout layers between convolutional layers. Adding dropout layers is a single way to prevent data overfitting by randomly choose some units and its connection to drop while training the model. We added two dropout layers to solve the problem of overfitting and it seems to work effectively in our model.

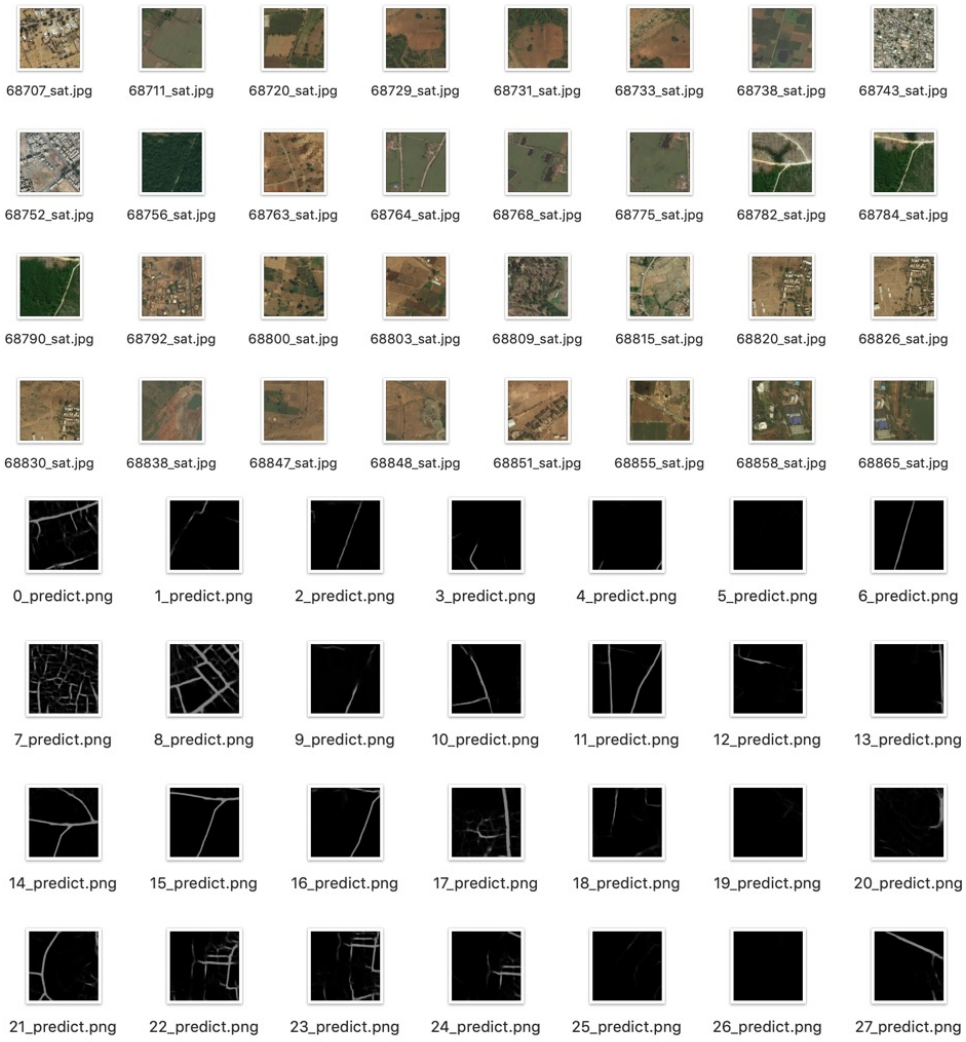
2. Every time we just save the model with the lowest loss not the model built at the last second. During the training process, in every epoch, we calculate the loss and accuracy and save the best model in a hdf5 file, which means if

the model created in the later epoch works worse than the current best one, the newly created model will not be saved.

3. We also did some tuning on the hyperparameters of the model. We tried different batch size, epoch and steps per epoch. And if the loss were too high than the current one, we will not use that model to train the test dataset to save the calculation resource.

3.1.4 Results

The results are shown below.



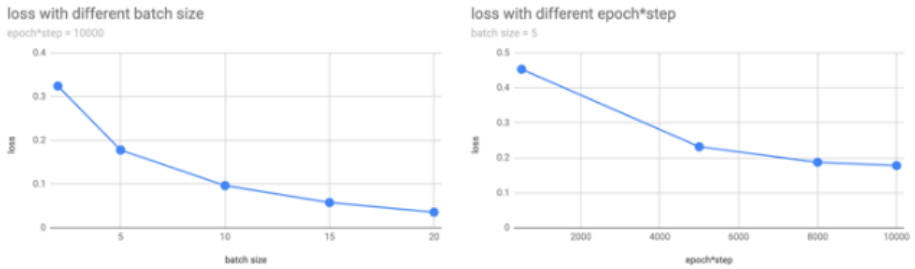


Figure 5. Results and loss with different hyper-parameters

3.2 Mask R-CNN

3.2.1 Introduction

We also applied Mask R-CNN to train the model. This model is introduced by Kaiming He and based on Faster R-CNN. Mask R-CNN is good at object detection and picture segmentation. Therefore, we choose to use this model to make semantic masks of road in satellite images.

Mask R-CNN totally includes two phases: ROI extract and production of bounding box and mask. In the first phase, Mask R-CNN use ResNet50/101as the backbone to produce feature maps. Then Region Proposal Network (RPN) is applied to get region proposals. RPN uses sliding window on the convolutional feature map and produce anchors on every position. For every anchor, RPN will figure out whether its foreground or background (judged by the value of Intersection over Union), and whether it cover target object. RPN includes 3*3 convolutional layers, one 1*1 convlutional layer for softmax classification and one bounding box regression. Therefore, the loss function contains three part: loss of classification, loss of bounding box and loss of mask. Then Mask R-CNN applied Roi Align rather than RoI pooling (in Faster R-CNN) to get fixed size RoI. In RoI pooling, because of the integer divide of the width and height to get fixed feature map, it will cause misaligned problem which means that the feature map cannot fit the position of original picture because of missing several pixels after rounding. Roi align use bilinear interpolation to find the exact position of every RoI.

In the second phase, Mask R-CNN produce masks and bounding boxes. For every image, Mask R-CNN will produce one mask for every class and bounding box locations for each proposal. Compared with Faster R-CNN, after figuring of the ROI, Mask R-CNN adds two convolution layers to build the mask.

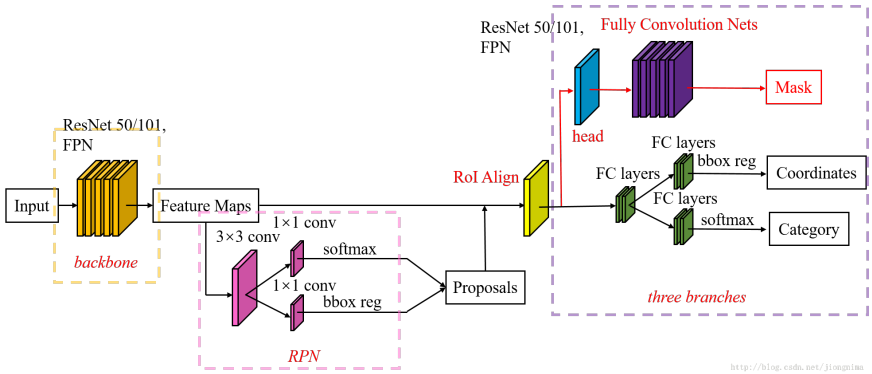


Figure 6. architecture of Mask RCNN

3.2.2 Implementation

Our implementation is based on the kernel called shapes developed by Matterport from github.

The first step is the preparation of the dataset. Firstly, I implemented function to load all images. The kernel requires me to load image path and produce an image id. At the beginning, I just put the id in the path as the required id. However, it will produce error while training. Then I found that the kernel strictly requires the id begin from 0 and must be a serial of number and We can get the corresponding mask by extracting the real id through path of image. Then we add one class to the dataset because we only need this model to detect road. The model also needs us to implement functions to load image and mask for the dataset. As for load mask function, we firstly put mask has a shape of [512, 512, 3] into the dataset, but it will cause error. Then I tried to convert the mask into array includes true or false whose shape is [512, 512] but it still caused an error. Then I look through the code in the model and found that the mask we should provide should in the format of [width, height, instance count] and we dont need instance count in this project, so we reshape the mask into [512, 512, 1]. For the kernel requires images dim = 3, we apply gray2rgb function to convert image whose dim = 2 and segment the array for image whose dim = 4.

Secondly, we load COCO dataset to initialize the weight of model. COCO is a large-scale object detection, segmentation, and captioning dataset, so its great to start with. Then we have a two-step train, one for header layer and one for all layers. During the train, it will output loss value of every index and the time of training. Finally, we transfer results into RLE encode and output results to a csv file.

3.2.3 Train process

Due to the necessary of GPU, we cant use our own MacBook to train the model, so we do the train on the AWS platform.

We use ResNet101 to extract the feature and RPN in function $rpn_graph()$ to find regions contains object. Then for train, it has two steps: firstly, only train for the head layer with learning rate 0.0001 to avoid overfit. Then train the whole layers of the model with learning rate 0.00001. The loss function being used here is per-pixel sigmoid and binary loss function. For each ROI, $loss = loss(class) + loss(box) + loss(mask)$. The loss for class is per-pixel sigmoid and loss for mask is an average binary cross-entropy loss.

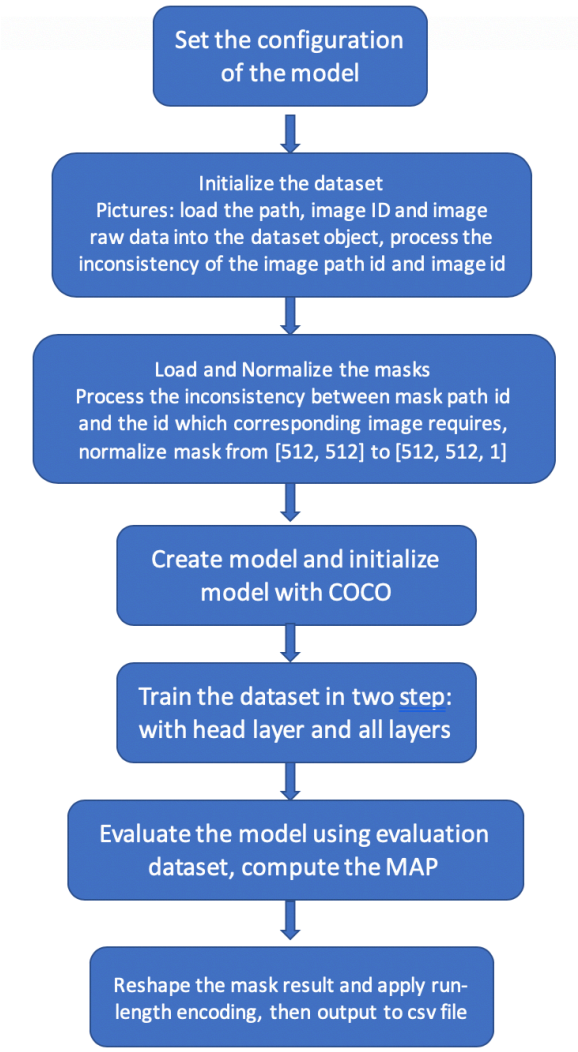


Figure 7. Workflow of Mask RCNN

3.2.4 Parameters

I tuned TRAIN_ROIS_PER_IMAGE from 256 to 200, because roads are not small objects, so theres no need to have so many ROIs on each image. Decrease the number of ROI can decrease the time of training. Because the edge of roads is not clear, so I tuned DETECTION_MIN_CONFIDENCE from 0.7 to 0.5 to let more anchors have the results. I tuned MAX_GT_INSTANCES from 100 to 256. As for IMAGES_PER_GPU, I tried 8 and 4 but both of them will cause out of memory of GPA, so I eventually set it as 2. As for the backbone, we use resnet101.

As for step and epoch, I tried { STEPS_PER_EPOCH = 50, VALIDATION_STEPS = 50, epochs(for header layer) = 3, epochs(for all layers) = 7}, { STEPS_PER_EPOCH = 150, VALIDATION_STEPS = 50, epochs(for header layer) = 70, epochs(for all layers) = 80}, { STEPS_PER_EPOCH = 1000, VALIDATION_STEPS = 50, epochs(for header layer) = 25, epochs(for all layers) = 25}.

3.2.5 Result

The default value of class loss and bbox loss is 1, total loss is 3.

As for { STEPS_PER_EPOCH = 50, VALIDATION_STEPS = 50, epochs(for header layer) = 10, epochs(for all layers) = 20}, when training for header layer, total loss are { 5.5715, 5.1760, 4.1336 }, class loss are { 0.2151, 0.0664, 0.0573 }, bbox loss are { 4.2886, 3.9515, 3.1044 }. We can find that even the loss is really high due to few steps per epoch, the loss decreases significantly as the increase of epochs. When training for all layers, total loss is between { 4.0274, 2.0449 }, class loss is between { 0.0489, 0.0309 }, bbox loss is between { 2.9989, 3.9515 } and loss decreases from the beginning epoch to the end epoch.

As for { STEPS_PER_EPOCH = 150, VALIDATION_STEPS = 50, epochs(for header layer) = 70, epochs(for all layers) = 80}, when training for header layer, total loss is between { 4.5456, 2.0234}, class loss is between { 0.0909, 0.0094 }, bbox loss is between { 3.2286, 1.3145 }. When training for all layers, total loss is between { 2.7586 , 1.4831 }, class loss is between { 0.0093, 0.0063 }, bbox loss is between { 2.0234, 0.7634 } and loss decreases from the beginning epoch to the end epoch.

As for { STEPS_PER_EPOCH = 1000, VALIDATION_STEPS = 50, epochs(for header layer) = 25, epochs(for all layers) = 25}, when training for header layer, total loss is between { 2.8207, 1.8173 }, class loss is between { 0.0353, 0.0083 }, bbox loss is between { 1.9317, 1.3324 }. When training for all layers, total loss is between { 1.8173, 1.1003 }, class loss is between { 0.0083, 0.0024 }, bbox loss

is between { 1. 3324, 0.4723 } and loss decreases from the beginning epoch to the end epoch.

As for simple image, the result seems good. However, for the complex images, the result seems not good.

Figure 8. result of Mask RCNN

Figure 9. result of Mask RCNN

3.2.6 Analysis

Before the poster presentation, I regarded the bad result as the unsuitable of the model applied on the project. Because when I google for other peoples result of Mask R-CNN applied on different projects, I found that their results are also consists small circles when segmenting lines and area such segmentation of water channel. However, after chatting with Eleni on poster presentation, she told me that I have so many epochs that may cause overfitting. When I observe the result of loss value, I find that the loss slightly increases in the last few epochs. An applied of early stop can help to solve this problem. Whats more,

the increase of step from 150 to 1000 can make the loss decreases from 1.5 to 1.1.

3.3 Compare of U-Net and Mask RCNN

Mask R-CNN is based on Faster RCNN and FCN. U-Net is based on FCN. They both works well on image segmentation but they performed differently on the given problem of the Kaggle competition.

From the result, Mask R-CNN works not good as U-Net on this dataset for the performance is uneven. Because as for Mask R-CNN, it will firstly find the ROI and then produce masks on each ROI. So, there could be many circle masks on the road instead of a consistent long mask. So, Mask R-CNN doesnt work well for segmentation of complex long shapes just like road. However, even Mask R-CNN cannot produce the precise mask, it can do the right thing on object detection.

U-Net performs better in this case. Since the data has simple semantics and the structure of the data is not complicated, which is suitable for U-Net model. U-Net first do down sampling and then up sampling, which acts like the compress the local feature first and then unzip it to identify the boundary of the images. The boundary in the images is the road that we are going to identify.

4 Platform Choose

4.1 Google Colabotory

We tried Google Colabotory firstly. The advantage of this platform is free and easy to use. After opening the page, people can straightforwardly upload the. ipynb file and execute it. However, the server will stop automatically if user do nothing. So, its not suitable to do long time training. Whats more, in my opinion, its not flexible on the file system compared with AWS.

4.2 Amazon Web Service

Due to the flexibility of AWS, we finally use AWS platform EC2 to train our model. with following instance configuration: Deep Learning AMI (Ubuntu) Version 22.0, p2. Xlarge instance. The advantage of this platform is that we can use Jupiter notebook on the local host and run it remotely. This instance has 1GPU which accelerates the training process a lot. However, we also met and solve problems and get lectures during the usage of the AWS. Firstly, the default of the storage of instance is only 60GB, which is really small, so we should modify the configuration of dis storage at the beginning of launch the instance. Secondly, after stopping the instance, if we start it again, the public DNS of the instance will change! Therefore, we cant connect to the AWS by using the

previous IPv4. Thirdly, the most serious problem we suffered is the disconnect of AWS. When the network as a little unstable, the connect to the AWS will get lost. Many days when I get up and check the result of the training, I can only find that due to the connect lost, the process of training just terminated, and I have to re-train it. Thats why we didnt tune the parameters of Mask R-CNN a lot. Even in one week, I trained the model for 7 times and only one time I successfully get a result without the loss of connection. I hope there can be some methods to bypass this problem because its really painful and wasting time.

5 Lesson

As for U-Net, the first lesson we learned is that the environment of doing experiment is very important. We first search for u-net on GitHub and found a repository with more than one thousand stars. We follow the instructions and make the code work for our project by making some changes on the parameter of input and output. But when we tried to run the code, we find every time we tune the parameters to do the training process, it will take about 30 hours to finish. We make progress very slowly. However, we then turn to use Google lab and the time reduced a lot.

The second lesson is that we can try part of the data to evaluate new parameters. Once we change the value of one of the parameters, we can try on a subset of the whole dataset. If the result is obviously wrong from expected, we can detect the error fast. We do not need to train the whole dataset using that parameter which will save time and calculation resources.

As for mask R-CNN, we should consult TAs about the possible reason for bad result after googling the problem. Different dataset has different situation, other projects parameters might not suit our project. Therefore, its wrong to charge the bad result upon the model, but our inappropriate parameters. Even though we didnt get a good precision result by using Mask R-CNN, we still gain a lot through the training of this model. We know the workflow of most powerful object detection model and figure out a way to modify the kernel developed by Matterport to make the kernel fit for our project. Whats more, the process of tuning the parameters of the model can also promote our understanding of the model, such as tuning the number of anchors per image can affect the result. We also gain a lot of knowledge of computer vision during data preprocess and postprocess and know methods in OpenCV which can help to augment the images. However, we just should have submitted the results not so late because we didnt notice the limitation number of submissions on Kaggle and some results cannot be upload due to the limitation. Also, it will be better to plot all loss values after training. We only observe on the loss information, but its not straightforward to get the information of loss change and overfit.

6 Conclusion and submission

During the term project our team, Yuxuan Cui and Bidan Zhu, after analyze data and read paper about image segmentation, tried two different CNN models to solve the problem: Yuxuan used Mask-RCNN and Bidan used U-Net. After building the model, doing data preprocessing(data augmentation), tune the hyperparameters and doing data postprocessing, we finally got a score of 0.563 from Kaggle.

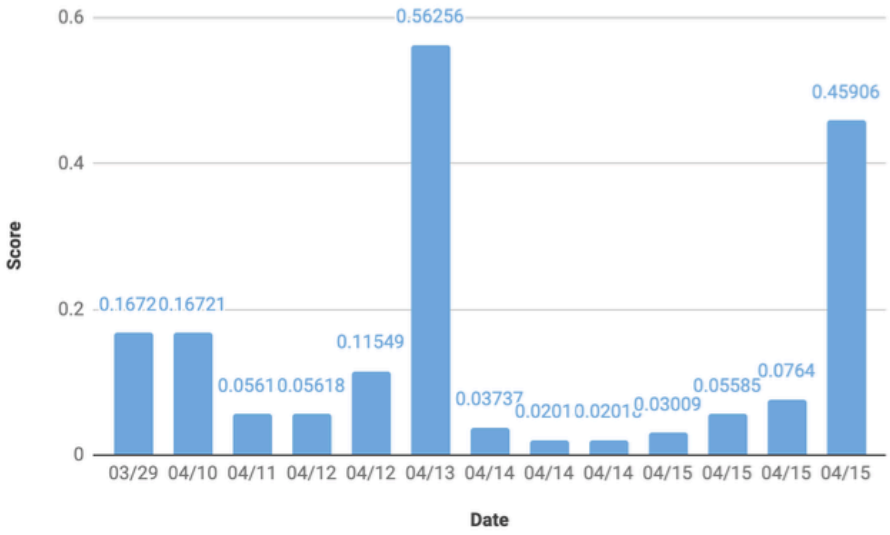


Figure 10. timeline of submission

At the very beginning of the project, we have some trouble doing the Run-Length-Encoding of the result data. Although the result pictures seem to be almost correct, the Kaggle score is very low.

We then realized that we have mistaken the output size and made the correct encoding to get the true scores of the result data.

7 Future

We could ensemble the results of U-Net and Mask R-CNN. A nave though is just combine the mask. For example, we can do the or operation on every pixel of the mask, or we can give a weight to the pixel of mask from U-Net and the pixel of the Mask R-CNN, then apply a threshold for the pixel to get the output mask. Whats more, as for Mask R-CNN, Im really glad to investigate the effect of every parameters by using control variate method.

8 Reference

[1]Unet introduction: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/
[2]Unet repository: https://github.com/zhixuhao/unet
[3]Mask R-CNN repository: https://github.com/matterport/Mask_RCNN
[4]Mask R-CNN introduction: https://medium.freecodecamp.org/mask-r-cnn-explained-7f82bec890e3
[5]Python Image Generators: https://www.kaggle.com/abhmul/python-image-generator-tutorial
[6]Run Length Encode Script: https://www.kaggle.com/timnonet/run-length-encoding-script-python2