Report for Practical Course: Hands-on Deep Learning for Computer Vision

INSTANCE-BASED SEGMENTATION

Qunjie Zhou, Yu Wang
Technische Universität München
Computer Vision Group

March 10, 2017

Contents

Chapter 1

Introduction

1.1 Instance segmentation and its challenge

The aim of the project is to study the state of the art methods on instance-based semantic segmentation and try to come up with new ideas using Convolutional Neural Network. Figure 1.1 is an example to illustrate the task. With a given raw image, to do semantic segmentation, the network should perform pixel-wise labelling, i.e. each pixel is assigned with a tuple class ID, instance ID. While for instance segmentation, different classes are assigned with different class IDs (colours) i.e. each individual car is assigned with a different instance ID.

The key challenges or difficulties of the task are as follows: (a) The number of instances in an image is not determined, i.e. it can be considered as a classification problem with unknown number of classes; (b) The result/loss should be invariant to the order of instance labelling, i.e. there are two cars (car A, car B) in the image, whether label car A as 1 and car B as 2 or label car A as 2 and car B as 1 should lead to the same segmentation result and loss. Thus, we need an appropriate loss to describe the properties of the problem; (c) There are also degenerated cases such as occlusions need to be handled.

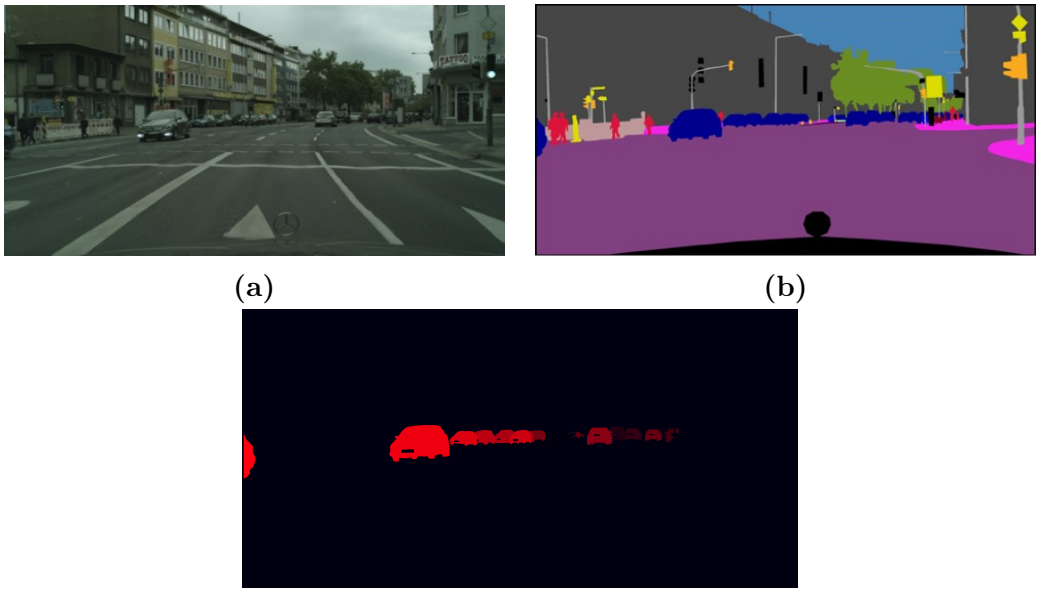


Figure 1.1: (a) is a given raw image, (b) is the ground truth of semantic segmentation, (c) is ground truth for instance labelling of car class

1.2 Related Work

People have developed several methods to tackle the problem and they can be categorized as following: (a) CNN-based multi-task network [1] or multiple networks [2]. They divide the task as sub-tasks: semantic segmentation and instance proposal. The semantic segmentation is usually done using Fully Convolutional Network (FCN) and instance proposal can be achieved by different techniques such as boundary detection [2], region proposal [1, 6] and depth ordering [3]; (b) RNN based network using LSTM to do instance counting [4]. (c) The state of the art method to our knowledge: Fully Convolutional Instance-aware Semantic Segmentation [5] and it is improved based on Instance-sensitive FCN [6] which we will implement in this project; (d) People have also developed a pipeline [7] using CNN as feature extractor and further make use of low level computer vision techniques such as contour detection to finally generate instance proposals.

1.3 Motivation

The reason why Instance-based semantic segmentation matters is that it can be applied to many applications, for example, Autonomous driving cars need to understand the surrounding areas to an instance level. And for robots to be interactive with specific individuals, e.g. grasping a specific object among others. Moreover, it can also be applied to track individuals in a sequence of images.

1.4 Datasets

There are several commonly used datasets for researchers to train and evaluate their networks, such as Cityscape, Pascal VOC, KITTI and NYU-Depth V2 and so on. We will however only use Cityscape in our project. The Cityscape dataset mainly focuses on street scene containing 30 classes, including persons, cars, traffic lights, etc. The dataset provides pixel-wise labelling for both semantic and instance. It has 3475 training and validating images, 1525 testing images with a resolution of 1024x2048.

Chapter 2

Experiments

2.1 Overview

After reviewing several latest related work, we notice that learning the semantic is a necessary subtask of instance segmentation. Therefore, we decided to first train a base network for the task of semantic segmentation and then extend the network for more complicated instance segmentation with our own idea, using a multi-stage training strategy. Therefore, we divided the project into following three phases:

1. Re-implement paper Fully Convolutional Networks for Semantic Segmentation[8]
2. Extend network for instance segmentation with spatial centre ordering
3. Re-implement Instance-sensitive Fully Convolutional Networks[6]

All experiments in this project are implemented with tensorflow v0.11.

2.2 Phase1: Fully Convolutional Networks for Semantic Segmentation

The paper proposed the idea to view fully connected layers as convolution with kernels covering entire input region. They modified VGG16 to a fully convolutional network, which was trained for semantic segmentation and achieved nice result. Also, considering the simplicity of the network architecture as well as the

understandable idea behind it, we decided to choose fully convolutional VGG16 network as our base network for further instance task.

2.2.1 FCN8s network

The basic network structure is illustrated by Figure 2.1. The three fully connected layers after pool5 in original VGG16 are replaced by conv6 layer with a 7x7 convolution kernel, conv7 layer with a 3x3 convolution kernel and score layer with 1x1 kernels to produce predictions. One difference in our implementation is to use three 3x3 kernels instead of a 7x7 kernel so that the number of parameters are reduced, meanwhile original receptive field is kept.

In the paper, they used skip connections from previous pooling layers to increase the output size. For example, fusing predictions from pool4 with a 2x upsampling of predictions from conv7, builds so-called FCN16s. The output will be 16x upsampled to original image size. We adopted the same scheme to build FCN8s as our base network.

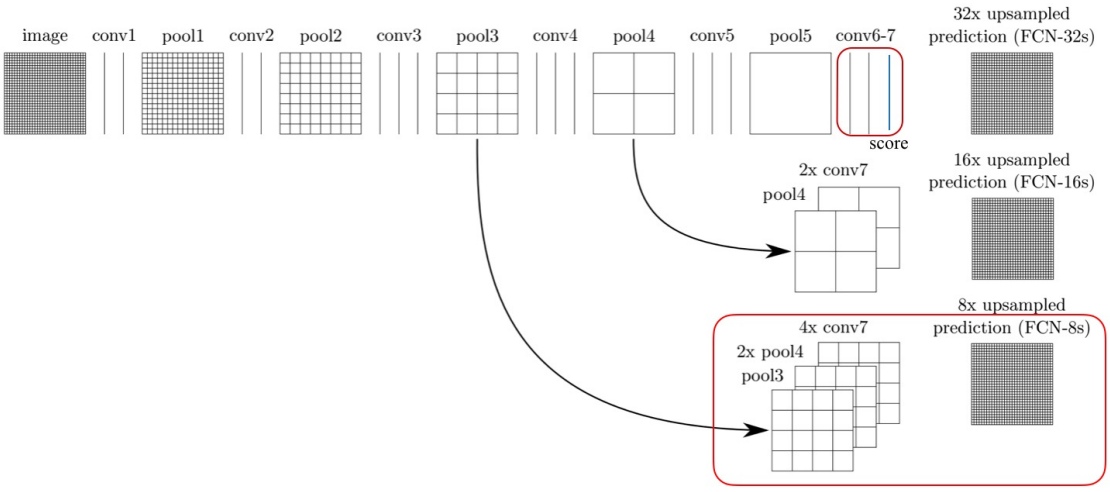


Figure 2.1: Architecture of FCN8s

To gradually improve the network and check its performance over different scales, we trained the network in a multi-stage fashion. Firstly, we trained FCN32s, i.e. training network without skip connection, for 15000 iterations using pre-trained VGG16 weight on ImageNet. Then we added the skip connection for

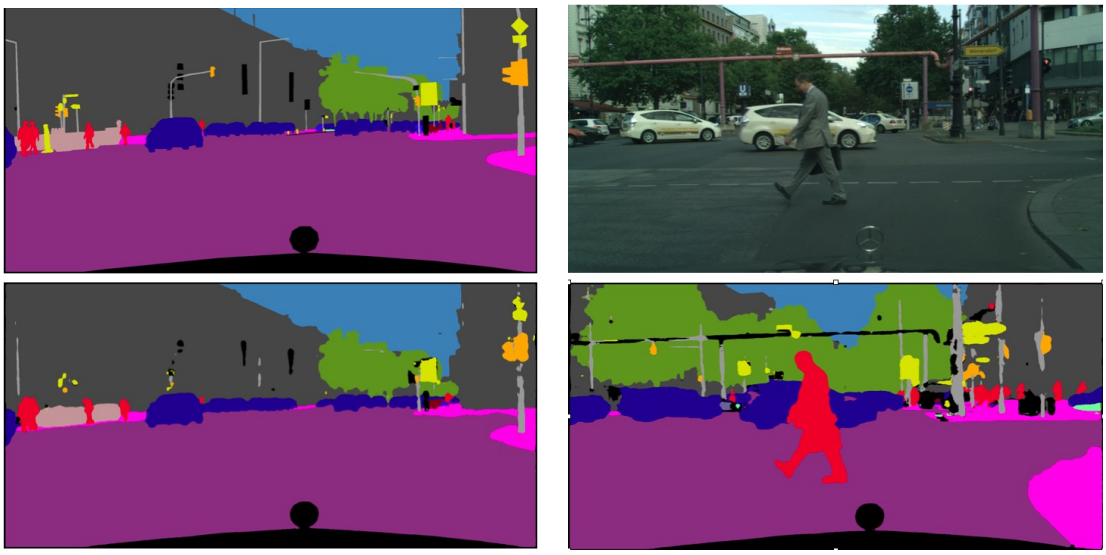
FCN16s and continued the training for 800000 iterations. Finally we trained FCN8s for 100000 iterations. We set learning rate to 1e-6 for whole training.

2.2.2 Inference result of our network

We then used the benchmark of Cityscape to verify the capability of our network, as results shown in Figure 2.2. And we noticed that car class has prediction accuracy of 91.95%, outperforming lots of other classes. The inference results of the network is illustrated in Figure 2.3

name	fine	coarse	16-bit	depth	video	sub	IoU class	iIoU class	IoU category	iIoU category	Runtime [s]	code
ENet	yes	no	no	no	no	2	58.3	34.4	80.4	64.0	0.013	no
FCN8s-QunjieYu	yes	no	no	no	no	no	57.4	34.5	81.8	68.7	n/a	no

Figure 2.2: Result on Cityscape benchmark



(a) ground truth(up), prediction(down) **(b)** raw image(up), prediction(down)

Figure 2.3: Inference on training image (a) and test image (b)

Until now, we think the result of semantic network is sufficient for testing the idea for instance task. Therefore we didn't specially tuned any hyper parameter to get better result, which is also not the focus of the project.

2.3 Phase2: From semantic to instance with spatial centre ordering

Our main idea is to tackle the issue of swapping ID i.e. label a car as car1 or car2 should both be regarded correct by fixing the order of instances. Other than using the depth information, we used the spatial centre location to order instances. The centre location is calculated by averaging the image location of pixels belonging to an instance. For two centre location a(x1, y1) and b(x2, y2), if x1 $<$ x2 then id(a) $<$ id(b). If x1 $==$ x2, then we compare y1, y2 in the same way. Here, we assume we don't have the case x1 $=$ x2 & y1 $=$ y2. To first test the feasibility of the idea, we work with a simplified model which has only cars class and we assume that there are at most 30 instances within each image.

Under the above setting, we can then consider the task of differentiating instances as a classification problem, where we have 30 classes in a fixed order. Therefore, it is straight forward to calculate loss by calculating cross entropy with softmax as distribution function. The hope is that the network can recognize the ordering pattern and use it for instance identification.

For convenience, we generated the ground truth before hand. As illustrated in Figure 2.4, for a semantic prediction, a set of masks are generated according to spatial centre ordering. If there are less than 30 instances, blank masks(with all zero) are appended after.

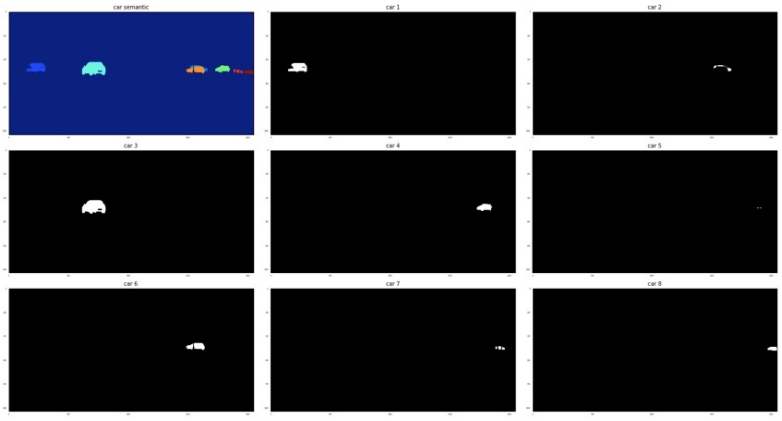


Figure 2.4: Generated ground truth for instance masks

2.3.1 Implementation with slicing operation

To only work with car class, we first take out the corresponding slice of car in score prediction by FCN8s. As shown in Figure 2.5, convolution is performed on the car slice, with a kernel of 3x3 size and 30 output channels, to produce 30 instance masks.

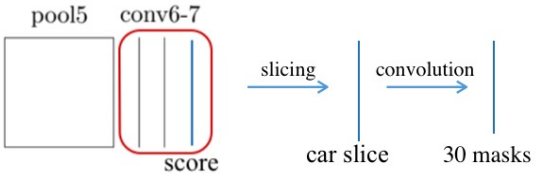
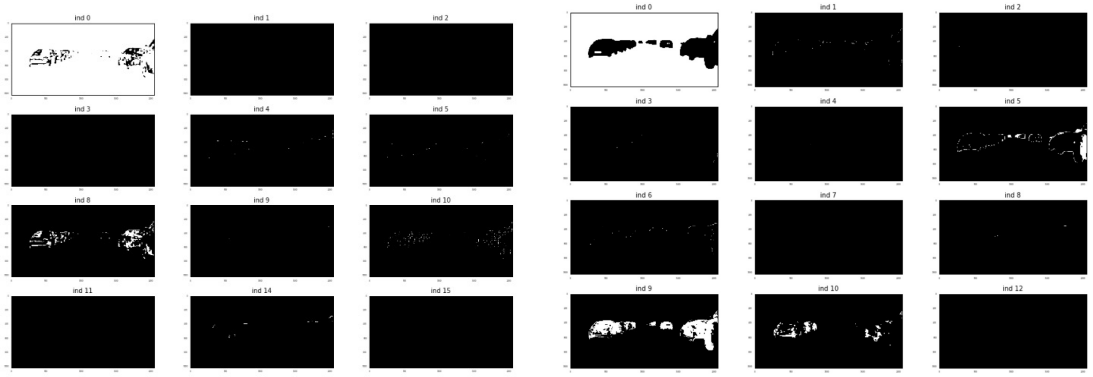


Figure 2.5: Network for instance task with slicing operation

Then the network is trained with learning rate 1e-4, for 80000 iterations using previously-trained FCN8s weight. The result after 15000 and 80000 iterations are displayed by Figure 2.6.



(a) Instance masks after 15000 iterations **(b)** Instance masks after 80000 iterations

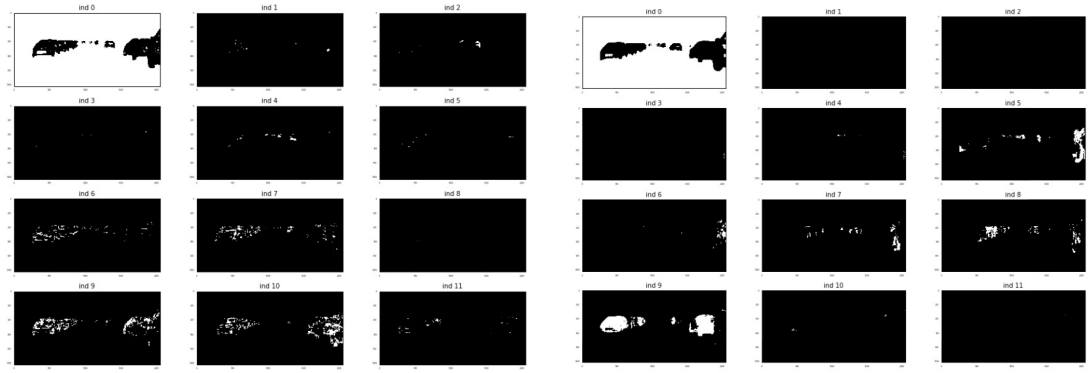
Figure 2.6: Inference result of network with slicing

We noticed in both (a), (b), background pixels are predicted as instance 0, while the pixels of different car instances looks like a random distribution over 30 masks. With the more iterations, there are no significant improvement on prediction, except for the background being labelled more accurately. Therefore, we concluded that the network failed to learn the instance segmentation. We guess that useful context is lost due to the slicing operation. As a result, we decided

to verify whether fully use of whole feature maps helps by directly fine tuning the FCN8s network for instance segmentation.

2.3.2 Implementation with full feature map

We changed the output classes of score map from 20 to 30, to directly generate 30 instance masks. And the network is trained in exactly the same way as before. The result after 15000 and 80000 iterations are displayed by Figure 2.7.



(a) Instance masks after 20000 iterations **(b)** Instance masks after 80000 iterations

Figure 2.7: Inference result of network with full features

Obviously, using all feature maps makes no significant difference from applying slicing operation. However, it's worth noting that after 20000 iterations, FCN8s filtered out information irrelevant to car class.

2.3.3 Conclusion

Both experiments revealed that the network failed to learn instance differentiation using spatial centre ordering. There are several guesses from us on such failure. As convolution operation makes the network invariant to translation, no matter where a car is, the network learns its feature anyway. Secondly, the spatial pattern is not stable enough for learning, because a car can appear anywhere in each image. Last but not the least, our loss function is not good enough to guide the learning, i.e. putting semantic and instance together could be too heavy.

2.4 Phase3: Instance-sensitive Fully Convolutional Networks

In the previous experiments, we know that CNN is invariant to translation: Our network learns the whole feature of a car in general and it can not differentiate between different cars. We have already tried to guide FCN to learn spatial ordering of different instances, yet the network cannot preserve spatial information well.

However, in this paper, the authors proposed a different learning scheme: they train the FCN to learn features of different parts of an instance instead of the whole feature of an instance, after that they make use of the learned features to assemble instances and pick out meaningful ones as the final instance proposal. The general idea is as follows: They divide an instance into different parts according to their relative positions, for instance, 3x3 grid (resulting in 9 parts). And the network learns 9 instance-sensitive feature maps, each of which corresponds to one part of instances as shown in Figure 2.8. For example, the 3rd feature map only captures the upper right part of a person and the 4th feature map only captures the left part of a person.

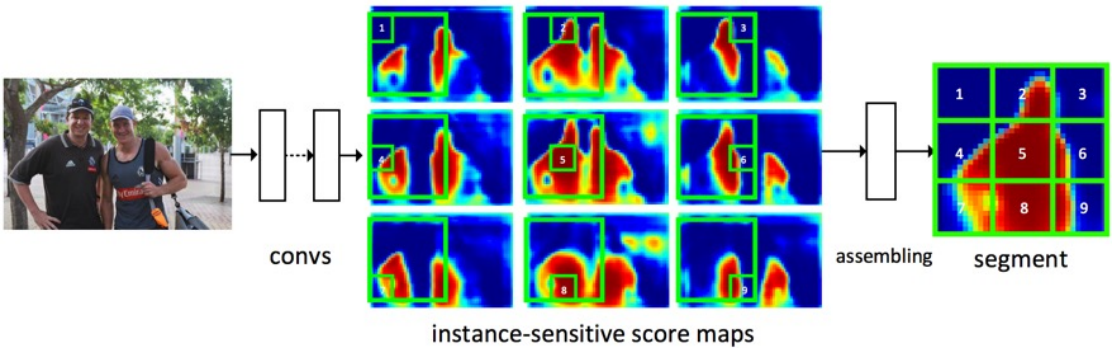


Figure 2.8: Instance-sensitive feature maps

Compared with other methods, for example, CNN based multi-task or multi-networks, this network is not complex and the loss function is straightforward. The paper achieved nice results, yet has similar architecture as ours and that's the reason why we tried to re-implement it and see how the network learns.

2.4.1 Assembling instance from feature maps

To generate a whole instance from different feature maps, they slide a window (3x3 grid) simultaneously on those feature maps and only copy a sub-window from each feature map, for example, copying sub-window 1 from feature map 1 and copying sub-window 2 from feature map 2. Finally, assembling those 9 copied sub-windows (according to their relative position) into one new window to produce a complete instance proposal.

However, one should be aware that the assembled instance proposal will not always be meaningful. Imagine that the sliding window is at the position where there is no instance at all, the assembled window will not be a valid instance. Therefore, another score map is needed to evaluate how likely the assembled window contains a valid instance. Thus each instance proposal needs a corresponding objectness score that measures the likelihood of that proposal being meaningful.

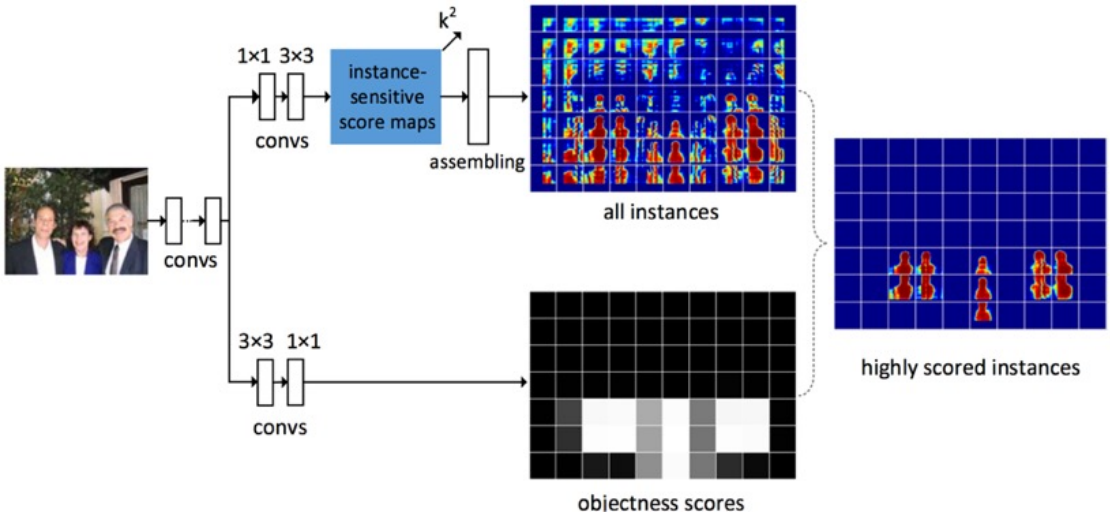


Figure 2.9: Network architecture for Instance-sensitive FCN

2.4.2 Network Architecture

The network (Figure 2.9) makes use of modified FCN8s as feature extractors, on top of it there are two additional branches, one (upper branch) for generating instance-sensitive feature maps, the other (lower branch) for scoring instances.

The upper branch has two fully convolutional layers, one with 512 kernels of size 1x1x512 followed by 512 kernels of size 3x3x9. The lower branch has two fully convolutional layers as well, one with 512 kernels of size 3x3x512 followed by 512 kernels of size 1x1x1. The stride for both branches is 2. Each instance proposal produced by the upper branch has a corresponding objectness score in the score map (lower branch). And the highly scored instances are picked out on the right on which non-maximum suppression (NMS) is performed to give the final set of proposed instances during inference. Note that this paper replaces the skip architecture with hole algorithm [9, 10] in FCN8s to do upsampling whereas we tried both of them in our implementation.

2.4.3 Training strategy and loss computation

In order to train the network, the paper proposed to slide 256 windows (size 21x21) randomly on the instance-sensitive feature maps. The instances are assembled from these 256 sliding windows for the loss computation. The loss function is defined as:

$$\sum_{i}(\mathcal{L}(p_i, p_i^*)) + \sum_{j}(\mathcal{L}(S_{i,j}, S_{i,j}^*)) \tag{2.1}$$

The p_i is the predicted objectness score for the sliding window i and p_i^* is the ground truth score, p_i^* is 1 if the window is positive (containing a valid instance) and 0 if it is negative. S_i is the assembled proposal of window i, S_i^* is the ground truth, and j is the pixel index in this window. $\mathcal{L}$ is the logistic regression loss. The training sliding windows consist of 128 positive windows and 128 negative windows. And the paper also randomly rescales the input images but we didn't for simplicity, for more details and other parameters, please refer to the original paper [6].

We followed the above training suggestions in the original paper, however we did some modifications. In the original paper, the 256 sliding windows are randomly generated on the instance-sensitive feature maps during training, however, we generated them for each image in advance so that we can know where each window locates in the image and on the feature maps. We then use the generated

256 windows (bounding boxes) to compute the loss. According to the original paper, the sliding window size is fixed, however, our window size can vary. Another reason why we generated our own ground truth boxes is that unlike the original paper, the dataset we use does not provide bounding boxes.

2.4.4 Generating ground truth

We generated the ground truth bounding boxes as follows: For the positive box (containing a valid instance), the object should be precisely centred in the box with a margin of 32 pixels to each side of the box boundary. And in order to introduce tolerance, we shifted the object on arbitrary direction up to +- 16 pixels randomly. The negative box either contains incomplete object or the object is shifted too much: at least +- 32 pixels.

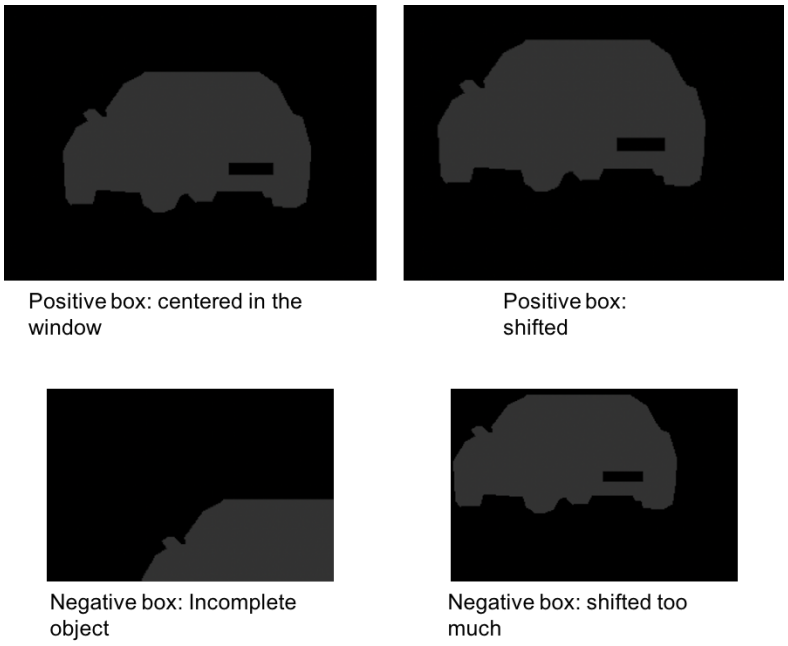


Figure 2.10: Ground truth bounding box

2.4.5 Implementation I

For the first implementation, we used the FCN8s with skip architecture before the two branches. For the instance proposal loss, we use exactly the same as

in the original paper. For the objectness loss, since each window produces one objectness score and the paper doesn?t mention how the predicted objectness score is computed, we think that the mean score over all pixels in the window might be a good measurement. Thus, we use absolute value of difference between predicted mean score and true score as the objectness loss. We only compute the instance proposal loss if the window is a positive one. We tune the network for 50000 iterations using previously trained FCN8s weight with a learning rate of 1e-6. The produced 9 instance-sensitive feature maps and the object score map is as follows (Figure 2.11):

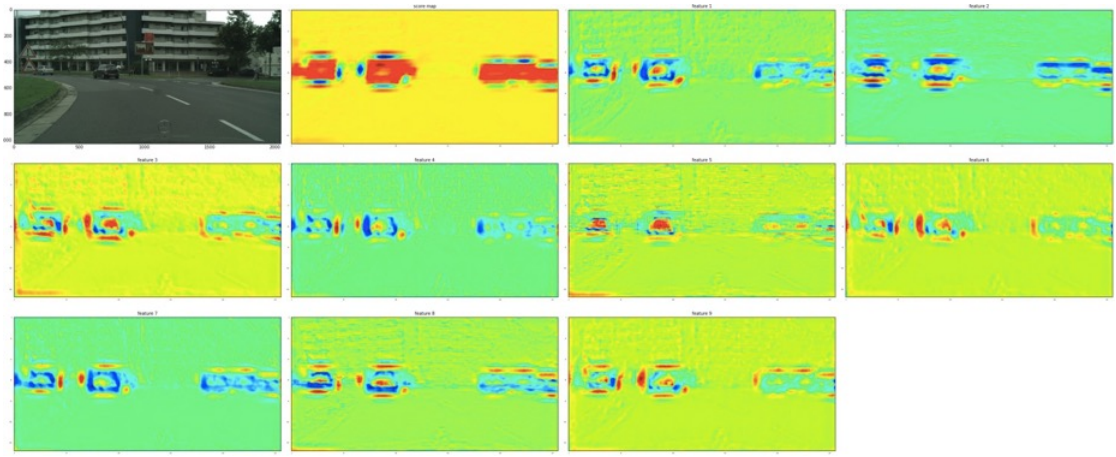


Figure 2.11: Result of the first implementation

Note that the 2nd map on the first row is objectness score map. The red pixels indicate that they are likely to be a valid instance. However, the network didn?t learn the expected instance-sensitive feature maps, for example, we expect the 3rd map on the first row to learn the upper-left part of a car and 4th map to learn upper part of a car. We think the possible reasons are: (a). The calculation of objectness loss may not be appropriate; (b). The calculation of instance proposal loss may not be fair due to the unbalanced portion of object and background: if the object is very small in a window, then labelling the object as background won?t contribute much for the loss; (c). The skip architecture we used in FCN8s may not provide sufficient fine details during upsampling; (d). The ground truth we used is different from the original paper.

2.4.6 Implementation II

Therefore, we tried another implementation. This time we used exactly the same architecture as in the original paper. For the instance proposal loss, we adopted the weighted loss in [10] to tackle the unbalanced portion problem. For the objectness loss, we guess that scores of central 4 pixels might be a better measurement and we used pixel-wise logistic regression to compute the loss. We tuned the network for 100000 iterations using previously trained FCN8s weight and decreased the learning rate to 1e-8. The produced 9 instance-sensitive feature maps and the object score map is as follows (Figure 2.12):

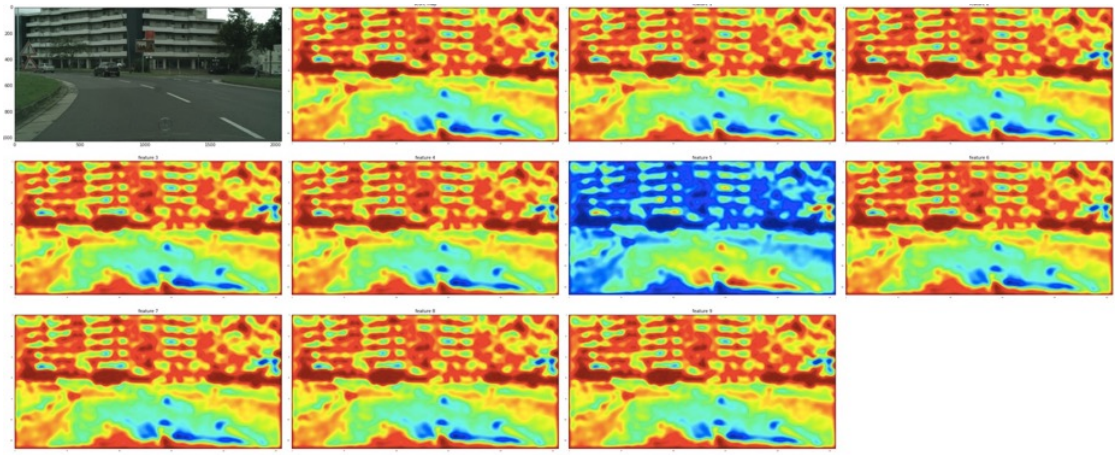


Figure 2.12: Result of the second implementation

Unfortunately, the network again didn't learn the expected feature maps again. We think the possible reasons are: (a). The calculation of objectness loss is not appropriate again; (b). The ground truth we used is different from the original paper; (c). Fine-tuning based on FCN8s-semantic segmentation might not be appropriate since we observed that the feature maps captured the semantics of buildings.

There are still a few things that we can try out: (a). Use the pixel-wise logistic regression over the entire window as the objectness loss; (b). Change the dataset as used in the original paper or generate the ground truth boxes exactly as in the original paper; (c) Use pre-trained VGG16 weight to fine-tune the network directly instead of FCN8s semantic weight.

2.4.7 Implementation III

We then tried our third implementation. Considering every pixel in a window should be counted into objectness score as calculating instance score, we calculate normal sigmoid cross entropy without weight over the entire window. And the objectness ground truth is a same window filled with either 0 (if it's a negative sample) or 1(if it's a positive sample). On the other hand, we decided to train the network from scratch i.e. using pre-trained VGG16 as weight initialisation. We trained the network first with 50000 iterations, and we tested the network after 30000 iterations. The result (Figure 2.13) shows that the network at least learned how to detect local features this time. Therefore, we continued the training with another 100000 iterations, hoping it could finally converge. However, as shown in Figure 2.14, the result after 150000 iterations differs very slightly from the result after 30000 iterations. Also by looking at the loss tendency (Figure 2.15) of second training of 100000 iterations, the loss doesn't converge at all.

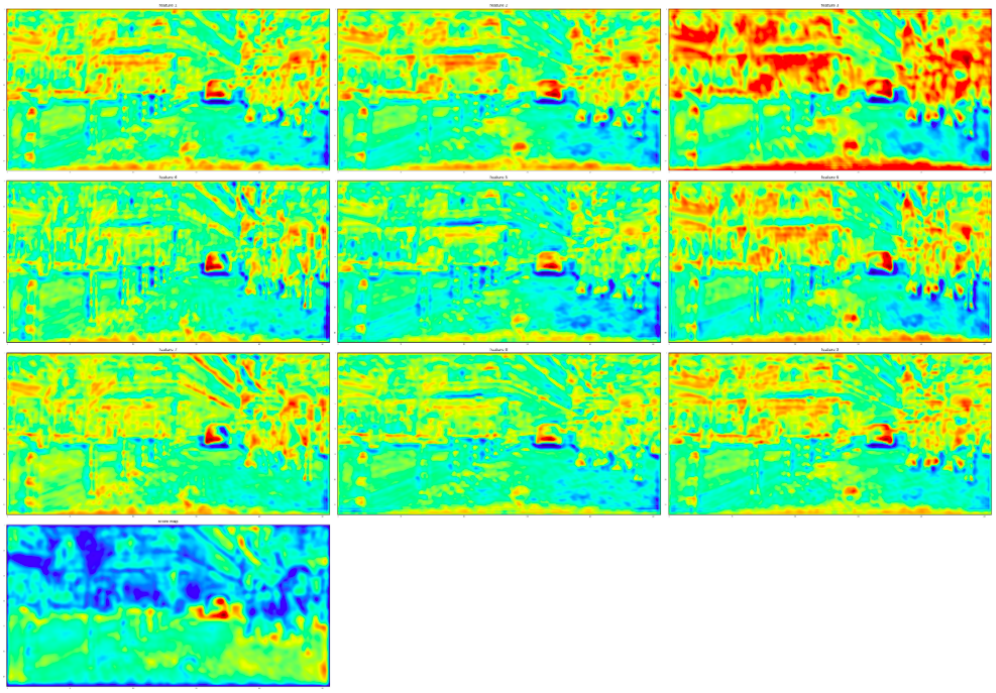


Figure 2.13: Feature maps and objectness score map after 30000 iterations

By now we would believe that the problem is more likely to be having used

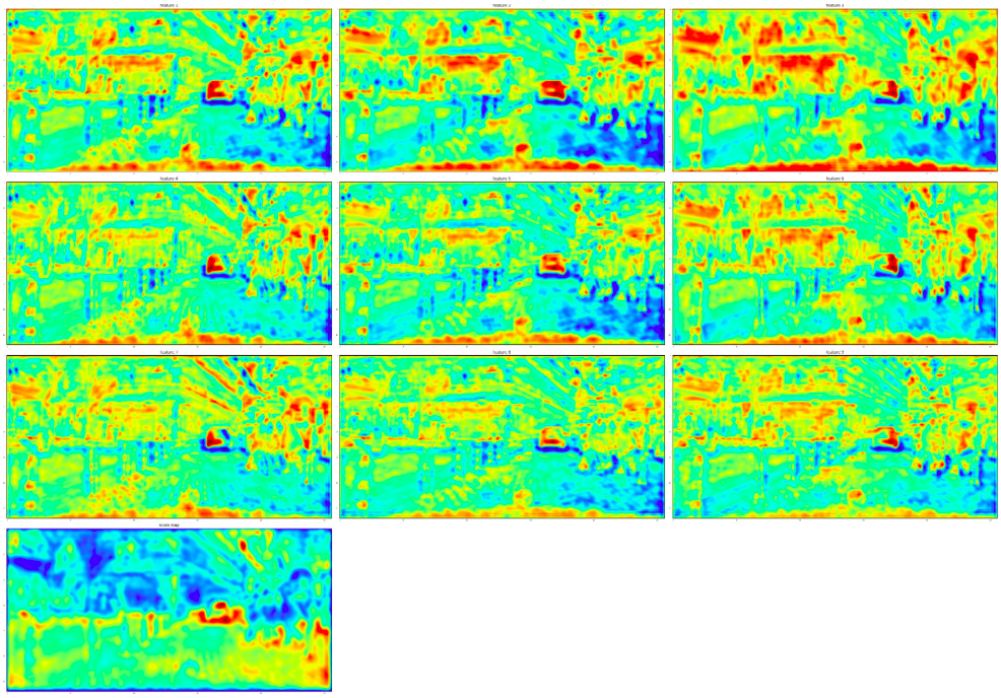


Figure 2.14: Feature maps and objectness score map after 150000 iterations

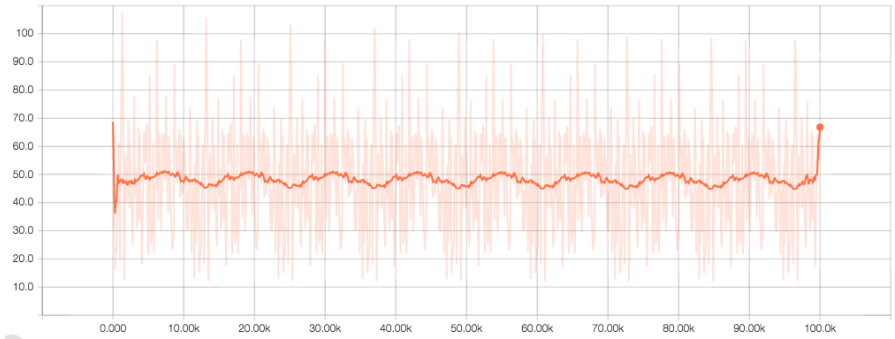


Figure 2.15: Result of the second implementation

a different dataset other than wrong implementation of the paper. The paper tested their method on PASCAL VOC and MS COCO, which contain more large objects, while Cityscape has much more small objects. Therefore, we guess that the method is not going to work on dataset with small objects. This guess should also make sense, because if an instance itself is already small, the distinction between its local features also becomes weaker. And till now, we decided not to continue experiments with this paper.

Chapter 3

Conclusion

To summarise, what we have done in this practical course are listed as follows.

1. First, we re-implemented FCN8s network for semantic segementation and achieved satisfactory results.

2. Then we tested our own idea to extend our semantic network for instance segmentation with spatial centre ordering, which didn't work out for the pattern is not learnable.

3. Finally, we re-implemented Instance-sensitive FCN, and concluded that the method might not work for dataset with small objects.

As instance-based semantic segmentation is a popular but tough problem, one still needs to come up with novel ideas for more experiments. Due to the time limitation, we can not try everything out. However, we think one future direction is to find a better pattern so that the network is able to learn and the loss can converge.

Bibliography

[1] Dai J, He K, Sun J. Instance-aware semantic segmentation via multi-task network cascades[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 3150-3158.

[2] Jin L, Chen Z, Tu Z. Object Detection Free Instance Segmentation With Labeling Transformations[J]. arXiv preprint arXiv:1611.08991, 2016.

[3] Uhrig J, Cordts M, Franke U, et al. Pixel-level encoding and depth layering for instance-level semantic segmentation[C]//German Conference on Pattern Recognition (GCPR). 2016.

[4] Romera-Paredes B., Torr P.H.S. (2016) Recurrent Instance Segmentation. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision ? ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9910. Springer, Cham

[5] Li Y, Qi H, Dai J, et al. Fully Convolutional Instance-aware Semantic Segmentation[J]. arXiv preprint arXiv:1611.07709, 2016.

[6] Dai J, He K, Li Y, et al. Instance-sensitive fully convolutional networks[C]//European Conference on Computer Vision. Springer International Publishing, 2016: 534-549.

[7] Silberman N, Sontag D, Fergus R. Instance segmentation of indoor scenes using a coverage loss[C]//European Conference on Computer Vision. Springer International Publishing, 2014: 616-631.

[8] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[9] Chen L C, Papandreou G, Kokkinos I, et al. Semantic image segmentation with deep convolutional nets and fully connected crfs[J]. arXiv preprint arXiv:1412.7062, 2014.

[10] Mallat S. A wavelet tour of signal processing[M]. Academic press, 1999.

[11] Caelles S, Maninis K K, Pont-Tuset J, et al. One-Shot Video Object Segmentation[J]. arXiv preprint arXiv:1611.05198, 2016.