Summary Report ----
Background Subtraction Using Deep Learning (Part I)

Contents

Part I A brief recall 1
Part II The proposed CNN model 2
2.1 The method in the reference paper 2
2.2 My model 2
2.2.1 Generate background image 2
2.2.2 CNN for background subtraction 3
Part III Experiment details 3
3.1 Dataset 3
3.2 Training 4
3.3 Challenges 4
3.4 Result 5

Part I A brief review

In the last report, I proposed a framework of vehicle counting based on deep learning (figure 1.1).

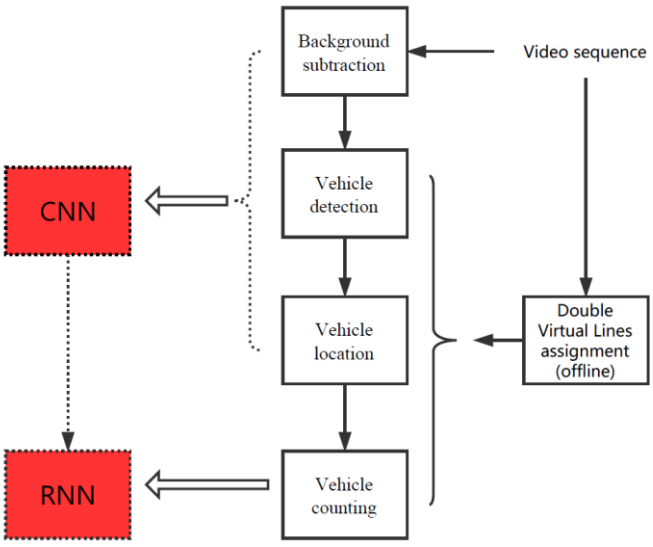

Figure 1.1 A framework of vehicle counting based on deep learning

During week 3-4, I focused on the first part, i.e. background subtraction based on deep learning.

This report is organized as follows: Part II describes the proposed CNN model for background subtraction. Part III talks about my current experiment and some main challenges I've faced.

Part II The proposed CNN model

2.1 The method in the reference paper

My work was inspired by [1]. Let's go through it as a beginning.

Figure 2.1 shows the method proposed in [1]. First, the author generates the basic background model based on some traditional algorithms (SuBSENSE [2] and Flux Tensor [3]). Then the author constructs a CNN model, followed by some post-processing methods, to get the final foreground mask.

Instead of feeding the whole image to the CNN, the author extracts none-overlapping patches and processes them one by one, during both training and testing. What's more, the network only has 3 convolutional layers and 2 fully connected layers. It is not deep enough.

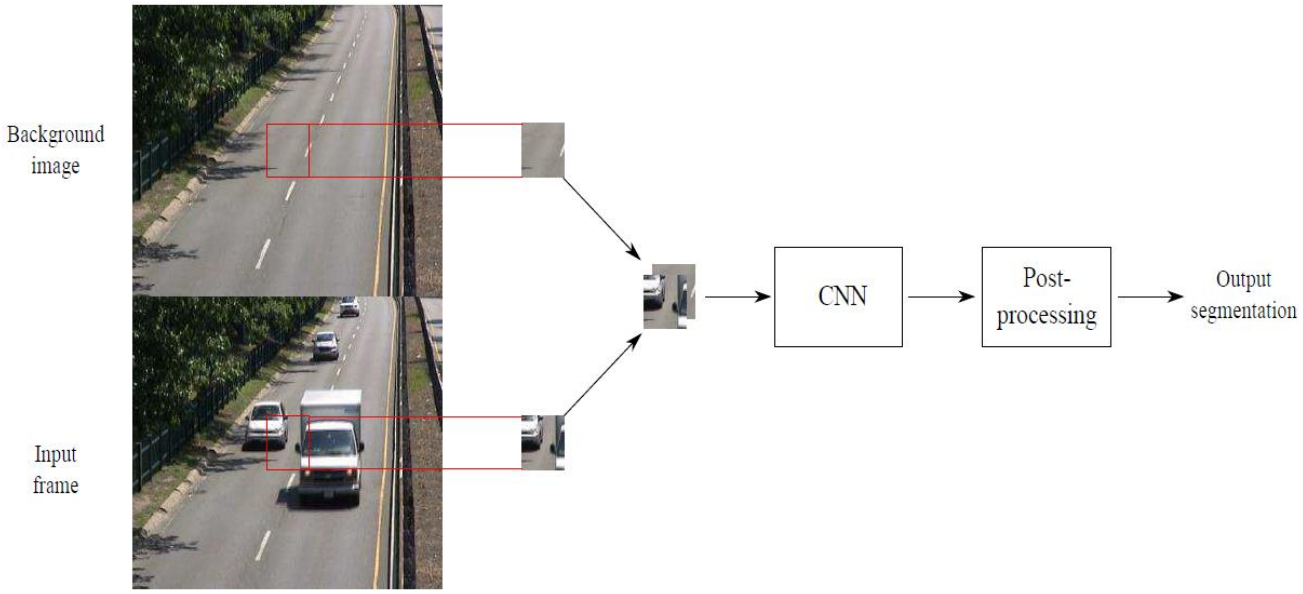

Figure 2.1 The method proposed in [1]

My model differs from [1] in the following 2 aspects.

1. I use the pre-trained 50-layer ResNet([4] [5]) for feature extraction. Hopefully, deeper network will be more powerful, and fine-tuning pre-trained models makes it possible to train very deep CNN.
2. I feed the whole image rather than small patches to the CNN. It is known that spatial relation is essential for CNN to extract features, and extracting patches from one image and processing them independently will eliminate some important spatial information.

2.2 My model

The framework is quite similar to [1]. It consists of two stages: generating background image and feeding forward through CNN.

2.2.1 Generate background image

Given one frame from the video, I get the background image using SuBSENSE[2]. [1] uses the combination of various algorithms to get a robust background model. In my model, I simplify this stage because a strong CNN model should compensate the imperfections of the background model.

In this stage, I first generate the background image. Then the current frame and the background image are resized to 321×321. Finally, the two resized images are stacked to form a 321×321×6 data cube, which is used as the input of the neural network. (Figure 2.2)

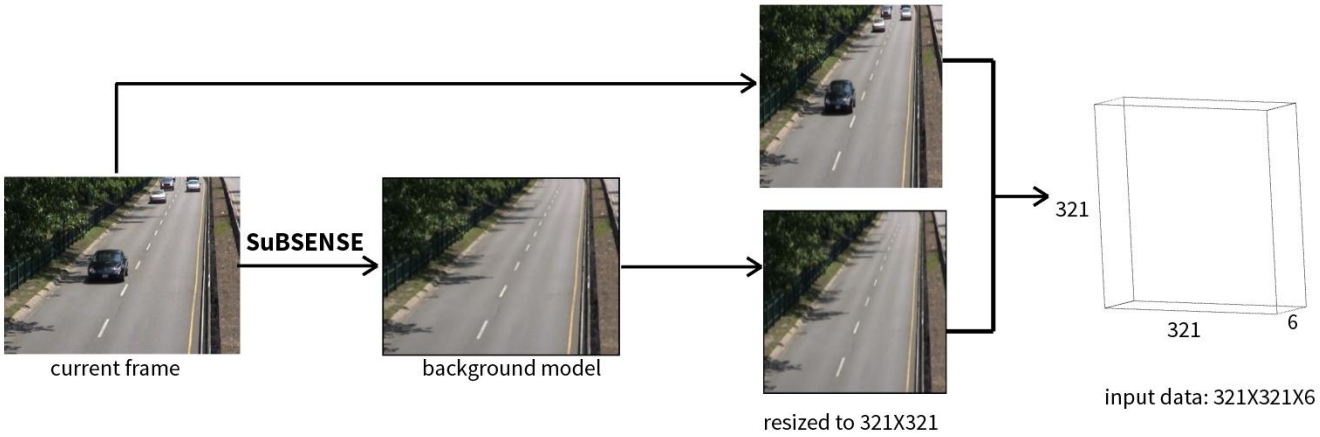

Figure 2.2 Generate background image and construct the input data for CNN

2.2.2 CNN for background subtraction

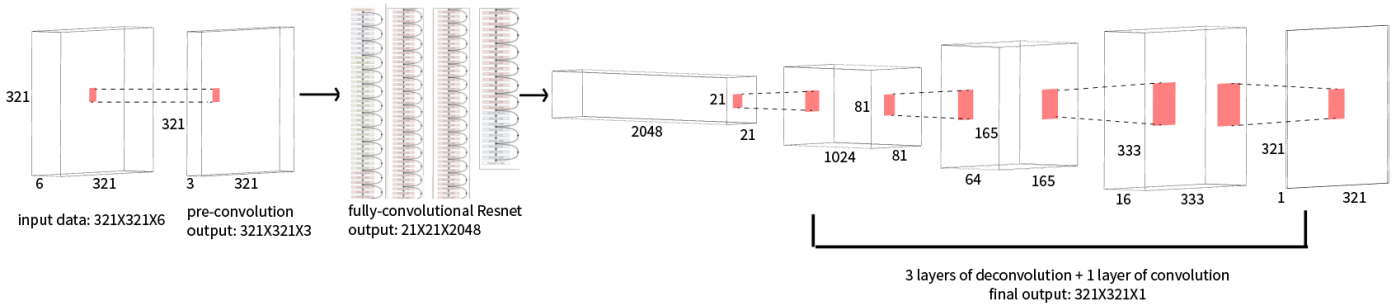

Figure 2.3 Diagram of the proposed CNN model

Figure 2.3 shows the CNN model. First, convolutional operation with 1×1 filters is used to map the input data cube into a 3-channel feature map. This is because ResNet only takes input that has 3 channels. Second, the 3-channel feature map runs through fully-convolutional ResNet for feature extracting.

Here comes a problem: the output of the convolutional operation has smaller size than the input. How to compute the loss between the output feature map and the ground truth image (the former has the size of 21×21×2048, while the latter has the size of 321×321×1)?

According to some state-of-art work on semantic segmentation ([6][7]), deconvolutional layers (also called transposed convolutional layers) prove to be useful in pixel-to-pixel estimation. So I add three deconvolutional layers to up-sample the feature map.

Part III Experiment details

3.1 Dataset

I use part of CDnet 2014 ([8]) to train the model. I modify the dataset so that is fit the model better. You can get the modified dataset from [here]. (If you cannot access Google Drive for some reason, please contact me by emailing yanyiqinwpu@gmail.com)

1. Category “intermittentObjectMotion” is removed because the foreground objects are intermittent in a video, which is not the case I will consider in traffic videos.
2. In category “shadow”, sub-directory “copyMachine” is removed for the same reason as 1.
3. In category "thermal", three sub-directories, “library”, “diningRoom”, “lakeSide” are removed for the same reason as 1.
4. In category “lowFramerate”, only one sub-directory, “turnpike_0_5fps” is used.
5. In category “nightVideos”, two sub-directories, “bridgeEntry”, and “busyBoulvard” are eliminated because the ROI region is so small that it is meaningless to use them for training (Figure 3.1).
6. In category “PTZ”, the background scene changes significantly in a video due to camera rotation, so

I manually select the part in which the camera is stable. They are: frame 1200-1462 and 1805-1980 in “intermittentPan”, frame 840-1045, 1160-1384 and 1460-1549 in “twoPositionPTZCam”. Sub-directory “continuousPan” and “zoomInZoomOut” are removed.

7. Category “turbulence” is removed.

Among the chosen categories, “highway” is selected as test set.

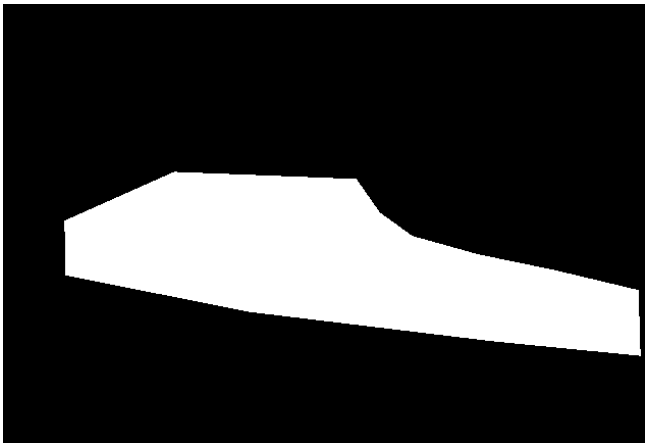

Figure 3.1 ROI region in the category “bridgeEntry”

3.2 Training

First, I run through the whole dataset to generate all the background images. These background images together with the origin frame images will be used to train the model.

As for the CNN model, I implement it based on Tensorflow. The ResNet model can be implemented based on TF-slim[9]. I train the network with mini batches of size 40 via Adam optimizer. The training runs for 2000 iterations, for the first 100 steps, the learning rate is 1e-3; for step 100-500 the learning rate is reduced to 1e-4; then the learning rate is reduced to 1e-5 and kept unchanged until the end of the training. For the lost function, I choose the classical binary cross-entropy loss.

Boundaries of foreground objects and pixels that do not lie in the ROI are marked by gray value in the ground truth segmentations (Figure 3.2). These pixels are ignored.

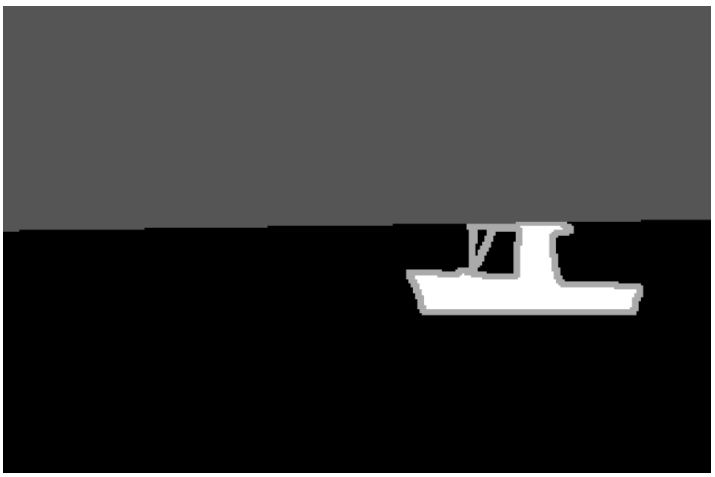

Figure 3.2 Boundaries of foreground objects and pixels outside the ROI are marked by gray value in the ground truth

3.3 Challenges

1. There were several time-consuming issues. First, generating background models for the whole dataset took several hours. Second, since I use Tensorflow to implement my model, I need to transform the dataset into binary file (“tfrecords” file required by Tensorflow). This also took several hours.
2. The biggest challenge is that the GPU on PC has too limited memory. Its memory is less than 4 GB, which is far from enough to load model parameters and image data. So I used cloud computing resources [10].

3.4 Result

Unfortunately, the training of the network is not finished yet. I will continue to test the model as soon as the training is done.

Reference

[1] Mohammadreza Babaee, Duc Tung Dinh, and Gerhard Rigoll. A Deep Convolutional Neural Network for Background Subtraction. arXiv:1702.01731v1 [cv.CV]

[2] St-Charles, Pierre-Luc, Guillaume-Alexandre Bilodeau, and R. Bergevin. SuBSENSE: A Universal Change Detection Method with Local Adaptive Sensitivity. CVPR Workshops 2014

[3] Rui Wang, Filiz Bunyak, Guna Seetharaman, and Kannappan Palaniappan. Static and Moving Object Detection Using Flux Tensor with Split Gaussian Models. CVPR 2014

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. CVPR 2016

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity Mappings in Deep Residual Networks. ECCV 2016

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. CVPR 2015

[7] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning Deconvolution Network for Semantic Segmentation. ICCV 2015

[8] Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar. CDnet 2014: an expanded change detection benchmark dataset. CVPR Workshops 2014

[9] TF-slim: tensorflow's high level API, https://github.com/tensorflow/models/tree/master/slim

[10] Cybera's Rapid Access Cloud, https://www.cybera.ca/services/rapid-access-cloud/