CS-433 Project2: Road Segmentation for Aerial Images

Liangze Jiang, Ruizhi Luo, Danyang Wang
École Polytechnique Fédérale de Lausanne, Switzerland

***Abstract*—This project aims to tackle a road segmentation task by patch-wisely distinguishing between road and background in a set of aerial images. To this end, we show convolutional neural networks, specifically U-nets, can produce satisfying results after data augmentation, dealing with imbalanced data, post-processing and tuning hyperparameters. Finally, our U-net model achieves a 0.892 F1-score on the AICrowd test set.**

I. Introduction

Image segmentation comes to be a complicated task in computer vision, which aims to classify the object on the image pixel by pixel. Similarly, the goal of this project is to partition aerial images into road class and background class. Due to the emerging of computational capability and technologies, convolutional neural networks(CNNs) become dominant in image segmentation task in recent years. In this project, we are going to explore CNNs performance on road segmentation task.

Generally, this report outlines the baseline we choose to handle the task, as well as the models and methods we attempt to improve the performance depending on the characteristics of dataset and results. Finally, summary and discussion are given.

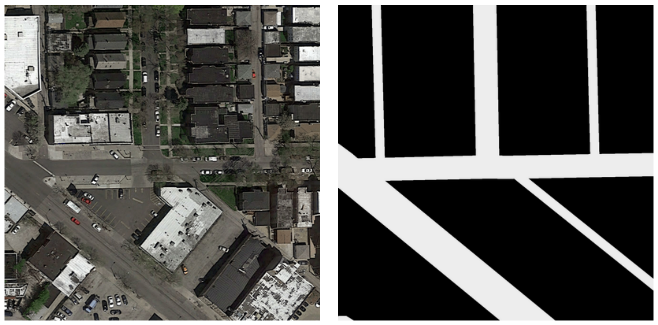
II. Data Exploration

A. Dataset

The training dataset contains 100 aerial images which are mostly shooted in urban area. Each image is of the size 400x400 pixels and has 3 channels. The ground truth of training dataset is also of the size 400x400 pixels but only has 1 channels, each pixels has the value of 0-1 to indicate the pixel is belong to road or background class. In Figure 1, an instance of training data and its ground truth are given. The test dataset consists 50 aerial images which have the same style as training dataset. The test images also have 3 channels but each is of the size 608x608 pixels.

After prediction, every images is cropped to a series of 16x16 pixel patches, then the entry value of patches are averaged, if the mean is beyond 0.25, then the entire patch is labeled to 1, i.e., the road class.

From Figure 1, some of the difficulties of this task lie on the complexity of images. Firstly, some of the road area is covered by trees and cars, this gives a great interference of detecting road pixels. Secondly, due to the characteristics of urban area, the surface of roads look very similar to the roofs of buildings or the parking lots. Thirdly, some relatively narrow roads show up in the landscape. Combining the first and second point, these road can be very hard to identify. Another crucial point is that the dataset in this task is small, which adds more difficulties to handle the task.



(a) Image (b) Groundtruth

Fig. 1. Example of Training Dataset

III. Evaluation Metric

Generally, the area of road class is less than the area of background class, which makes the dataset a imbalanced one. Therefore, accuracy is not the best choice to make a evaluation, because a naive predictor that predicts all pixels as background can result in a considerable accuracy.

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} \tag{1}$$

So, in this project F1-score metric is considered as the evaluation metric. F1-score in equation 1 is the harmonic mean of the precision and recall, where precision is the fraction of real road pixels among the model-detected pixels and recall is the fraction of the total amount of road pixels that were actually detected. Because this metric takes both false positives and false negatives into account, it is better to evaluate the performance on our imbalanced data.

IV. Models and Methods

A. Baseline

Fully Convolutional Network architecture has shown SOTA results in semantic segmentation. Being one of them and firstly designed for biomedical image segmentation, U-Net was shwon a great generalization on other segmentation tasks. This model is proposed by Olaf Ronneberger et al. in the papar *U-Net: Convolutional Neural Nets for Biomedical Image Segmentation*[1] and became the champion model of the contest ”Kaggle Carvana Image Masking Challenge” at that time. Also, the original model has only 31M parameters, which fits our situation of lacking computational power. Hence, we consider this model as our baseline model.

U-Net is a classic fully convolutional network with a symmetrical encoder-decoder structure. There are three main part in this model: encoder, bottleneck and decoder. The structure of the U-Net model is shown in Figure 2.

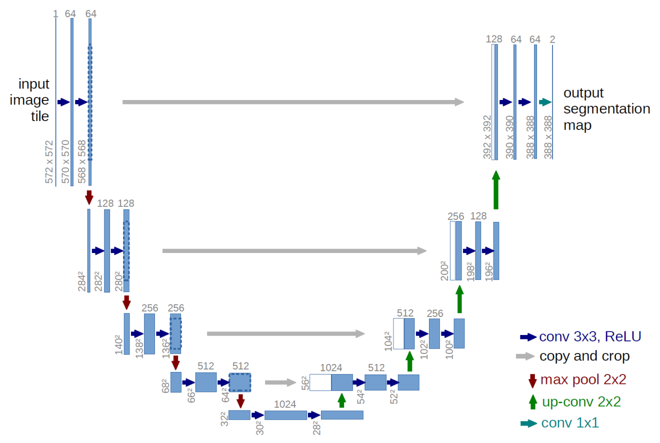


Fig. 2. U-Net model in the original paper.

Encoder is used to extract the feature of images and it has four contraction steps, each step contains two 3x3 convolutional layers with a activation layer behind. There is a max pooling layer at the end of each contraction step, which halved the size of the image and doubled the feature of the image.

Bottleneck contains two 3x3 convolutional layers with a activation layer behind each of them. At the bottleneck the size of the feature map is the smallest and the number of feature map channels is maximized.

Decoder recovers image from small feature maps and it has four expansion steps corresponding to the Encoder structure, in each step, the image is first up-sampled and the size of image is multiplied by 2. After that the image is concatenated with the corresponding contraction result of the encoder and then be convoluted and activated twice. the image size is doubled and its number of feature is halved in each expansion step.

We first chose ReLU as the activation function since it can avoid gradient vanishing problem and its calculation consumption is small.

B. Dealing with Imbalanced Data

During our training process we found that binary cross entropy loss is not the best indicator of F1-score, sometimes when the loss went down, the F1-score did not increase. Recall that our data is imbalance, we want to optimize directly on the F1-score or its approximation form. To that end, we use Jaccard index(IoULoss) in equation 2, which punishes wrong classifications on both background and road class, to develop a cost function.

$$Jacc(y, \hat{y}) = \frac{y \cap \hat{y}}{y \cup \hat{y}} = \frac{TP}{TP + FP + FN} \tag{2}$$

Because our predicted mask and ground truth is not boolean but a float type, we can directly use one minus equation 2 as loss function, but the gradient exploding can happen. Therefore, we add a smooth term to the equation, the final loss is as equation 3.

$$Jacc(y, \hat{y}) = 1 - \frac{smooth + \sum y\hat{y}}{smooth + \sum y + \hat{y} - y\hat{y}} \tag{3}$$

Implementing this cost function further boosts our F1-score for 4% on test set from 0.809 to 0.851. We also implement loss function for F1-score in the similar way, but the results are very close.

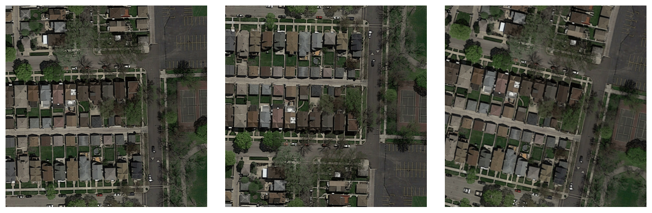
C. Data Augmentation

Machine learning models perform better with the more training data they have. Hence, augmentation of images on our small training set is implemented. In the project, we apply two different ways, flipping and rotation to enrich the training set.

Flipping an image adds horizontal and vertical symmetries along two axes. Rotations of the images means that the model is dealing with roads that travel in various directions and not only parallel and perpendicular orientations. Here, we rotate the images between -90 degrees and +90 degrees.

Since a rotation of an image results in a larger image with empty corners, we rotate the image based on its center, discard the outside part and fill the corner with symmetric mode(i.e., reflect the neighbor part including the boundary to the corner). Example of augmented images are shown in Figure 3.

As a result, we independently use horizontal flip, vertical flip and random rotation on our training images and each doubles our training dataset, thus we end up with 4 times of training set. There are other useful ways such as shear, shifting that we can use or even combine several methods to further enlarge our dataset, but considering our computational power, we did not use in this project. The data augmentation part further gives us 3% improvements on test set.

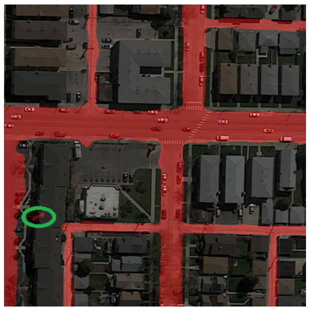


(a) Original (b) Flipping (c) Rotate & Fill Corner

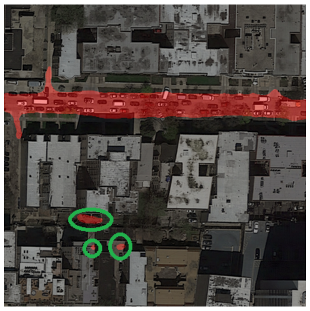
Fig. 3. Example of Flipping and Rotation

D. Post Morphological Processing Block

We observe from the predicted image mask that sometimes the model can be confused by a small series of pixels that have similar characteristics with road, this leads to a small blur on prediction mask, for example in Figure 4. Therefore, we make our attempts about this using image morphological transformations.



(a) Example 1



(b) Example 2

Fig. 4. Noise Patches in Prediction

1) Morphological Opening: Firstly, we try to handle it by morphological opening, which is the combination of erosion and dilation. The in equation 4.

$$M \circ K = (M \ominus K) \oplus K \tag{4}$$

where $\ominus$ and $\oplus$ are erosion and dilation, M is predicted mask and K is structuring element.

This process can remove small foreground noise without change the shape of main roads. We use a kernel of 64x64 ones matrix and perform opening on our predicted mask, but this did not give great results. It is because some roads are as narrow as the noise patches, when we use a suitable kernel size, these narrow road would be eliminated together with noise patches.

2) Thresholding Noise Patches by Area: Secondly, we figure out that one difference between narrow roads and noise patches is the area of pixels. Although narrow roads is slim, they tend to have larger area than noise patches. So we first find contours of every objects on the mask, then compute the area on and inside the contour, if the area is beyond our threshold, then we eliminate them. Also, We are aware that narrow roads can be inconstant patches because of the limitation of our model, thus we carefully tune the threshold pixel area as 600 and remove the small objects below this area, then we got a improvement from 0.883 to 0.887. Algorithm 1 shows the entire process.

```
Algorithm 1: Eliminating Noise Patches by Area
input :  $M_{pre}$ ← predicted mask
output:  $M_{pro}$ ← processed mask
$M'_{pre}$ ← pixel value under 0.5 to 0;
$M''_{pre}$ ← float mask to boolean;
Find contour $O_i$ for each closed objects;
$O$ ← closed objects contours in $M''_{pre}$;
for $O_i$ in $O$ do
    Calculate area $A_i$ inside $O_i$;
    if $A_i > 600$ then
        Pixel $P_j \in O_i \to 0$
    else
        Do nothing;
    end
end
```

V. Model Selection

Due to the lack of computational power, we cannot tune the hyper-parameters by formal cross-validation. And one sufficient way is to tune the parameter one by one to find a good combination. Generally, we tune the parameters on dropout, batch normalization, activation function, learning rate and its decay as well as the network's architecture. The final chosen parameters are in Table I.

TABLE I
FINAL TUNED HYPERPARAMETERS

Methods or Parameters	Value
Learning Rate	Start from 5×10^{-4} and decrease dynamically
Batch size	2 considering the computational power
Activation Function	Used LeakyReLU as it slightly improves the results (0.01-0.02)
Batch normalization	Yes as it increases stability
Dropout	Yes as it increases the generality of model
Jaccard Smooth term	50

A. Batch size

Considering the computational power and specific semantic segmentation task, we set batch size equals to 2. This further influences our choice of learning rate, because when batch size is small, a large learning rate may cause fluctuation.

B. Learning Rate

Learning rate can influence tremendously on the convergence. When the learning rate is too large, the loss may fluctuate, and a too small learning rate may cause model convergence slowly. Therefore, we reduce learning rate when the training loss has stopped decreasing, by $pytorch$ provided dynamic learning rate scheduler. We initially chose an appropriate learning rate(in our experiment, 0.0005) and if loss cannot go down for 5 epochs, we divide learning rate by 5. This makes our model converges faster than decay in every fixed epochs.

C. Dropout

We use dropout on our model after activation function as it is a common regularization method. But dropout with a large value(especially in CNNs) may cause a slow convergence and this is consistent with our observation when we trained ours. Hence, we tuned the probability of dropout and set $p = 0.2$ as it makes model generalize well without losing too much convergence speed.

D. Jaccard Smooth term

The smooth term in jaccard loss function is set to an appropriate value in order to avoid either gradient explode or gradient vanishing.

E. Others

We also tried to enlarge the kernel size of convolutional layer to increase receptive field or tune on the number of feature maps of each layer to change the model capacity. Unfortunately, no great results happened.

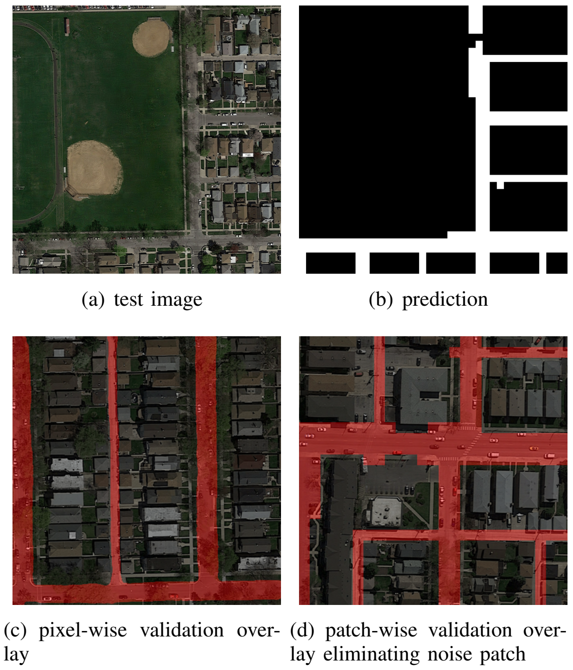
VI. Implementation Details

We built our model by $Pytorch$ and trained it on $GoogleColab$, which provides a random free GPU(Nvidia K80, T4, P4 or P100) for us. We perform stochastic gradient descent on our model, using Adam optimizer, where we set the beta coefficients as default(0.9, 0.999).

In addition, the U-Net model in original paper add no padding on each convolutional layer, resulting in a different size between input and output image. To ensure a same size, we set padding equals to 1 on every convolutional layer.

VII. Results

After tuning the parameters, we finally got a 0.892 F1-score on AICrowd. Figure 6 indicates that our model generalizes well on validation set. Figure 5 shows an example of segmentation results of testing image, indicating that U-Net can achieve fine results on this task. However, there are several disadvantages such as the model sometimes cannot detect very narrow roads and it cannot detect diagonal roads very well even though the data is augmented by rotation. Table II shows the improvement of different approaches on F1-score. The baseline did not use data augmentation for the purpose of saving training time and get intuition about the performance on small dataset.


(a) test image (b) prediction

(c) pixel-wise validation overlay (d) patch-wise validation overlay eliminating noise patch

Fig. 5. Example of prediction on validation and test set

TABLE II
Improvements on Baseline

Methods	F1-score on test set
Naive Baseline	0.712
U-net with dropout	0.809
U-net + Jaccard Loss	0.851
U-net + Jaccard + Augmentation	0.883
U-net + Jaccard + Augmentation Post morphology block	0.887
Final tuned Model(best model)	**0.892**

VIII. Summary and Discussion

We use U-Net to build a road segmentation model with considerable F1-score, after using proper loss function, data augmentation, post morphological processing, regularization and tuning parameters.

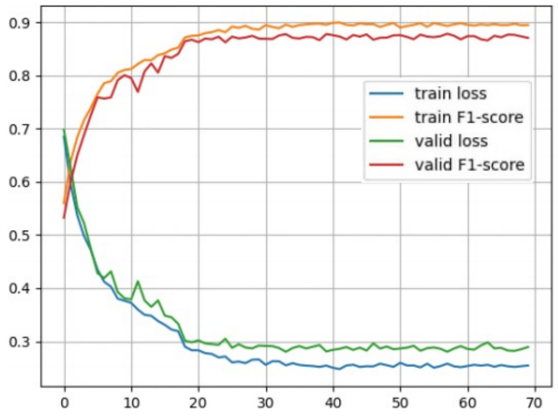


Fig. 6. Loss and F1-score on train and validation set

Further works can be done in the following aspects: Firstly, if computational power permits, one can further enlarge the dataset using various transformations such as shear transformation and shifting, or one can find more data to improve the model. Secondly, the post morphological opening block still can eliminate some real road classes by fault, because the model can predict some narrow road discontinuously as small pieces. Thirdly, the hyper-parameters can be better tuned with proper cross validation. Last, we think that U-Net does not perform best with different input size in training and testing stage, and this can be fixed by further approaches.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.