# Simultaneous Detection and Segmentation in Road Scene

**Group 32**


| Xin Li | Zequn Zhou | Jiahui Li | Anna Deichler | Fenglu Xu |
|---|---|---|---|---|
| 4721101 | 4714903 | 4734769 | 4507533 | 4579976 |



## Abstract

*Computer vision is an important tool for autonomous driving systems. Computational speed is a crucial point for real time applications such as autonomous driving. We propose to develop an approach to joint detection and semantic segmentation via a unified architecture where the encoder is shared amongst the two tasks just like MultiNet [5]. Cyclist detection in autonomous driving is still a relevant problem [6]. Compared to pedestrian detection, less attention has been given to the problem of detecting cyclists [7]. In this project we aim at delivering an algorithm with an enhanced cyclist detection capability.*


## 1. Introduction

To perform detection and segmentation, we build on the MultiNet architecture [8]Figure 1), we will implement simultaneous detection and road segmentation for cyclists. Our solution will include two components: object detection and road segmentation. Contrary to the MultiNet implementation, we will not implement road type classification, since we do are not considering multiple road types. The MultiNet architecture is modular, composing of KittiBox (car detection), KittiSeg (road segmentation) and KittiClass (road classification) submodules.
Consider that this system is designed for autonomous driving, which requires real time detection and segmentation, but segmentation requires more time to process due to pixel-wise prediction, which will be the bottleneck of this system. Inspired by Enet [5] , we investigate several methods to speed up segmentation process.

## 2. Architecture

The whole architecture contains three different parts, one encoder and two different decoders. The encoder processes the image and extract rich abstract feature for later segmentation and detection decoder. The encoder consists of the convolutional and pooling layers. The weights of the encoder are initialized using the weights of VGG16 pre-trained on the ImageNet.

### 2.1. Detection

We basically focus on cyclist detection. The encoder is from the top 13 layers of VGG16. and the decoder is built upon Fast-RCNN. We use the sum of two losses for detection: Cross entropy loss for the confidences and an L1 loss on the bounding box coordinates. Note that the L1 loss is only computed for cells which have been assigned a positive confidence label. The object detection part (KittiBox) of the MultiNet architecture is taken as a base algorithm. The weights for the car class are available online. The training for the cyclist class on the Daimler dataset is initialized from this car class trained weights. Our research question in the detection case was whether it is possible to use this pre-trained network and fine-tune it for the cyclist detection problem.

### 2.2. Road Segmentation

The network for road segmentation is mainly based on VGG16[7]-FCN8s[4]. This segmentation model keeps the first 13 layers of VGG16, and followed with 2 fully convolution layer. After that, it combines the up-sampling structure from FCN. The final network structure is shown in figure 2. Like the classification task, the segmentation is also trained by using a softmax cross-entropy loss function.

## 3. Improvements

Semantic segmentation has been well studied on improving the accuracy, but less attention has been paid on computationally efficient solutions. In Enet [5], the author proposed a new neural network architecture optimized for fast inference and high accuracy. To improve the speed based on KittiSeg, we use some tricks that has been proposed in Enet.

### 3.1. Dilated Convolution

Dilated convolution is a widely used trick to improve segmentation accuracy. Dilated convolution is basically the same with convolution, except that it introduces gaps into it's kernels, so that the receptive filed is larger while keep the same number of parameters as in Fig 3.

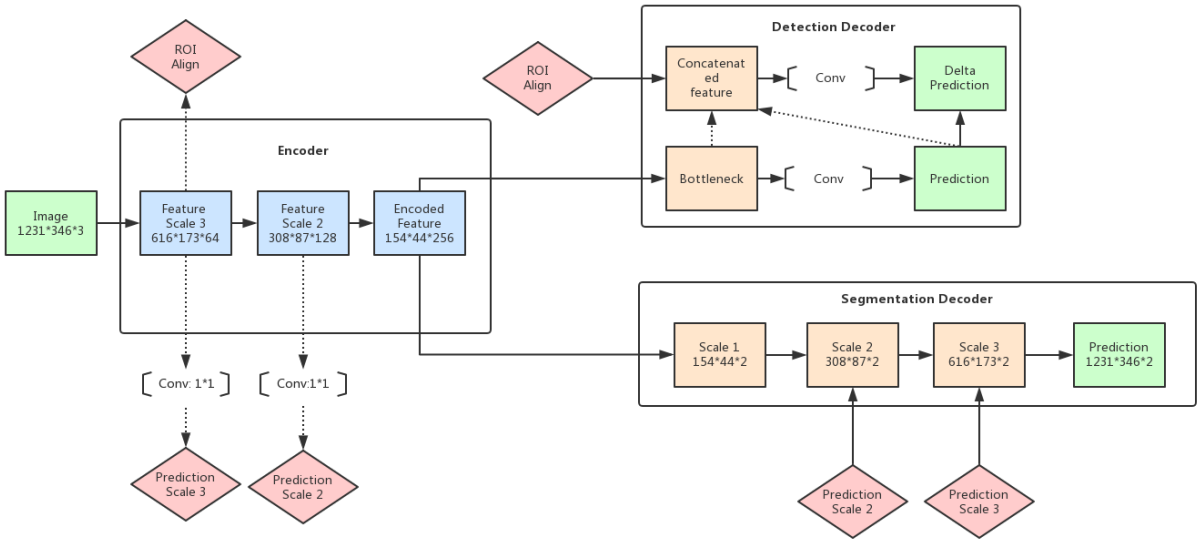



Figure 1: Our Net

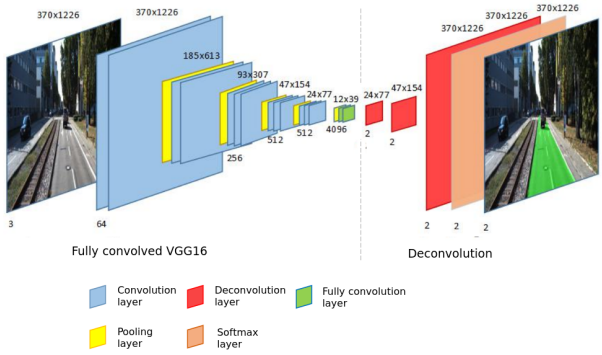
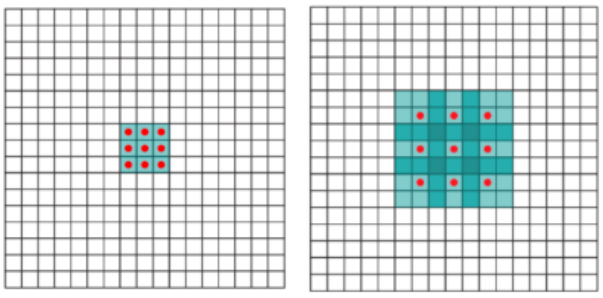



Figure 2: The original segmentation model



(a) convolution (b) 2 di-convolution

Figure 3: Compare convolution and dilated convolution

### 3.2. Model Compression

To speed up the inference process, reducing the number of parameters and computation complexity are basic methods.

#### 3.2.1 Early downsampling

In [5] one crucial intuition to achieving good performance and real-time operation is realizing that processing large input frames is very expensive. The computation overhead is basically determined by the size of the image. A simple intuition is that if we can predict the high resolution map given a low resolution image. The idea behind it, is that visual information is highly spatially redundant, and thus can be compressed into a more efficient representation. Also, our intuition is that the initial network layers should not directly contribute to classification. Instead, they should rather act as good feature extractors and only preprocess the input for later portions of the network.

#### 3.2.2 Network compression

Consider the expensive cost for fully convolution layers, (4096) dimensions, we directly discard fc6 and fc7 layer from origin KittiSeg, show as in Fig 2 (green colored).
One of the reason we downsample the feature map is to get a larger receptive field and generate high level features. To improve the speed and accuracy, we did not downsample the image 32 through pooling layers as MultiNet did, instead, we only down sample the image 8 times to make this model

simpler and introduced dilated convolution to compensate the loss of receptive field.

### 3.3. Batch Normalization

Batch normalization [3] is a technique for improving the performance and stability of neural networks by reducing internal covariate shift. VGG doesnt have a batch norm layer in it because batch normalization didnt exist before VGG. If we train it with batch normalization, the pre-trained weight will benefit from the normalization of the activations, like improve the speed of training.

### 3.4. Some general tricks

Data augmentation for segmentation(only 500 images used for training), Spatial dropout.

## 4. Experiment

### 4.1. Dataset

The MultiNet network weights trained on the KITTI dataset for autonomous driving. This dataset has few examples for the cyclist class. For the fine-tuning of the cyclist detection problem we have decided to use the Tsinghua-Daimler Cyclist Benchmark, which is a dataset specifically for cyclist detection. Since we use 2 different dataset of different size and labels, we perform data pre-processing by padding and manual labeling.

#### 4.1.1 Padding images

Figure 4 and 5 depicts the output of KITTI pre-trained MultiNet (trained to detect cars) for a Daimler example. The aspect ratio is different for the two datasets and Figure 3. shows that the algorithm resizes the Daimler image. Our hypotheses is that the bounding box prediction might not be accurate, therefore in further testing Daimler images are padded 6 to have the same aspect ratio as the KITTI dataset. The padding is already implemented and the padded Daimler images will be used for fine-tuning the network. Additionally we are also planning to run the network training on the original (not padded) images for comparison.

#### 4.1.2 Converting label format for detection

The Daimler and KITTI datasets have different label formats, which has to be taken care of. The Daimler dataset was converted to be in accordance with the KITTI dataset label format. Due to the different aspect ratios of the images in the two datasets, image padding was used in the detection case too. The Kitti dataset occlusion levels are defined as : 0 = fully visible, 1 = partly occluded, 2 = unknown. The Daimler dataset defines occlusion in percentages (no occlusion: $<10\%$, partially occluded: $10\% - 40\%$ ,heavily occluded: $40\% - 80\%$). No occlusion was given 0, partially and heavily occluded were given 1 in the relabeling.

, partially occluded (10

Figure 4: original Daimler image

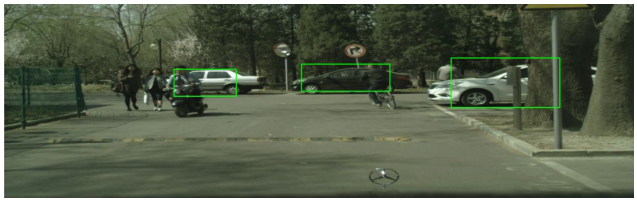


Figure 5: output of pre-trained MultiNet

Figure 6: example of padded image

#### 4.1.3 Labeling on Daimler dataset for segmentation

To fine-tuned the network for the road segmentation on Daimler dataset, we need to retrain the model on Daimler dataset, which only has detection labels but no segmentation labels. Therefore, we select 200 images from Daimler dataset and then manually label them as the label image of the Kitti dataset. We use magenta(255, 0, 255) to represent the road and use red(255, 0, 0) to represent other parts of the image. As an example, figure 7(a) and 7(b) show an original image and a manually labeled image of Daimler dataset respectively.

### 4.2. Detection Experiment

Based on Figure 15, the VGG-fc7-a architecture is used for detection. The evaluation protocol suggested [8] is used. The Daimler cyclist detection training and evaluation dataset contains 5000 images and 300 images are used

(a) Raw image (b) Manually labeled image

Figure 7: Manually labeled Daimler image

for testing. The bounding box detection evaluation is based on intersection over union (IOU). The results show that it is possible to fine-tune the pre-trained car detector network for cyclist detection (Figure 8). The difficulty levels are defined in accordance with the KITTI dataset. (easy: Min. bounding box height: 40 Px, max. occlusion level: fully visible, moderate: Min. bounding box height: 25 Px, Max. occlusion level: partly occluded, hard: Min. bounding box height: 25 Px, Max. occlusion level: Difficult to see).

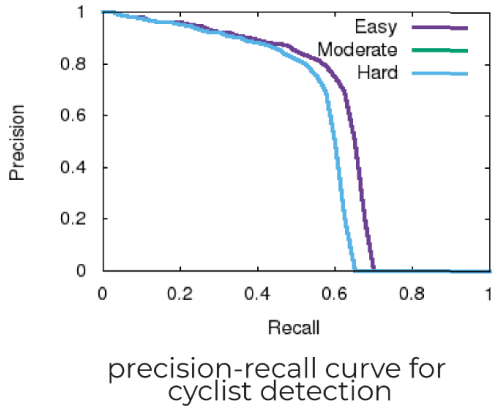



Figure 8: precision-recall curve for cyclist detection

#### 4.2.1 Failure cases

We observed the following problems in the cyclist detector

1. objects with too small bounding boxes are not detected (FN)
2. pedestrians (without bicycles) are detected (FP)
3. car wheels are detected (FP)

Figure 9 illustrated that, the fine-tuned cyclist detection network gives false positives for standing bikes and misses detecting cyclists with small bounding boxes. Figure 10 shows an example of the network giving a FP detection for car wheels. Some of these problems could be potentially alleviated by using a larger cyclist dataset. For example the case of detecting car wheels is understandable, given the

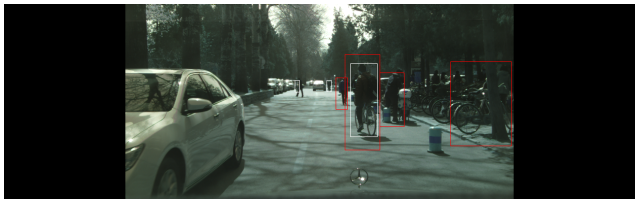


Figure 9: detection results showcasing failure cases (red:detection, white: groundtruth

Figure 10: detection results showcasing failure cases (red:detection, white: groundtruth

fast that a car detector was used for initialization. Additionally, in the Daimler dataset the images have a low contrast, which makes it difficult for the network to differentiate between a car and a cyclist, the wheels dominate.

### 4.3. Road Segmentation Experiment

To investigate the functionality of $KittiSeg$, we run it on the original KITTI dataset and got some interesting founds. The segmentation is sometimes not accurate enough and loses some details of the image. As shown in the figure 11, we can see that the left part of the road was not segmented appropriately, which indicated that $KittiSeg$ may be less sensitive to lower-level features. It is also noticeable that two fully convolution layers are employed in the segmentation model (figure 2). However, the two layers contribute to 80 % of parameters in the network and they are initially used as 'classification' in FCN[4]. In the segmentation task, it is redundant. Therefore, we removed the 2 convolution layers and directly up-sampled the output from pool5 layer. The result is shown in figure 12, the path is segmented comparably better. By removing the fully convolution layers, the segmentation speed has been increased evidently.
This experiment is a trial on modification of $KittiSeg$ with only KITTI dataset involved. In the next step, Daimler dataset will be added and more serious modification on $KittiSeg$ with proper evaluation metric will follow.

For segmentation, we follow the evaluation protocol suggested in [8]. In total, we test 5 models whose architectures vary a little. In the first model VGG16-fc7-a, We add atrous convolution and arrange them in sequence. To avoid overly down-sampling the features maps, in the second model VGG16-pool5-a, we unsample it from pool5.

Figure 11: up sample from fc7

Figure 12: up sample form pool5

Model E-v1 is applied with early downsampling as well as interleaving the atrous convolution with other convolutions. Model E-v2 is similar to Model E-v1 but takes downsample at a deeper layers. Model E-v3 takes one step further by adding batch normalization based on E-v2. Figure 14 shows the learning curve for these models on training and testing set respectively. We can see that E-v2 and E-v3 have comparatively better performance than other models in terms of accuracy and time to converge. E-v2 slightly outperform E-v3, which converges faster.

Then, we make a graph to present the relationship of speed and accuracy of these models. From figure 15, we can see that the accuracies of VGG16-fc7-a, E-v1, E-v2 and the original model are almost the same, but their speed are different. The segmentation speeds of E-v1 and E-v2 are faster than the original model, which means that downsampling can accelerate the segmentation but still maintain the accuracy. The speeds of E-v3 and VGG-pools-a are faster than the original model but their accuracy are slightly lower.

## 5. Conclusion

For real-time segmentation and detection system, we applied several tricks to compress the model, like only downsample 8 times of the origin image instead of 32 times and compensate receptive field's shrinkage with atrous (dilated) convolution. We get our best model: E-v1, approximately the same accuracy as the original one, the speed has been improved by 62.4%. But as in figure 13. (b) the result of E-v1 is not robust to light, while E-v2 is better, and its speed has improved by 40.25% The result of E-v3(applied batch normalization) is not as good as we expected, it is slower, probably we should set a larger learning rate. And the curve in 15 seems have not converge yet, maybe it can get a good result if continue training.

Our change of model is basically focus on the encoder, which is shared by segmentation and detection. Currently we do not test it on detection network and did not perform joint training due to limited time, too much data/label conversion for detection, also running out of credits of GCP.

There are some proposed ways to improve: Like ICNet [9], use different resolution of the image to perform cascade inference. Or write the maxpooling and convolution in parallel, this could get 10 times speed improvements according to [5]. Also to fine tune the model, the previous model mentioned in this article, which is trained on the Kitti dataset may be used as the pre-trained model. And the labeled Daimler image may train on this pre-trained model once again.

The structures like U-net[6], Segnet[1] and Deeplab[2] may also be considered as the future potential options to improve the performance of the segmentation part of this model.

## References


[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[2] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

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

[4] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[5] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[6] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[8] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun. Multinet: Real-time joint semantic reasoning for autonomous driving. *arXiv preprint arXiv:1612.07695*, 2016.

[9] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia. Icnet for real-time semantic segmentation on high-resolution images. *arXiv preprint arXiv:1704.08545*, 2017.

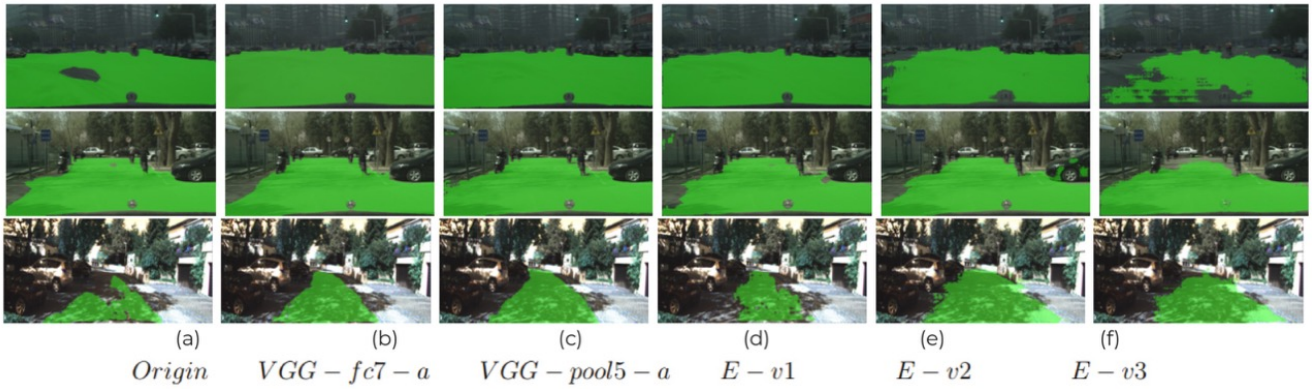



Figure 13: some segment results from different models
a)orignal image b) atrous convolution c) cut at pool5 and atrous, d) and e) early down sampling at different layers, f) additional batch normalization

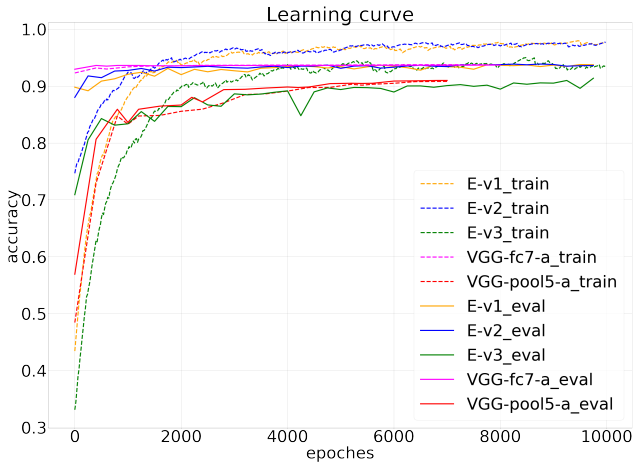



Figure 14: learning curve for six models. line of different color represents different models. Dashed line refers to the learning curve in training set while solid line refers to the learning curve in testing set

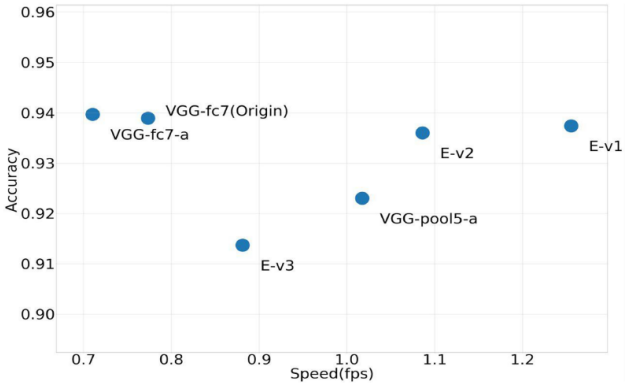



Figure 15: speed VS accuracy

## 6. Appendix

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Figure 16: original image

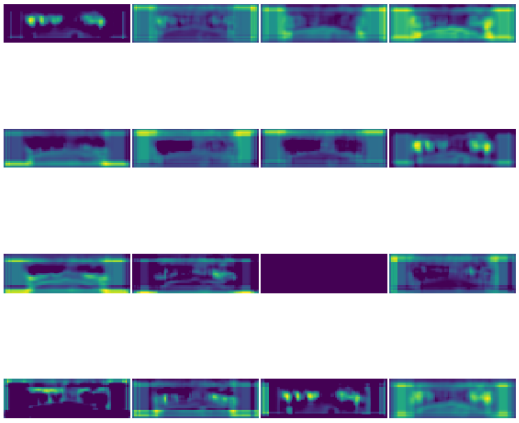


Figure 17: vgg fcn.pool5 feature map of car tuned detection network

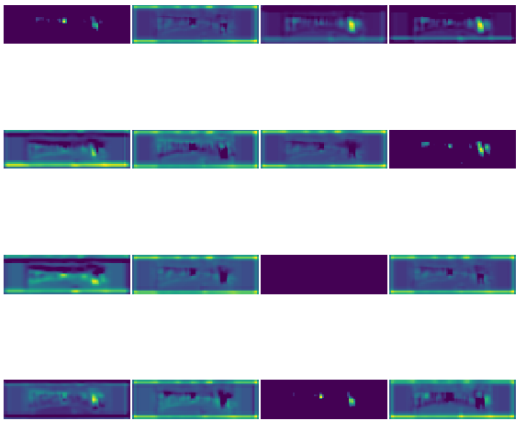


Figure 18: vgg fcn.pool5 feature map of fine-tuned detection network