UNIVERSITY OF ALBERTA

2017 Mitacs Internship Presentation
Vehicle Counting in Surveillance Videos

Yiqi Yan

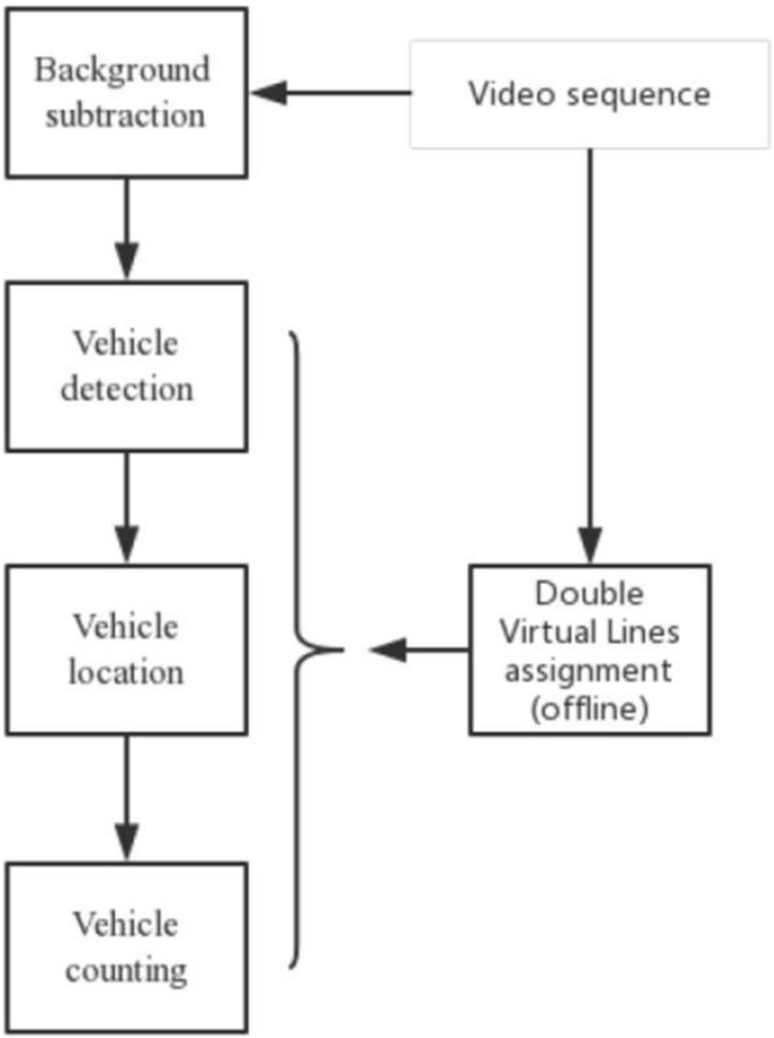
2017.9.5

Part I

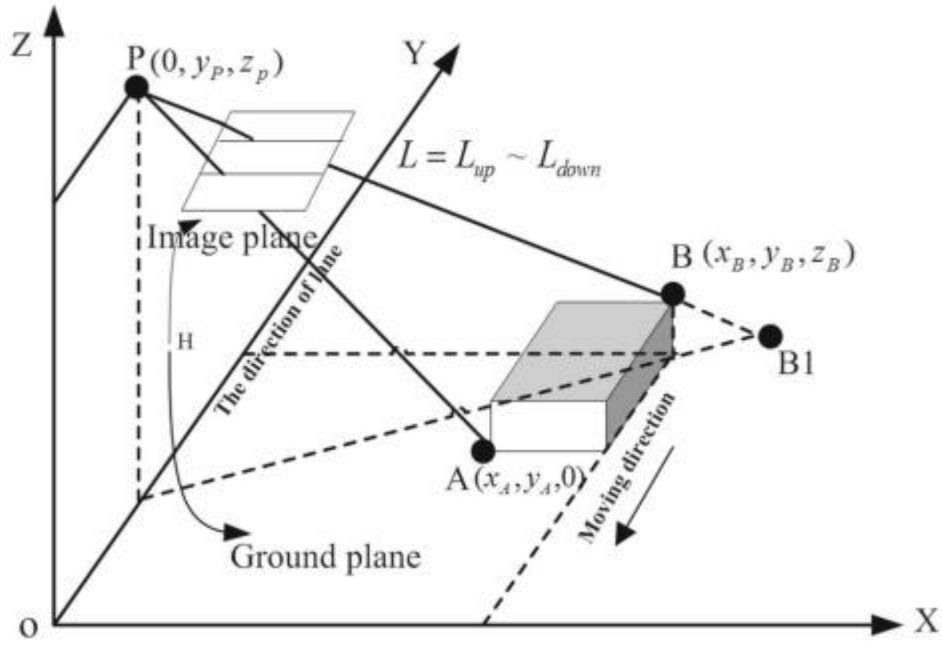
Vehicle Counting Using Double Virtual Lines (DVL)

1.1 Motivation

- Traditional vehicle counting methods: virtual loops
  - × ***Problem***: repeat counting may occur when vehicles are roadway departure due to overtaking or crossing
  - ✓ ***Solution***: assigning Double Virtual Lines
- How to count?
  - × ***Problem***: background subtraction results are not perfect
  - ✓ ***Solution***: template convolution combined with efficient counting rules



1.2 DVL Assignment



The DVL is assigned by estimating the vehicle's 2-D projection on the image plane. The projective transformation matrix of the camera is needed.

UNIVERSITY OF ALBERTA

1.3 Vehicle Detection and Location

- Background subtraction

Mixture of Gaussians (MOG) is used to model the background. Foreground mask is computed by subtracting background from the original image.

$$D_i(x,y) = f_i(x,y) - f_{bg}(x,y)$$

- Morphological filtering

Morphological filtering is used to remove the holes and enhance the targets. Concretely, dilation operation with a disk-shaped structuring element is used.

$$D_{i\_obj}(x,y) = dliate\{D_i(x,y)\}$$

1.3 Vehicle Detection and Location

- Template convolution

The template is a matrix filled with 1's, whose height is the same as the distance between DVLs. The convolutional operation is performed only in the detection zone, i.e. between the DVLs.

1.4 Counting Rules

- Rule #1 Large peak value

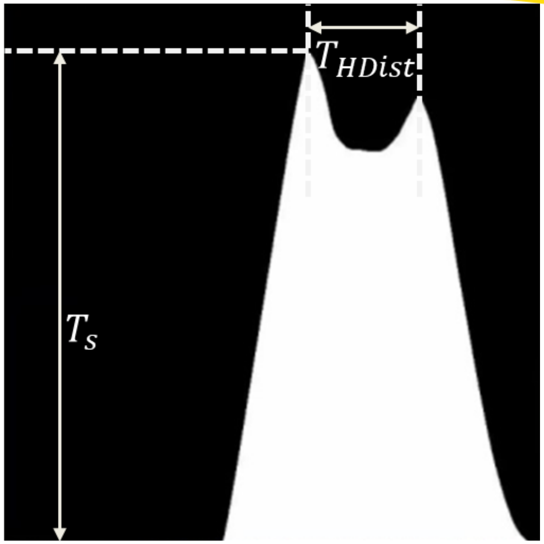
The peak value corresponding to the target should be larger than the threshold (T_s). This is designed to rule out the influence of noise.

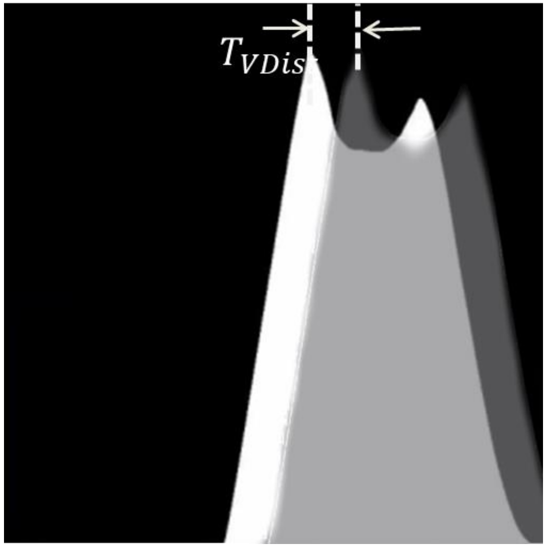
- Rule #2 Horizontal safety space

The distance between two neighboring peaks should be larger than the threshold (T_{HDist}).

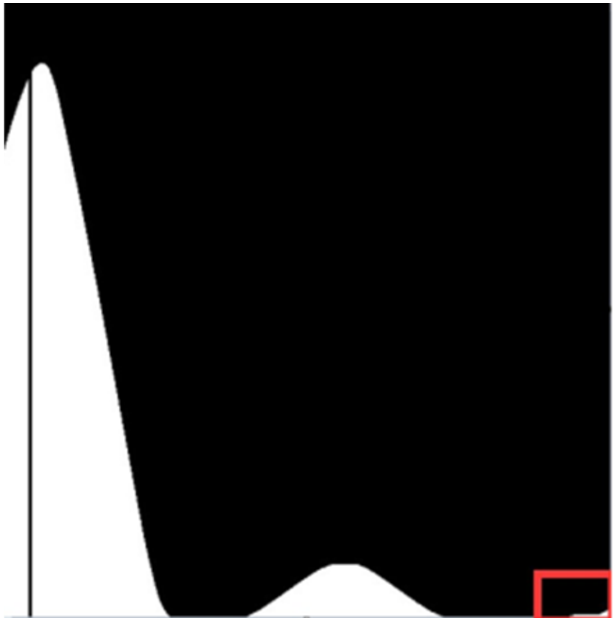
- Rule #3 Vertical safety space

The distance between any of the two peaks in two consecutive frames should be larger than the threshold (T_{VDist}). This rule is designed to eliminate repeat counting.



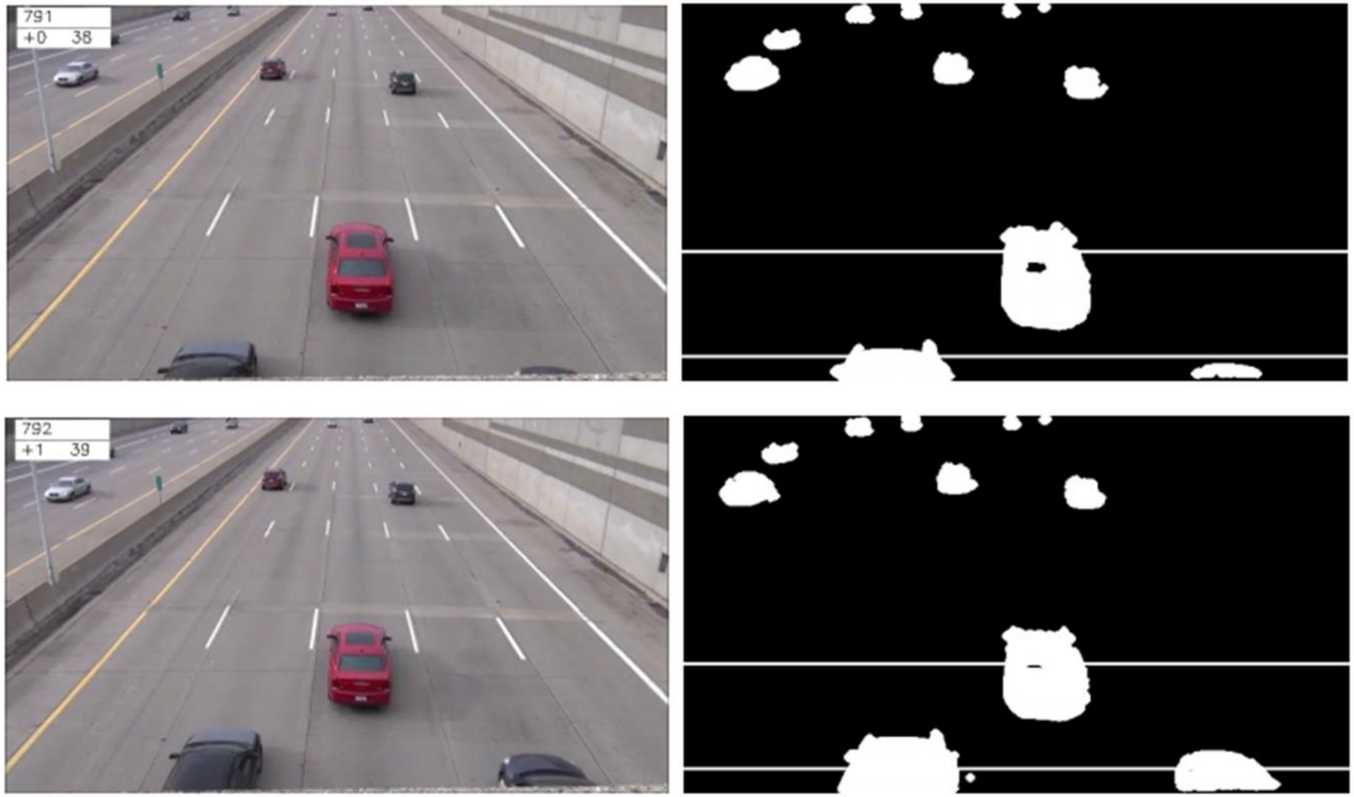


1.4 Counting Rules: Case Study



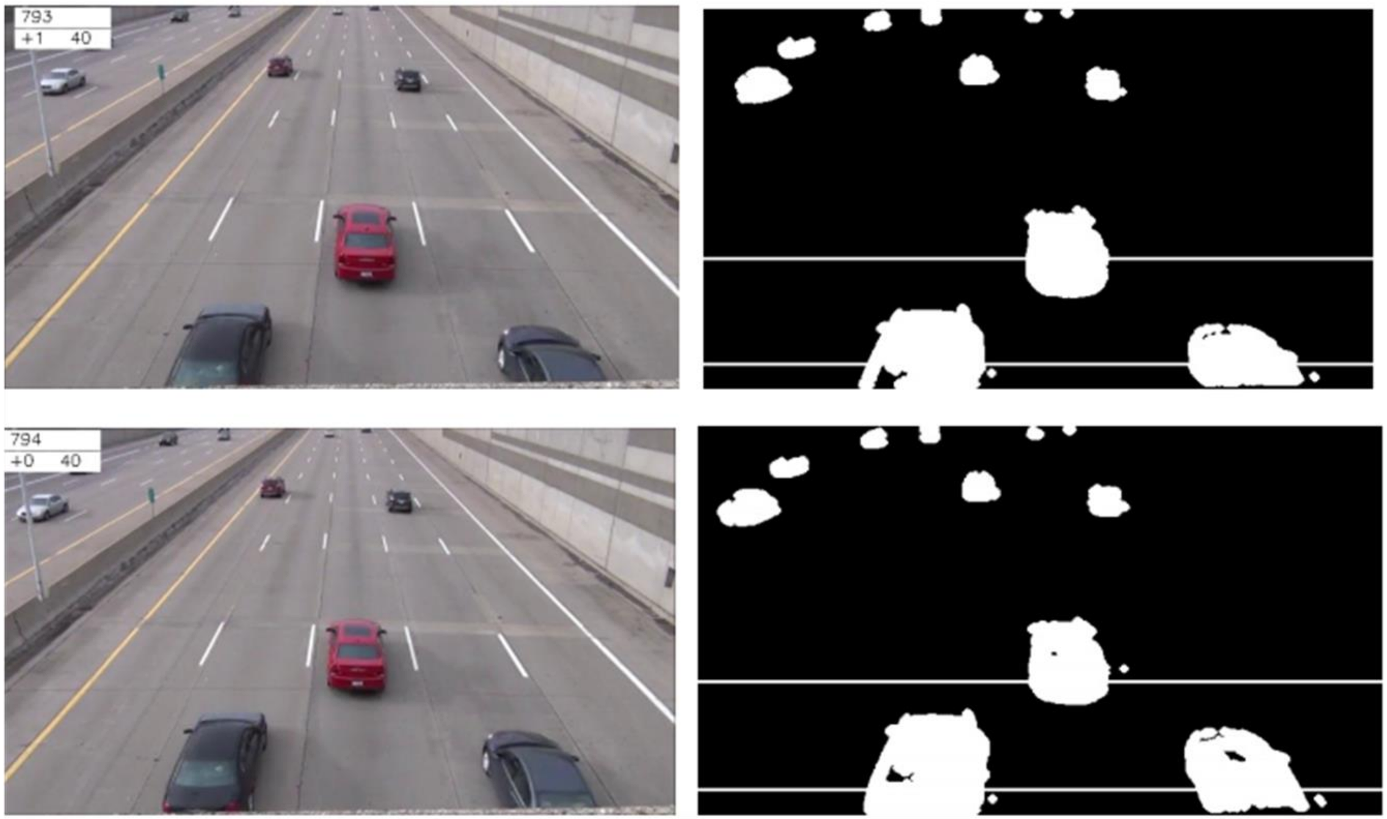
Rule #1 rules out the influence of noise.

1.4 Counting Rules: Case Study



Rule #2 & #3 prevent repeat counting.

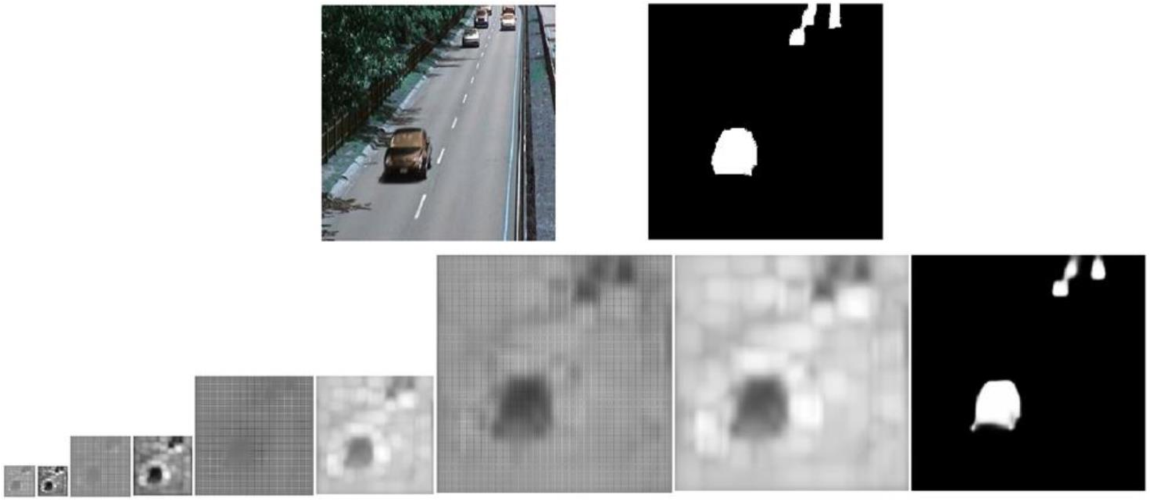
1.4 Counting Rules: Case Study



Rule #2 & #3 prevent repeat counting.

Part II

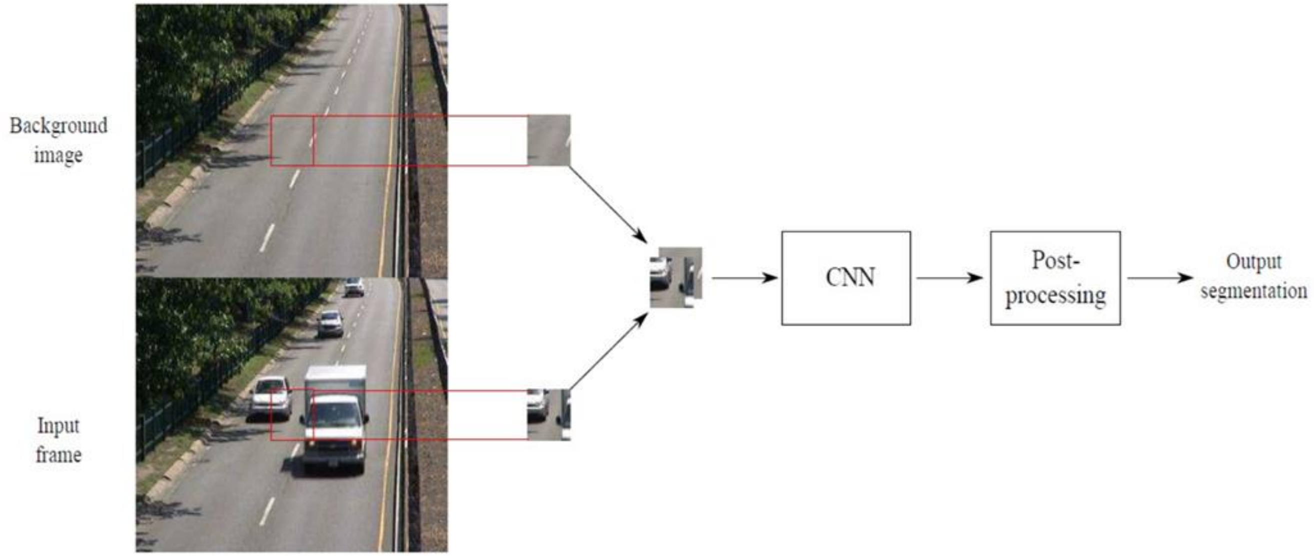
Background Subtraction Using Deep Learning



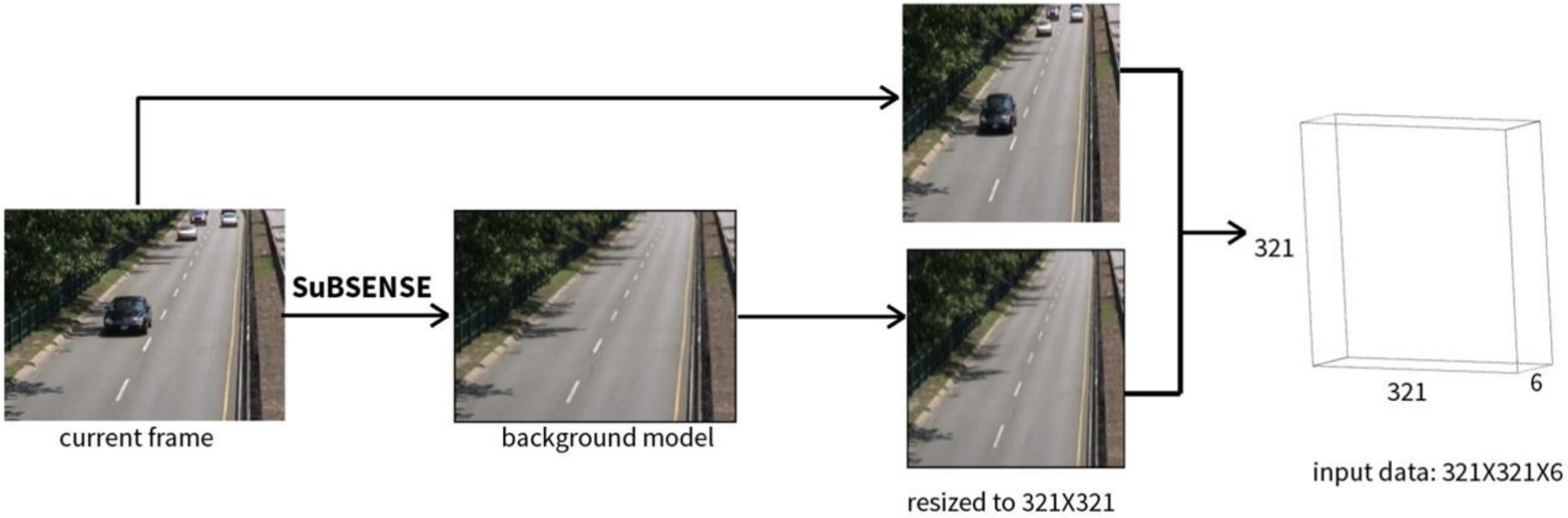
2.1 Motivation

Traditional methods: directly ***subtracting*** the background from each frame.

× ***Problem***: relies on the quality of the background model.
✓ ***Solution***: let CNN learn to ***compare*** the information of the background image and original image.



2.2 Generate Background Image



- SuBSENSE: iterative method; able to run online to generate background
- Merge background image & original frame: force CNN to consider both

2.3 Network Architecture

- 50-layer Resnet: feature extraction
- deconvolutional layers: up-sample the feature map
- max-pooling layers: eliminate extra zero elements in feature maps
- convolutional layer before Resnet: map the input data into a 3-channel feature map

	Filter size / Pooling window size	Stride (H,W,D)	Input size	Output size
pre-conv	1x1x6x3	1, 1, --	321x321x6	321x321x3
resnet_50	----	----	321x321x3	21x21x2048
3D avg_pool	1x1x48	1,1,40	21x21x2048	21x21x51
deconv_1	3x3x32x51	2, 2, --	21x21x51	43x43x32
3D max_pool	3x3x2	1, 1, 2	43x43x32	41x41x16
deconv_2	3x3x8x16	2, 2, --	41x41x16	83x83x8
2D max_pool	3x3	1, 1, --	83x83x8	81x81x8
deconv_3	3x3x4x8	2, 2, --	81x81x8	163x163x4
2D max_pool	3x3	1, 1, --	163x163x4	161x161x4
deconv_4	3x3x1x4	2, 2, --	161x161x4	323x323x1
2D max_pool	3x3	1, 1, --	323x323x1	321x321x1
conv	1x1x1x1	1, 1, --	321x321x1	321x321x1

2.4 Training

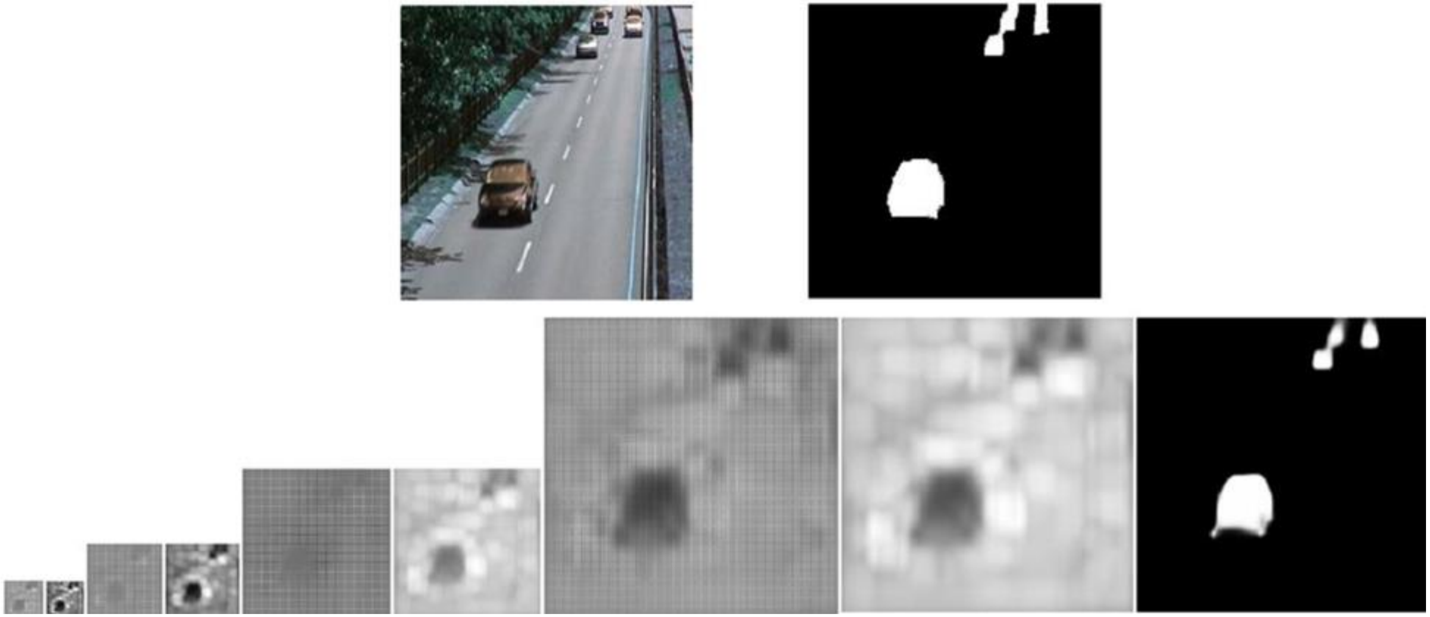
- Dataset: CDnet 2014
- Hardware information

RAM	8 GB
Disk	40 GB(system) / 100GB (hard drive)
GPU	Tesla K80
Total GPU memory	11.17 GB
Available GPU memory	11.09 GB

- Hyper-parameters

Optimizer	Adam
Mini-batch size	40
Maximum iteration	10000
Learning rate	1e-3 for the first 500 steps; 1e-4 for the last 1000 steps; 0.5e-3 for all other steps

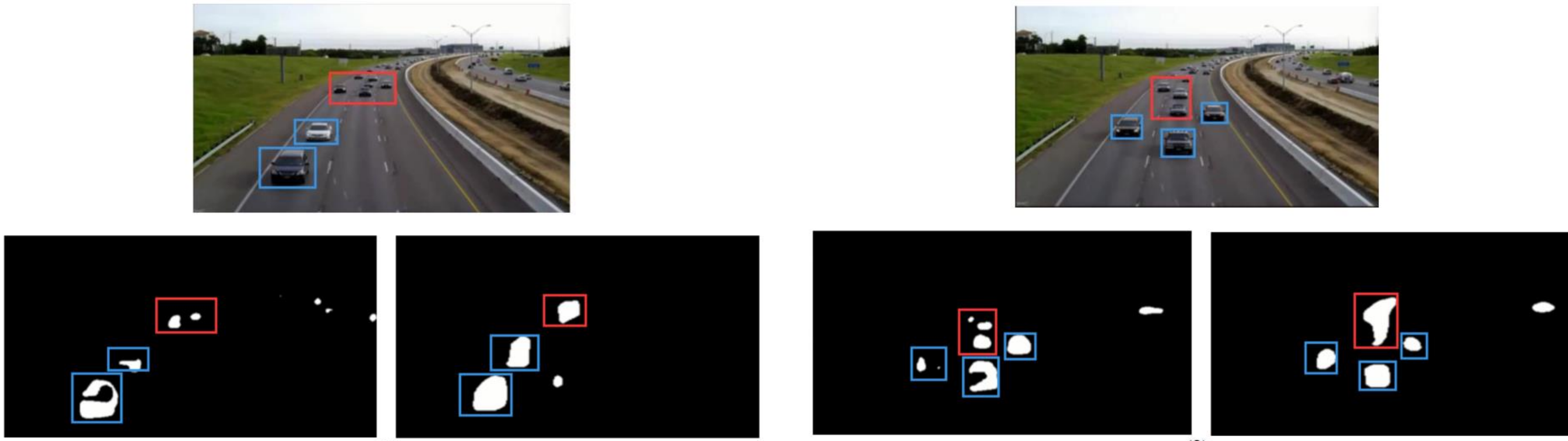
2.5 Results: Visualization



Output features of four pairs of deconvolutional-pooling layers, and the final sigmoid activation

Max-pooling layers reduce checkboard artifacts.

2.5 Results: Comparison with SuBSENSE



(1) (2)

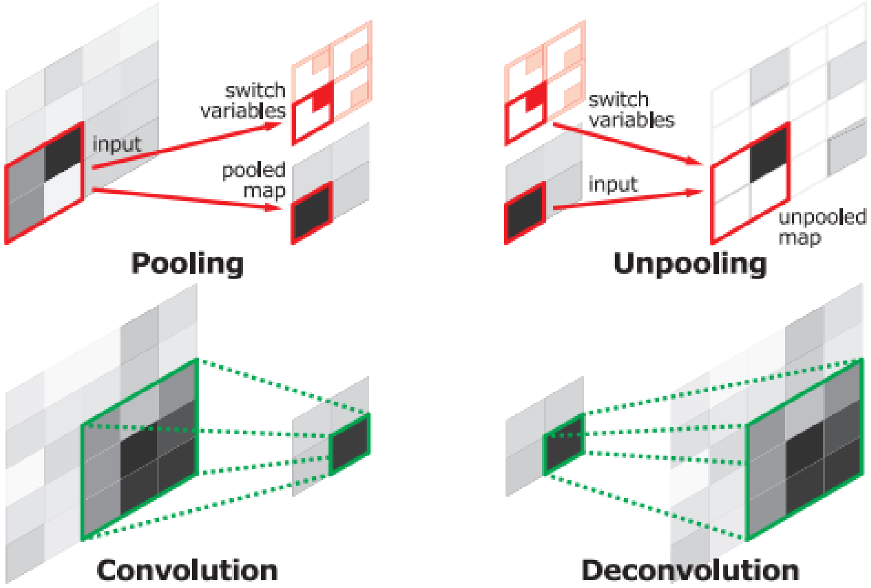
- Top: original frame
- Bottom left: foreground mask created by SuBSENSE
- Bottom right: foreground mask created by Model II

CNN model stands out in detecting large targets,
but fails in detecting distant, smaller ones.

Part III

Most Recent Work

(not included in reports or poster)



3.1 Motivation

- Problem of the aforementioned CNN model

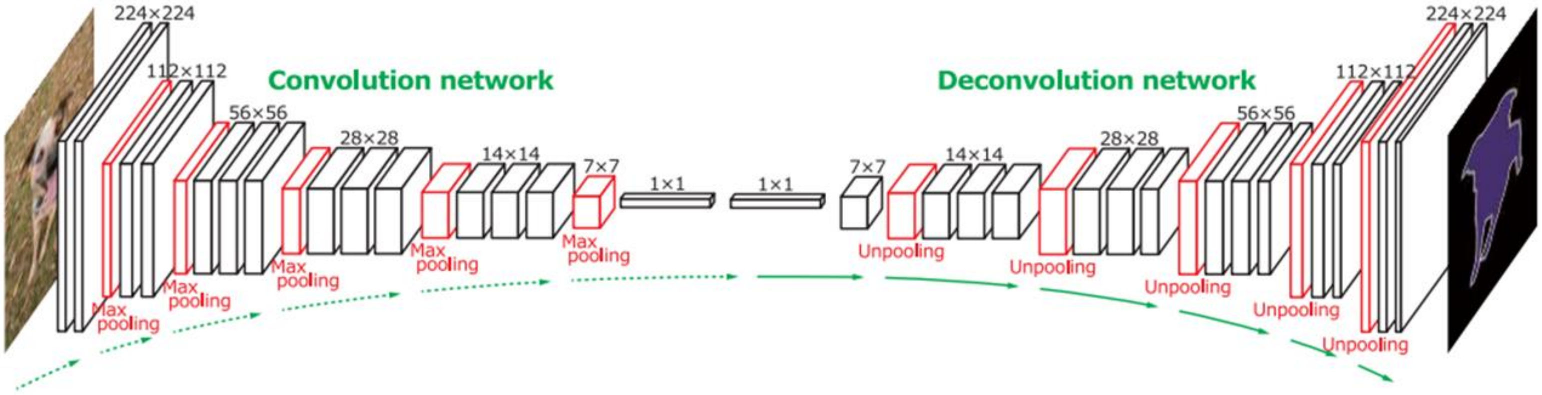
  × Input size must be fixed (321X321)
  × Sigmoid function is incompatible with ReLU activation

- Solution

  ✓ Use fully convolutional-deconvolutional network [ICCV 2015]
  ✓ Use soft-max function for pixel-wise classification

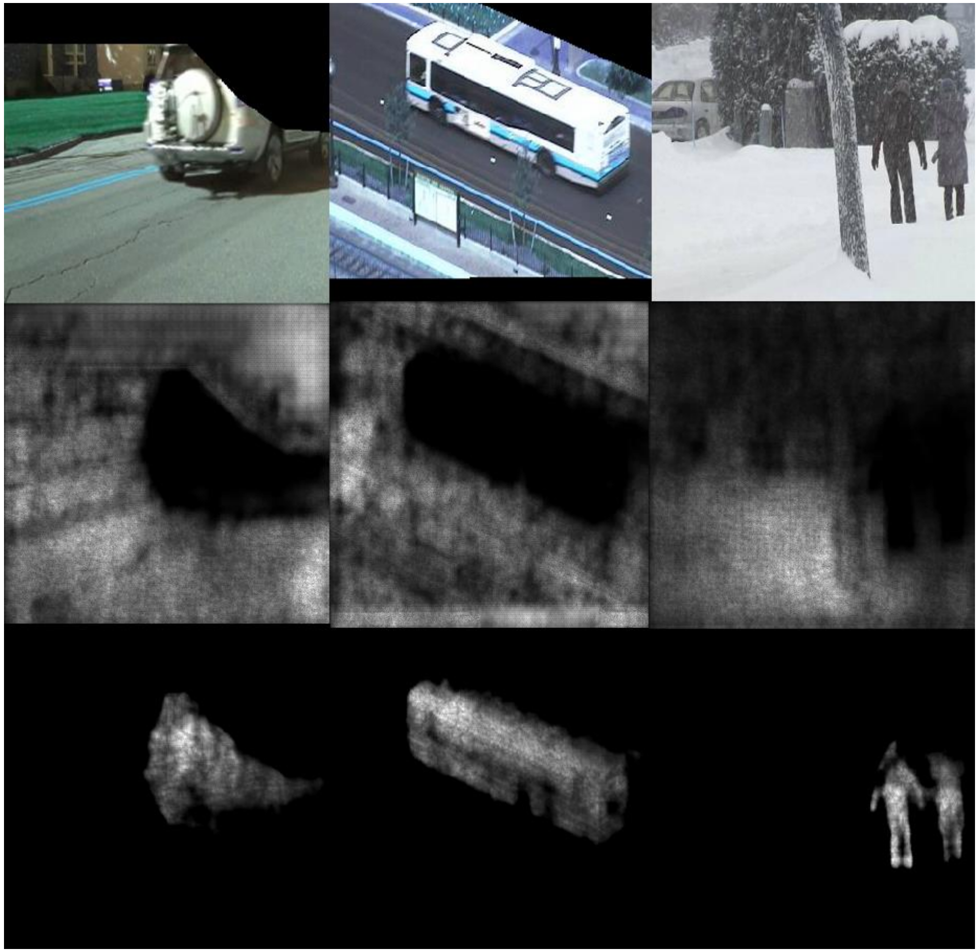
3.2 Architecture

- Encode: VGG-16
- Decode: deconvolutional layers and unpooling layers
- All the convolutional and deconvolutional layers use ***same padding***, down-sampling and up-sampling are performed via pooling and unpooling
- Output: 2-channel feature map (2-class pixel-wise classification)



Reference: H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in ICCV, 2015.

3.3 Result



- 1st row: original frame
- 2nd row: 1st channel in the output feature map
  target regions un-activated
- 3rd row: 2nd channel in the output feature map
  target regions activated