Depth-aware Instance Segmentation with a Discriminative Loss Function

Z.Wang

Master of Science Thesis

Delft University of Technology

Cognitive Robotics Lab

Depth-aware Instance Segmentation with a Discriminative Loss Function

Master of Science Thesis

For the degree of Master of Science in Cognitive Robotics Lab at Delft University of Technology

Z.Wang

August 16, 2018

Student number:	4590058	
Project duration:	September, 2017 – July, 2018	
Thesis committee:	prof.dr. D.M. Gavrila,	TU Delft, chair
	dr. J.F.P. Kooij,	TU Delft, supervisor
	ir. E. Pool,	TU Delft, daily supervisor
	dr. J. van Gemert,	TU Delft, reader
	dr. L. Nan,	TU Delft, reader

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Cognitive Robotics Lab, Delft University of Technology

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Depth-aware Instance Segmentation with a Discriminative Loss Function

by

Z.Wang

in partial fulfillment of the requirements for the degree of

Master of Science Vehicle Engineering, Perception and Modelling

Dated: August 16, 2018

Supervisor(s):

prof.dr. D.M. Gavrila Chair

dr. J.F.P. Kooij Supervisor

ir. E.P. daily Supervisor

Reader(s):

dr. J. van Gemert Reader

dr. L. Nan Reader

Abstract

This work explores the possibility of incorporating depth information into a deep neural network to improve accuracy of RGB instance segmentation. The baseline of this work is semantic instance segmentation with discriminative loss function [1] and the incorporating method is inspired by the work [2].

The baseline work proposes a novel discriminative loss function with which the semantic network can learn a n-D embedding for all pixels belonging to instances. Embeddings of the same instances are attracted to their own centers while centers of different instance embeddings repulse each other. Two limitations are set for attraction and repulsion, namely the in-margin and out-margin. A post-processing procedure (clustering) is required to infer instance indices from embeddings with an important parameter **bandwidth**, the threshold for clustering.

The contribution of the work in this thesis are several new methods to incorporate depth information into the baseline work. One simple method is adding scaled depth directly to RGB embeddings, which is named as **scaling**. Through theorizing and experiments, this work also proposes that depth pixels can be encoded into 1-D embeddings with the same discriminative loss function and combined with RGB embeddings. Explored combination methods are **fusion** and **concatenation**. Additionally, two depth pre-processing methods are proposed, **replication** and **coloring**. From the experimental result, both scaling and fusion lead to significant improvements over baseline work while concatenation contributes more to classes with lots of similarities.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my parents first for providing emotional and financial support to me. Otherwise I cannot manage to finish this expensive while worthy master program in Delft University of Technology.

I would like to thank my supervisor Dr.J.F.P. Kooij for his assistance during the writing of this thesis. The door of Julian's office is always open to me whenever I run into a dilemma or have questions or ideas. Julian is a very responsible supervisor who inspired my critical thinking and research ability. My thesis is an independent work instead of part of an integral project. This allows me to explore those possible directions freely and implement my own idea. Julian is serious while easy going. Every time when we discussed about something, he always listened to my idea respectfully and gave me important and useful feedback. I feel it is a honor for me to be his first master student. I learned a lot from him, being critical, being attentive, being patient, etc.

Then I would like to thank Ewoud, my mentor, for his assistance during the reproducing procedure of the baseline work. It is not easy to reproduce results to be exactly the same as presented in papers, especially in the field of deep learning. We struggled for a long time to figure out the problems we met. I remembered I almost gave up at this stage. Luckily, I managed to reproduce the baseline work and then carried on with my own creativity. Thanks to Ewoud for all his help.

And thank Thomas and Ronald for the free coffee and hot chocolate. It helps me to relax a bit during working. Thank you!

My family-like housemates provided me with much warm and help in the past two years. I went through some tough time in the first two quarters. I really appreciate it that I have someone to talk to, Bianca. She is always rational and supporting all the housemates like a sister (She does not want to be referred to as 'mom').

I spend around 7 months in Eco-runner, one of the dream teams in TU Delft. I met many nice people there. It is a precious memory.

Delft, University of Technology
August 16, 2018

Z.Wang

“Everything in this world is magic, except to the magician.”
— *Dr. Robert Ford (Anthony Hopkins), Westworld, Season 1: Chestnut*

Chapter 1

Introduction

One intuitive purpose for having autonomous driving is facilitating disabled people or seniors who have mobile difficulty. They can easily move around with an intelligent vehicle (IV) performing dynamic driving task (DDT) itself. Juniors and drunk people are also target customers. Another reason for the rapid development of IV is comfort requirement. It is not hard to imagine that the driver feels like to relax a bit on his or her way back home after working all day. Keeping focus on road is neither comfortable nor safe under this circumstance considering that fatigue is one of the main causes of traffic accidents. This is another purpose for developing IV: to partially avoid human errors.

Perception is one of the core technologies that render automated driving possible. Perception involves various signal processing tasks related to radar, camera and Lidar. Vision task, the most prevalent one among them, has a high demand on computer science technology, such as deep learning, machine learning and computer vision. Deep learning had been neglected as an image processing technique for a long time until 2012, AlexNet [21] won the ImageNet Large Scale Visual Recognition Competition (ILSVRC). This denotes the revival of deep learning. ImageNet is a dataset containing RGB images of 1000 classes' objects. The work involved in this thesis, depth-aware instance segmentation with a discriminative loss function, is based on deep learning.

This Chapter is arranged as follows: Section 1-1 presents a short description for intelligent vehicle and its automation level. Then Section 1-2 will go into perception of intelligent vehicle and elaborate some vision tasks, classification, localization and segmentation. Requirements of these applications on board are listed in Section 1-3. This section also discusses why is instance segmentation necessary. Research question is proposed at the end of Section 1-4.

1-1 Intelligent Vehicle

Vehicle automation is officially classified into 5 levels according to SAE international criterion [3]. Level-1 automated vehicle is capable of generating recognizable signals to warn the distracted driver of certain dangerous situation. It cannot take over any Dynamic Driving

Tasks (DDT) for the driver to prevent emergency. With one level up, level-2 automated vehicle is able to perform partial dynamic driving task, either lateral control or longitudinal control. For instance, lane keeping and adaptive cruise control[22]. If both lateral control and longitudinal control are included, this vehicle is in the third level indicating that driver is allowed to take several minutes "eyes-off the road". A level-3 vehicle can response to critical situations immediately. Level 4 and 5 are highly to fully automated driving. The difference between these two high levels is that driver is still required in level-4.

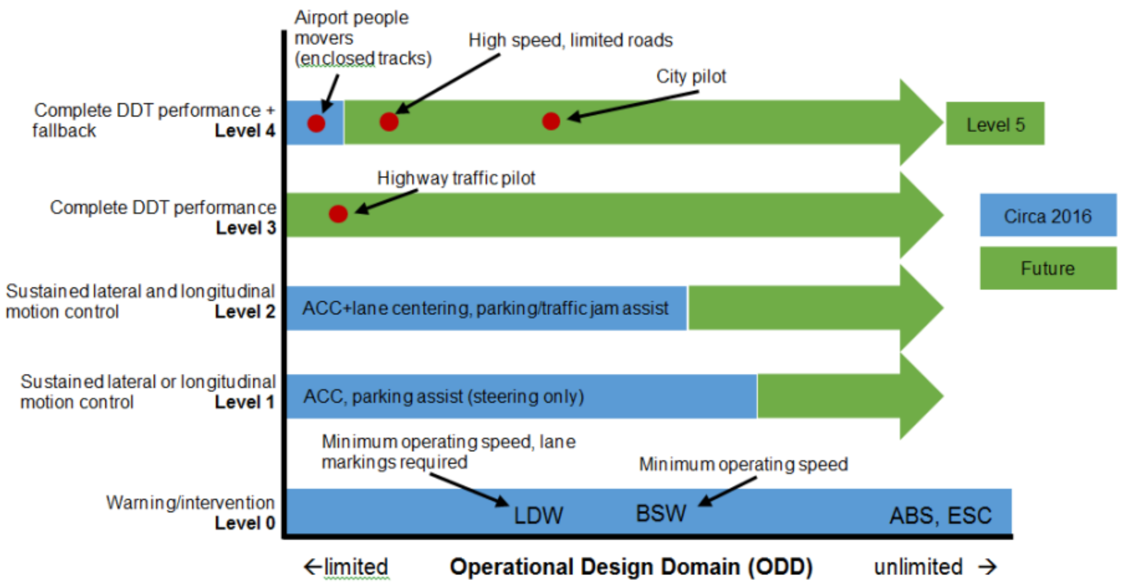

Airport people movers (enclosed tracks)
High speed, limited roads
City pilot
Complete DDT performance + fallback Level 4
Level 5
Highway traffic pilot
Complete DDT performance Level 3
Circa 2016
Future
Sustained lateral and longitudinal motion control Level 2
ACC+lane centering, parking/traffic jam assist
Sustained lateral or longitudinal motion control Level 1
ACC, parking assist (steering only)
Minimum operating speed, lane markings required
Minimum operating speed
Warning/intervention Level 0
LDW
BSW
ABS, ESC
←limited
Operational Design Domain (ODD)
unlimited →

Figure 1-1: Automation Level [3]

1-2 Perception of IV

Perception is a primary mission for IV including comprehensive tasks, visual, acoustic, etc. Autonomous control and dynamic driving tasks will be conducted based on the perceived environment. It is very similar to human's behaviour, for instance, grabbing. Our eyes perceive the environment and locate the target object. Then our body is controlled by the brain to grab the object. In the scenario of automated driving, visual task is the most important perception mission while acoustic perception is also under developing. The work in this thesis focuses on visual perception.

Various sensors adopted for perception will be introduced in Section 1-2-1. Then Section 1-2-2 will go deeper into visual modality obtained by commonly used sensors. Section 1-3 will elaborate different vision tasks, classification, location, segmentation, and discuss the difference between them.

1-2-1 Sensors

Perception of environment is a prerequisite for IV to operate autonomously. Radars are usually mounted around the vehicle to detect distances between other objects and itself.

Radar transmits radio pulse and receive the reflected one upon obstacles. Distance can be calculated out from the difference between transmitted and received wave frequencies using the theory of Doppler effect.

Lidar works in the same principal but uses narrow light pulse instead of radio pulse. Lidar also provides the position information. Radar and Lidar can detect objects but cannot recognize them as convenient as camera does. Recognition requires both detection and classification. For example, a pedestrian should be detected and classified to the class it belongs to. The work done in [23] fulfilled the task of pedestrian and cyclists recognition with radar based on Doppler effect associated to their movements. However, this method is limited to the velocities and accelerations of objects. Recognition with Lidar requires classification on each cloud point or features abstracted from the cloud map. The costing is high.

Camera is the perfect sensor to perform this recognition task [24]. It is low-costing and of high-level performance. Recognition by vision is important due to the fact that vehicle can learn to understand the environment by its eyes, the cameras actually, in a humanoid way. Vision makes it possible to classify which object is vehicle and which one is pedestrian while radar usually only carry out class agnostic detection. This visual perception facilitates vehicle's scene understanding ability.

Sensor fusion combines various formats of signals, e.g. camera, Lidar and radar [25], [26]. Sensor fusion is closely related to multi-modal network which usually yields better result compared to network with single modal signal input. Sensor fusion can benefit from the complementary advantages from different signals. This will be further explained in Chapter 3.

1-2-2 Visual modality

RGB image

RGB image is not a type of image generated by a certain device, for instance, camaera, but refers to colorful image presented by additive colors, red, green and blue. Therefore, RGB image is consisted of three channels corresponding to the three colors. The digital photos are usually RGB images. RGB images are the main source for training a deep neural network due to its low-costing and convenient collection method. It satisfies the requirement of deep learning for large amount of training data.

Depth image

Depth image is usually a single channel image containing information in the longitudinal direction in a scene. Depth can be inferred from disparity images where two cameras are employed to obtain two RGB images for the same scene. These two cameras are set parallel in the lateral direction with a distance called baseline. In this situation, images captured by the two cameras have a disparity for the same point because the optical centers are not the same. Obtaining depth in this way is cheap but less accurate. Another method of obataining depth is using Kinect depth sensor [27] where one camera and one infrared sensor are adopted. Kinect sensor is usually used for indoor scene understanding.

1-2-3 Vision task

Section 1-2-1 and section 1-2-2 discuss individual characteristics of radar, Lidar and camera and corresponding data model. In summary, camera is the most suitable sensor for scene understanding of intelligent vehicle. This section will introduce different types of vision tasks. Most prevalent ones are classification, object localization, semantic segmentation and instance-level segmentation as shown in Figure 1-2.

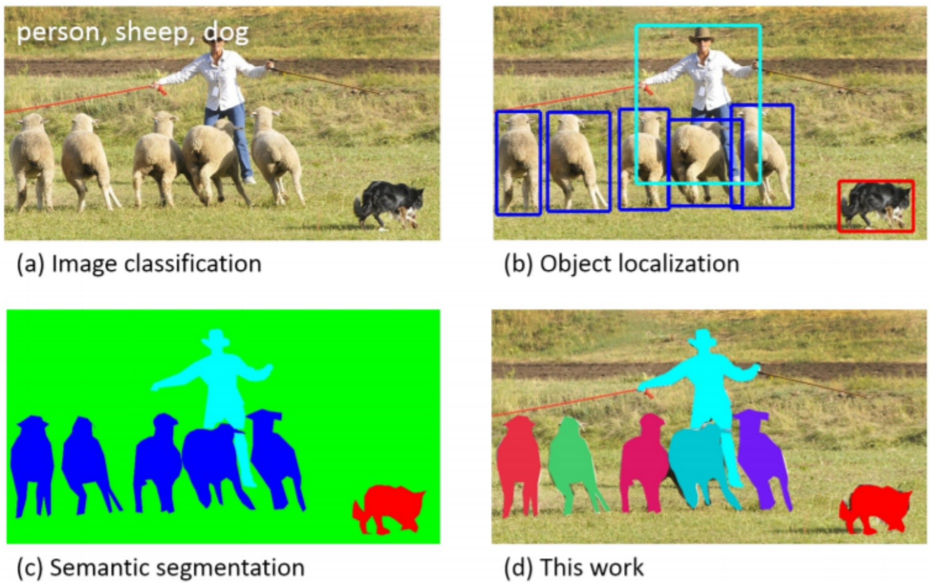


Figure 1-2: Difference between classification, detection, semantic segmentation and instance-level segmentation [4]

Image classification (Figure 1-2.a) only yields key words like 'sheep', 'person' to categorize this image into corresponding classes. Object localization or recognition (Figure 1-2.b) improves a bit by generating bounding boxes of different colors around different categorical objects. It is a combination of detection and classification. For example, green boxes for sheep and red boxes for human. It does not matter which color is used for which class as long as all the objects in the same class are marked by the same color.

Bounding boxes cannot describe the boundaries of objects precisely, which however, can be accomplished by semantic segmentation. Semantic segmentation (Figure 1-2.c) segments objects at pixel level, it cannot tell the difference between instances in the same class though. Instance segmentation provides more spatial and fine-grained information than semantic segmentation as shown in Figure 1-2.d.

1-3 Application of Instance Segmentation on IV

Figure 1-2 shows a living scenario where precise segmentation at instance level is not as necessary as in driving scenario. Autonomous driving need the spatial information to determine which instance is further away and which one is closer to adopt different avoiding strategies.

Under this circumstance, instance-level segmentation is necessary. It can make distinctions between one sheep and another or a vehicle and another vehicle. Unreliable results of vision tasks could be mortal for automated driving. This Section will present some application requirements of scene understanding for intelligent vehicle. After discussion in this section, the importance of instance-level segmentation and the real-time requirement for it will be clear.

1-3-1 Tackle occlusion

Figure 1-3 is a combination of various visual processing results [5]. The subfigure in the left bottom corner is the result of semantic segmentation where all pixels of cars are marked by the same color. It is difficult to tell which pixel belongs to which specific vehicle. The subfigure in the right bottom corner is the depth map and the middle bottom one is the final result of instance segmentation. The top images are the comparison between ground truth and predicted result. Depths are marked out above each bounding box. In this work [5], semantic segmentation, instance-level segmentation and depth are predicted parallel. We can see from the figures that instance segmentation provides us with much more spatial information and clearer scene parsing result than semantic segmentation. Specifically, when vehicles are parked in a line and occlude each other, instance-level segmentation still offers a meaningful segmentation result while all cars are clustered to a regiment of pixels in semantic segmentation.

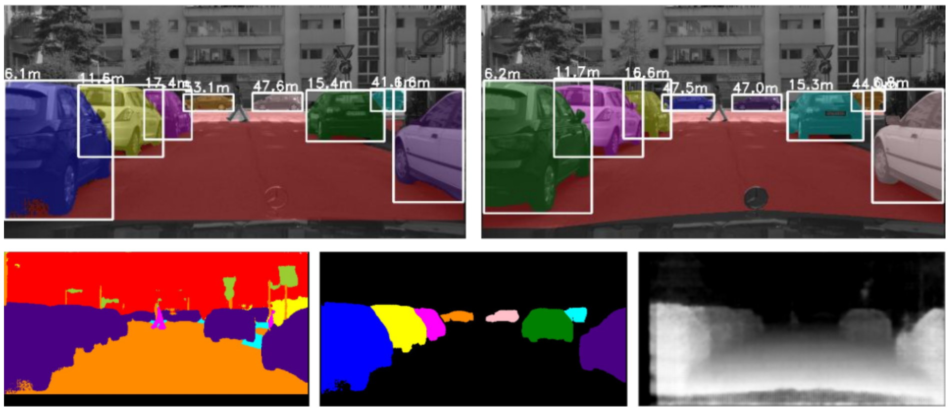


Figure 1-3: Driving Scenario [5]

1-3-2 Tracking

It is difficult to track all objects correctly all the time due to the overlaps between them. If all instances can be segmented at pixel level, it will be much easier to track them separately even with the existence of intersection and occlusion since it is rare for two instances to overlap with each other completely. So, we can track the independent pixels of each object. This cannot be achieved by semantic segmentation which does not describe the boundaries of two pedestrians or vehicles clearly if they are occluded as shown in left bottom sub-figure of Figure 1-3.

1-3-3 Real-time requirement

Object detection on intelligent vehicle can be performed almost in real time. YOLOv2 [28] processes images of resolution 544*544 at the speed of 40fps with mean Average Precision (mAP) of 78.6. mAP is the criterion used to evaluate detection or segmentation, refer to Section 3-3-1 for detailed information. Besides the benefits brought by instance segmentation, it is also required to be implemented on intelligent vehicle as efficient as object detection. Real-time performance is desired and some work have already been done. [1] adopts a clustering method based on a discriminative loss function aiming to improve efficiency.

1-4 Research Question

The importance of intelligent vehicle and their perception tasks have been discussed above. In conclusion, instance segmentation is the most suitable approach for urban scene understanding and reliable tracking. Most instance segmentation methods only leverage RGB images during training but sensor fusion provides more complementary information which inspires me to fuse two types of data to improve instance segmentation performance. Compared to point cloud image, depth image is easy to be accessed and low-costing. Therefore, the research question is: Does incorporating depth into RGB information increase instance segmentation accuracy?

More specific research questions can be derived:

1. What are the impacts of parameters involved in instance segmentation?
2. What is the best way to incorporate depth information into original RGB network? Does data fusion work better or is there other superior method?

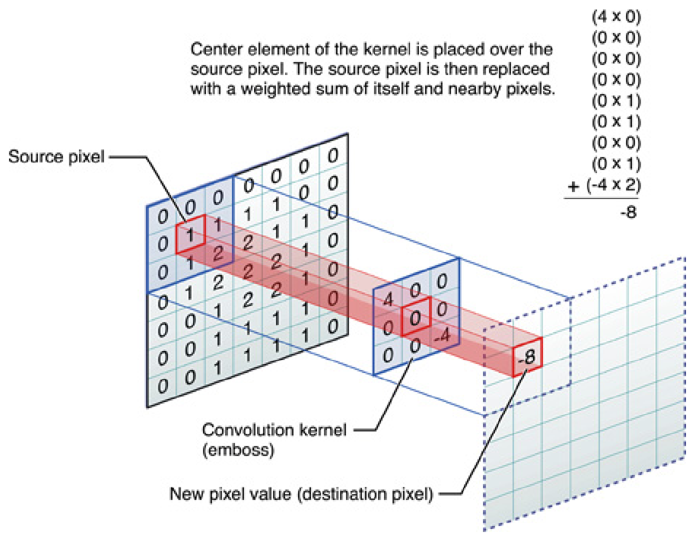
Chapter 2

Related Work

Convolutional Neural Network(CNN) is the state-of-the-art method parsing images and segmenting objects. This chapter is formed by a progressive order from the introduction and development of CNN (Section 2-1) through semantic segmentation (Section 2-2) to instance-level segmentation (Section 2-3). Section 2-4 will introduce the concept of multi-modal networks. Creative contributions of this work will be presented at the end in Section 2-5.

2-1 Convolutional Neural Network

Hubel and Wiesel found that there are neurons responding to stimuli independently on the visual cortex. Inspired by their work, Neocognitron [29] was introduced in 1980. LeNet [30] which came out in 1998 can be considered as the true beginning of CNN. It is a 7-layer network with only two convolutional layers used for classifying hand written digits.



Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)

Figure 2-1: Convolutional operation [6]

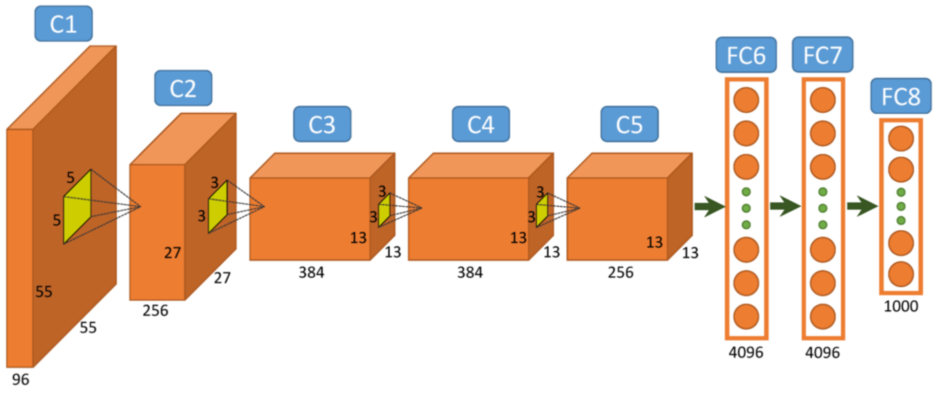


Figure 2-2: AlexNet[1]https://www.saagie.com/blog/object-detection-part1

Figure 2-1 shows the working principal of convolutional operation. Convolution is actually a weighted sum up of a source pixel and pixels around it. For each pixel in the input image, it is multiplied by the central weight in the convolution kernel and the nearby pixels are also weighted by the corresponding values in the kernel. This kernel is applied to all the pixels in a sliding-window way.

Convolutional layers are used to extract features and preserve local(in the receptive field) spatial structures. Sub-sampling(pooling) layers discard redundant information to reduce computation cost. Convolutional and pooling layers together consist the feature extraction part of CNN. Fully connected layers combine those extracted features non-linearly to get better classification result. After LeNet, CNN has been neglected for a long period. Its performance was worse than other classifiers such as SVM using manually designed features. The historical break occurred in 2012 when AlexNet [21] performed amazingly in ImageNet Large Scale Visual Recognition Competition (ILSVRC). AlexNet consists of 5 convolutional layers and 3 fully connected layer as shown in Figure 2-2. ReLU, a nonlinear activation function, was adopted. Dropout and Data augmentation were used to prevent overfitting. Besides these, large scale dataset(ImageNet) and the development of Graphics Processing Unit (GPU) creates possibility to fully train a CNN. All these factors contributed to the revival of CNN.

2-2 Semantic Segmentation

Semantic segmentation has a preciser performance than object detection where only bounding boxes are drawn around the detected objects while the detected objects are classified at pixel-level in semantic segmentation. This means, the segmented output image is of the same size as the input image. However, after convolutional and pooling layers, the input images are down-sampled to a smaller size. Then how to up-sample the down-sampled image to its original size properly becomes the main issue. Following this clue, Fully Convolutional Network(FCN) is the first reliable network performing this task. FCN and FCN-based approaches will be discussed in Section 2-2-1. After that, an encoder-decoder structure (Section 2-2-2) was put forward to up-sample images more precisely.

2-2-1 FCN-based methods

Fully Convolutional Network (FCN) [7] is the first reliable pixel-wise prediction neural network for semantic segmentation by reinterpreting classification networks such as AlexNet and VGG16 [31]. In their work, the fully connected layers playing the role of classification are substituted by fully convolutional layers (see Figure 2-3), so the direct output of the fully convolutional part is a heat map ideal for segmentation. Afterwards the coarse heat map is upsampled to obtain the dense pixel-wise prediction.

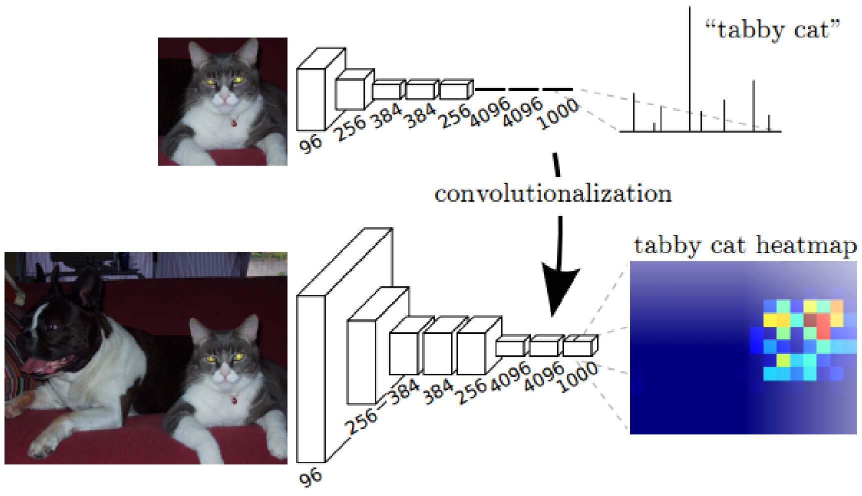


Figure 2-3: Fully Convolutional Network [7]

However, the dense prediction obtained by only up-sampling the heat map from the last convolutional layer is coarse. In a network, usually features learned at low level layers are fine but contain less semantic information while high-level features are coarse but semantic. Therefore, the authors propose to concatenate heat maps from different convolutional levels as show in Figure 2-4.

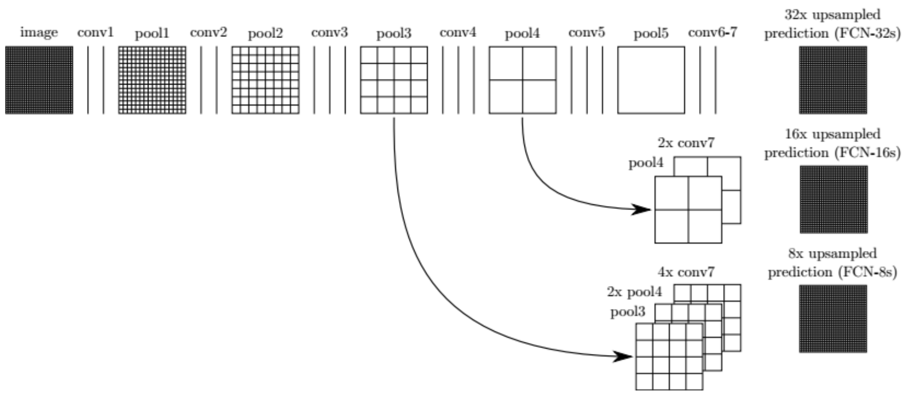


image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled prediction (FCN-32s)
2x conv7
pool4
16x upsampled prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled prediction (FCN-8s)

Figure 2-4: Combination of features from different layers
First stream: Single stream by up-sampling the output from the final layer. Second stream: Leveraging both features from final layer and 4th layer. Third stream: Features at 3rd layer is further added to enhance precision [7]

2-2-2 Encoder-Decoder structure

Another main stream semantic segmentation structure is encoder-decoder. Encoder-decoder is originated from auto-encoder which is an simple unsupervised learning approach. It is mainly used to preprocess images to obtain a pre-trained encoder. This pre-trained encoder part can be fine-tuned for other supervised learning tasks in which way fewer labeled training data is required. Figure 2-5 is the encoder-decoder structure of SegNet, an efficient semantic segmentation network mentioned in Section 1-3-3.

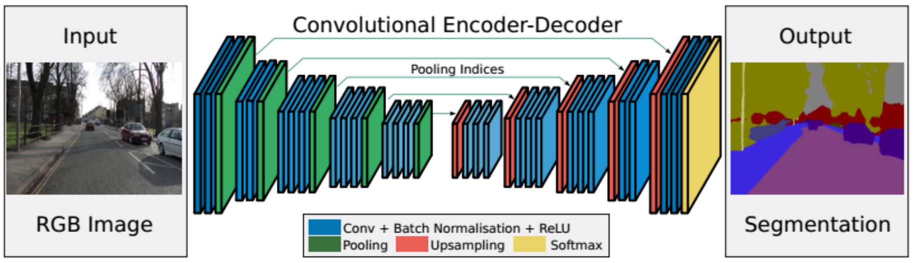


Figure 2-5: SegNet structure [8][9]

DeconvNet [32] is the first network adopting the encoder-decoder structure. The encoder part is topologically same to the convolutional part in backbone network, for example, VGG. The decoder part learns to map the feature map back to its original input size gradually instead of applying simple interpolation. Decoder part is learned together with the encoder part.

SegNet discards all fully connected layers to save time for training as shown in Figure 2-5. Different from DeconvNet, a maxpooling indices strategy is used for the decoder to refer to all feature maps in the encoder. So the decoder part leverages feature maps of different resolution. It combines low-level fine-grained details and high-level semantic information in a similar way to the FCN combination method. Each encoder corresponds to a decoder as a match. The decoder part up-samples the image back to its original size gradually. The first 13 convolutional layers in the encoder are constructed based on VGG16, the backbone network. Softmax is used to activate the last layer to classify pixels. The direct output from SegNet is already good enough without any post processing. Later on, ENet was designed based on this encoder-decoder structure but was more compact for the purpose of real-time operation.

2-3 Instance Segmentation

Instance-level segmentation has similar principle and characteristics as semantic segmentation discussed in Section 2-2. At the very beginning, instance segmentation can be considered as a combination of semantic segmentation and object detection. Segmentation is performed on the proposed bounding boxes. This conventional yet still popular approach will be explained in Section 2-3-1. Another proposal-based approach is directly predicting the masks of each instances omitting the procedure of object detection rendering instance segmentation more efficient as shown in Section 2-3-2. Recurrent methods and loss function based approach will be presented in Section 2-3-3 and 2-3-4.

2-3-1 Bounding-box proposal methods

Generally speaking, the quality of instance segmentation with object detection as foundation is highly dependent on the accuracy of detection. Segmentation is performed as a refinement in the detected boxes/proposed Region of Interest (RoI). Two main streams of network structure will be introduced in this section, namely Region-based Convolutional Neural Network (R-CNN) [33] [34] [35] [10] approach and FCN approach [7] [36] [12].

R-CNN stream

R-CNN [33] is an excellent frame for object detection which contains three main steps. First, it generates a set of regional proposals (bounding boxes) and feed them into a CNN to extract features. Then a SVM layer is attached to the network as classifier to identity the classes for these bounding boxes. An additional part is 'box regression' which generates tighter bounding boxes for the objects through a regression model. R-CNN is not an end-to-end structure but adopts different networks for classification and box regression. However, Girshick, the first author of R-CNN, found out that there were large amount of overlapping patches between the RoIs so he designed a new structure, RoI pooling [34] where features are extracted for the whole image from which RoIs were extracted. Fast R-CNN also combines the extractor, classifier and regressor in the same network. After these improvements, a problem is still noticeable: RoI is generated inefficiently by selective search [37]. Region Proposal Network (RPN) brought forward in Faster R-CNN [35] made a difference in 2016. RPN and classifier share the same features generated from CNN. In 2017, Kaiming He and his team created Mask R-CNN for instance segmentation based on faster R-CNN. Mask R-CNN has two branches, on is the FCN branch to obtain binary pixel-wise masks for objects and the other one is faster R-CNN branch for detection and classification, see Figrue 2-6.

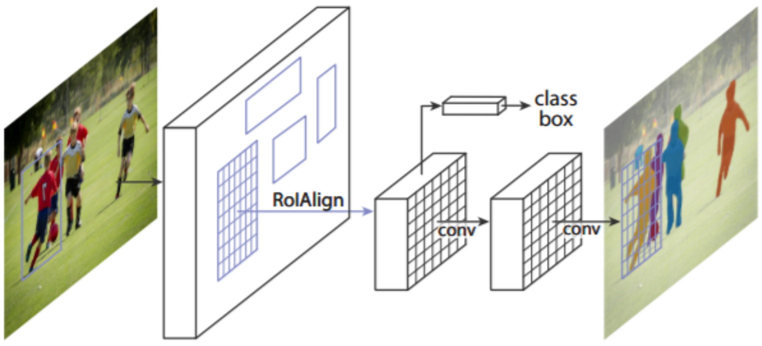


Figure 2-6: Mask R-CNN
RoIAlign is a preciser pooling method with bilinear interpolation to avoid misalignment in RoIPool. 'class box' classifies the category of an object. [10]

FCN stream

Instance-sensitive Fully Convolutional Networks (Instance-FCN) [11] and Fully Convolutional Instance-aware Semantic Segmentation (FCIS) [12] are the two main approaches for instance segmentation based on FCN. Instance-FCN is actually a mask proposal method so it should be elaborated in Section 2-3-2. However, FCIS leverages the concept of instance-FCN so it is included in this Section. The origin of mask proposal methods will be presented in Section 2-3-2. Section 2-2-1 elaborates how FCN works for semantic segmentation. Instance-FCN is inspired by this semantic segmentation structure. Is it possible to generate heat maps similar to semantic heat maps but can make a difference between instances? To tackle this question, the authors introduce an instance-sensitive score map where each pixel is a 'classifier of relative positions of instances' instead of a 'classifier for object category'. As shown in Figure 2-7, k^2 pixel-wise instance-sensitive score maps are learned for each image. Accordingly, each pixel is represented by k^2 channels. These k^2 score maps are assembled to get the 'all-instances' score map. Another branch of Instance-FCN is the prediction of objectness scores which is another score map to identity whether a pixel belongs to an object or not. The two branches are trained together with the loss function 2-1. This loss function contains two parts for 'objectness score' prediction and 'all instances score' prediction separately. i is the index of one of the windows, j is the index of pixel in the window. p_i is the predicted objectness score and p_i^* is the groundtruth label. $S_{i,j}$ is the pixel after assembling and $S_{i,j}^*$ is the groundtruth pixel.

$$\sum_i (L(p_i, p_i^*) + \sum_j (L(S_{i,j}, S_{i,j}^*))) \tag{2-1}$$

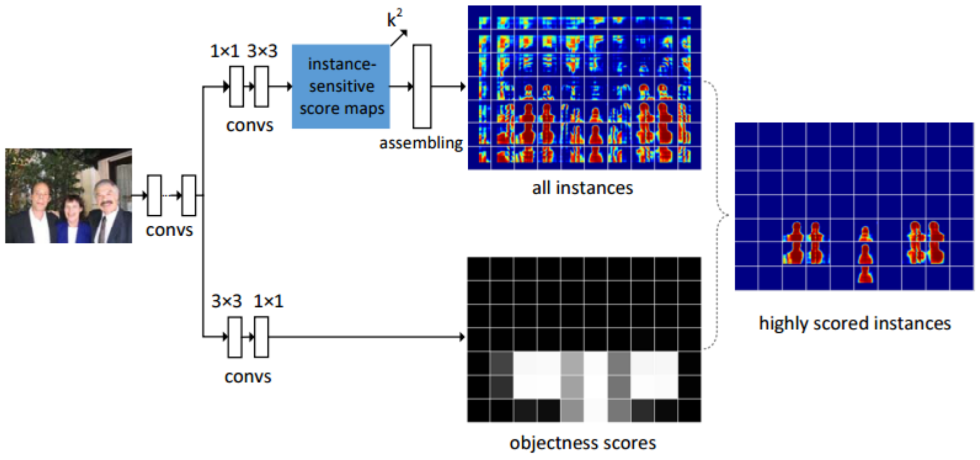


Figure 2-7: Instance-FCN
The top branch is the prediction of instance-sensitive score maps and assembling procedure. The bottom branch is the prediction of objectness scores which identifies whether a pixel belongs to an instance or not. [11]

The author of FCIS stated that instance-FCN is inefficient at generating instance-sensitive score maps result from using certain size of square sliding windows which has to refer to image pyramid scaling to find various scaled objects. Besides, instance-FCN performs classification independent from mask proposal so it is not an end-to-end method. Hence, FCIS incorporates RPN into mask proposal procedure as shown in Figure 2-8. Moreover, the instance-sensitive score map used in instance-FCN is adapted to position-sensitive inside/outside score map which makes a further distinguishment on whether a pixel locates inside an instance or not. Then mask prediction and classification are jointly performed.

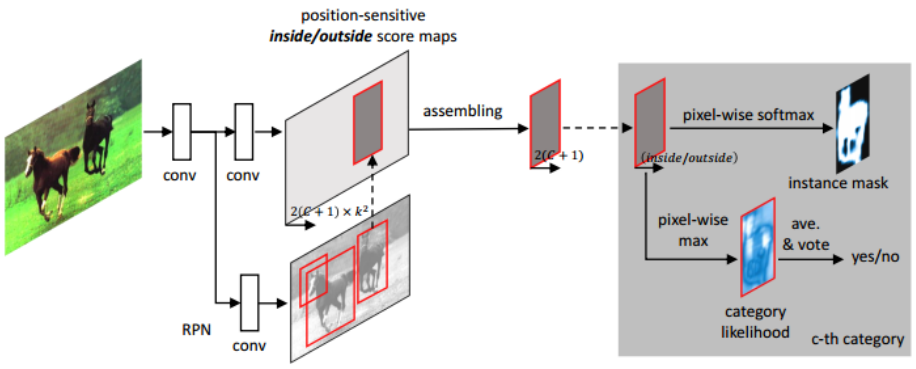


Figure 2-8: Network structure of FCIS
RPN generates bounding box proposals and apply these RoIs on the position-sensitive inside/outside score maps as a substitution to sliding window selection implemented in instance-FCN. Afterwards, mask prediction and classification are performed jointly as shown in the gray area. [12]

2-3-2 Mask proposal methods

As mentioned in Section 2-3-1, instance-FCN is actually a mask proposal method and it is originated from DeepMask [13], Figure 2-9. DeepMask does not process the whole image as an unit but processes different patches of an image densely at various scales to find only one instance at the center of the patch each time. For each patch, a mask and object score are predicted from the shared features. This operation is quite similar to instance-FCN. The only difference is that instance-FCN extract features for the whole image and perform prediction on the image once and for all. Although it also need to be applied at various scales which is further solved by RPN in FCIS.

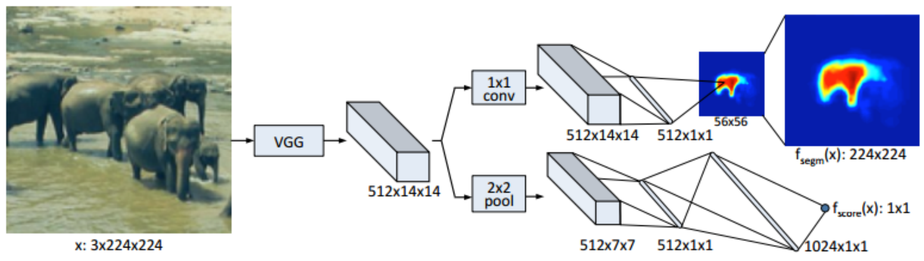


Figure 2-9: Structure of Deep Mask
The top stream is mask prediction and the bottom stream is object score prediction. Features are extracted by VGG and shared by the two prediction tasks later. [13]

SharpMask [38] is a refinement method based on DeepMask. Section 2-2-1 has explained the relations between features and layers. Low-level features are fine but less semantic while high-level features are coarse but contain meaningful information. So FCN adopts a skip strategy to concatenate feature maps from different layers, Figure 2-4. A preciser refinement method is proposed in SharpMask where a top down refinement branch is added to DeepMask. This top down refinement branch works at several connection parts of convolutional layers. Mask encoding from previous layer and extracted feature map from next layer are downsampled and concatenated to maintain fine-grained details.

2-3-3 Recurrent methods

Recurrent Neural Network (RNN) is inspired by the principle of human visual detection. Human detects objects sequentially by storing the location of already counted objects in brain [39] [40]. Accordingly, RNN not only takes in input from a certain time stage but also preserves the output from last time stage and involve it in current learning. Long short-term memory network [41] is a special form of recurrent neural networks that is tremendously applied in various tasks [42] [43] [44]. LSTM is prevalent in natural language processing which requires consecutive logic between words. Now it has been adopted for instance segmentation to segment out objects one by one.

Recurrent instance segmentation [14] is a combination of FCN and RNN, see Figure 2-10. FCN extracts heat map from the input image and feed it to RNN to obtain binary masks for different instances one by one.

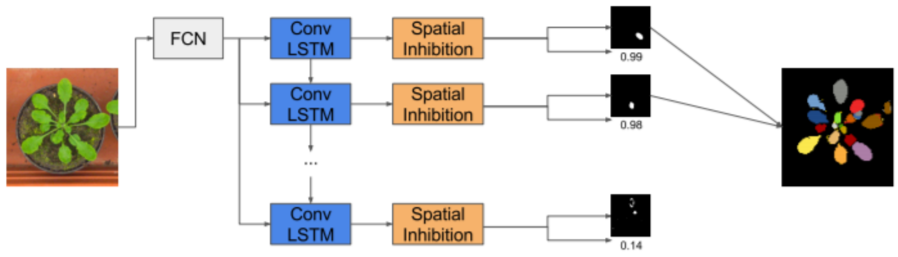


Figure 2-10: Recurrent Instance Segmentation
Binary masks for instances are obtained sequentially. Numbers under these masks are confident scores. This iteration process will stop when the confidence score drops below the threshold. [14]

2-3-4 Other methods

Instance segmentation with a disriminative loss function [1] is a novel method inspired by the works [45], [46], [47], [48], [49]. This method does not involve any proposal or recurrent iteration but is only based on a discriminative loss function similar to Fisher criterion. This discriminative loss function can be plugged in any off-the-shelf semantic segmentation network to learn a n-D embeddings for all instances. Embeddings of the same instance are pulled together while embeddings of difference instances are pushed away from each other. Inference procedure is performed with simple meanshift clustering algorithm [50]. Time consumption of inference procedure can be ignored stated in the paper. This method is the cornerstone of this thesis so it will be well explained in Chapter 3. This instance segmentation method with a discriminative loss function is a plug-in approach. Any semantic segmentation network could be adopted to generate instance segmentation masks by substituting the original loss function with this novel discriminative loss function. In this way, no additional operation is required to predict bounding boxes or other preliminary results before getting the final instance-level

Table 2-1: Performance comparison between SegNet and ENet[20]

	NVIDIA TX1						**NVIDIA Titan X**					
Model	480*320		640*360		1280*720		640*360		1280*720		1920*1080	
	ms	**fps**	**ms**	**fps**	**ms**	**fps**	**ms**	**fps**	**ms**	**fps**	**ms**	**fps**
SegNet	757	1.3	1251	0.8	-	-	69	14.6	289	3.5	637	16
ENet	47	21.1	69	14.6	262	3.8	7	135.4	21	46.8	46	21.6

segmentation. The efficiency is then affected by the semantic segmentation network. So a review on semantic segmentation network is necessary.

Table 2-1 compares the performance of SegNet [8] [9] and ENet[20], two most efficient networks for semantic segmentation. It is obvious that image resolution has a great compact on the processing speed. Low resolution saves time while high resolution preserves more fine-grained details in the image.

Depth aware instance segmentation [2] is designed with the object to avoid misclassification at the overlapped parts of instances. It has two branches as shown in Figure 2-11. One branch predicts instance segmentation masks and the other branch predicts depth image. Instances in the mask are arranged by predicted depth order. Pixels belonging to the overlapped parts are assigned to the instance closer to the camera by estimated depth. Therefore, false semantic segmentation such as segment the leg of horseman as part of the horse will be partially solved. If predicted depth is substituted by groundtruth depth, accuracy should be further improved.

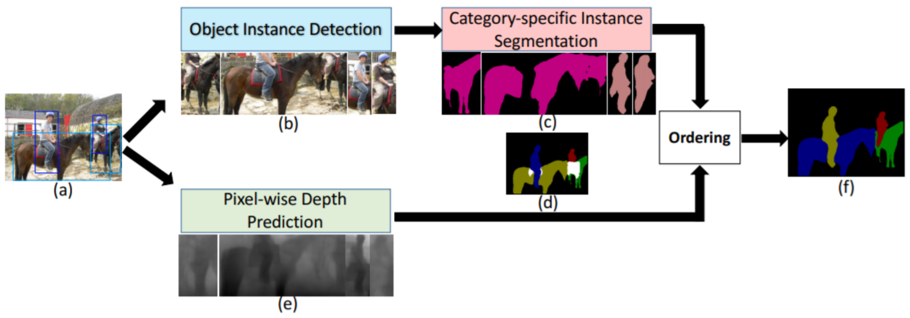

Object Instance Detection
Category-specific Instance Segmentation
Ordering
Pixel-wise Depth Prediction
(a)
(b)
(c)
(d)
(e)
(f)

Figure 2-11: Depth aware instance segmentation
The top branch is instance segmentation mask prediction and bottom is depth prediction. [2]

2-4 Multi-modal Network

The work in depth-aware instance segmentation [2] involves prediction of depth for monocular images. Then can we leverage existing depth information to improve segmentation performance? RGB images do not contain explicit coordinate information, though implicit depth information can be inferred from RGB images by deep learning. If we have available depth image or point cloud from Lidar, then probably fusion of two types of data will produce result better than the state-of-the-art. How to combine the additional information with conventional RGB network efficiently still remains a challenge. Fusion is the most ordinary approach to achieve the goal as shown in Figure 2-12. Two networks receive RGB images and depth images as input independently and fuse the feature maps (inputs) at a certain stage. Preprocessing of depth is necessary to adapt single channel depth image to 3-channel depth image for existing networks. One novel method is coloring which colors depth information to form a RGB image [51].

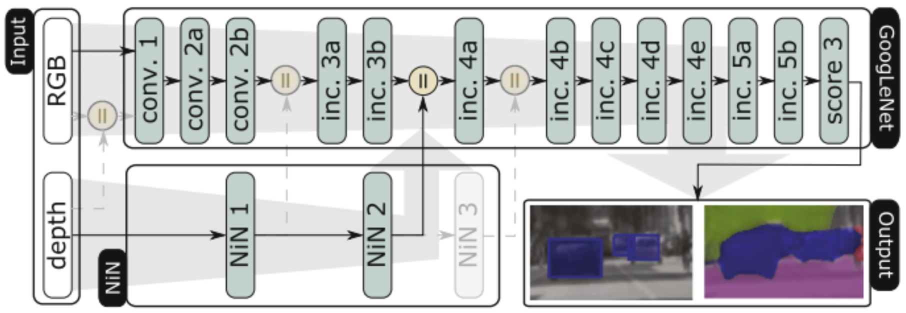


Figure 2-12: Fusion at different level [15]
The left dash line refers to early fusion which is direct fusion of inputs. The dash line in the middle represents mid fusion. The right dash line is the late fusion where outputs from two networks are fused before loss function.

2-5 Contribution

As discussed in Section 1-3-3, real-time application is significant for automated driving. For this purpose, the method of instance segmentation with discriminative loss function [1] without any proposal or recurrent procedures is of many interests in this research. However, segmentation with a discriminative loss function does not yield ideal instance masks as shown in the example pictures from their paper, see Figure 2-13. Obviously, the two nearest cars are clustered as one. In fact, even two cars that are far away geographically could also be clustered as one car if their n-D embeddings are not repulsed away from each other. Meanshhift clustering will process them as one instance.

Depth-aware instance segmentation inspires me to explore the possibility of adding depth as the $(n+1)_{th}$ dimension to RGB n-D embbedings mentioned in the work of discriminative loss function. In depth-aware instance segmentation, depth is introduced at the end of the branch for instance mask prediction. Usually pixels at boundary are ambiguous to be reliably identified. Depth works as a refinement arranging the order of instances. It provides spatial information to further classify that to which instance should a certain pixel regiment belong.

Therefore, this research focuses on incorporating depth information into RGB embeddings where depth plays the role of additional separable dimension(s) to improve the clustering reliability on the overall embeddings.

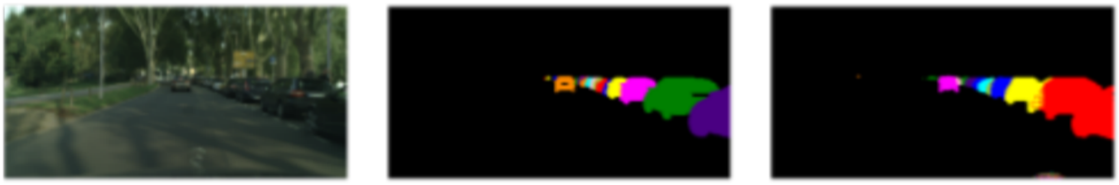


Figure 2-13: Example for car class presented in the paper [1]
The left image is the original image and the middle one is the groundtruth for car class. Prediction of instance mask for car class is on the right.

Three methods are introduced to incorporate depth into RGB network, namely **scaling**, **fusion** and **concatenation**. In scaling, depth is added directly to the n-D embeddings as an additional dimension with the object to distinguish possible overlaps in n-D embeddings. Under this circumstance, depth image is not fed in any networks but only scaled by a learned scaling parameter. Another method is encodding depth image into 1D embeddings similar to the branch of RGB network. The semantic segmentation network can also learn a 1D embeddings for depth image with the same discriminative loss function used for RGB embeddings. This 1D representation will be fused with RGB network before loss function (**fusion**) or concatenated directly to the output, the embeddings, of RGB networks (**concatenation**).

Preprocessing of depth image to adapt it from single channel image to 3-channel image is necessary for the reason that popular networks are only designed to take in 3-channel RGB images. The most intuitive and simple way is to replicate single channel depth image to 3 channels. Instead of replicating depth image from single channel to 3 channels, the processing method proposed in [51] colors depth image with a 'jet color map' to make it a RGB image which does not contribute much in this work.

In view of the focus of this thesis, only Cityscapes [17] dataset is employed to analyze urban street scenes. Cityscapes dataset will be elaborated in Section 4-1.

These research questions are closely related to experiments so they will be examined in Chapter 4. Results will be discussed in Chapter 5. This work achieved significant improvement on Cityscapes dataset compared to the baseline work, semantic segmentation with a discriminative loss function [1].

Chapter 3

Methodology

The basis of the work introduced in this thesis is reproducing the method mentioned in [1] which will be presented in Section 3-1. Only a single modal RGB network is involved in this basic work. Then the creativity of incorporating depth into this network will be elaborated in Section 3-2.

3-1 Baseline

Instance segmentation with a discriminative loss function is comprised of two parts, learning to encode instances in a desired way and inference. The first part can be learned by any semantic segmentation network with a discriminative loss function which will be introduced in Section 3-1-1. Inference or the post-processing procedure, is based on center thresholding or meanshift clustering method. Center thresholding leverages ground truth information while meanshift is an unsupervised clustering approach. Both of these two clustering methods will be discussed in Section 3-1-2. Encoding procedure and inference procedure can be formulated as 3-1 and 3-2 where p_i denotes a certain RGB pixel and the footnote refers to its index. E is the encoding function which encodes the RGB pixel p_i into embedding e_i with the discriminative loss function. Inference procedure is represented by I. Inferred instance index for embedding e_i is denoted by i_i .

$$e_i = E(p_i) \qquad \text{(Encoding RGB)} \qquad (3\text{-}1)$$

$$i_i = I(e_i) = I(E(p_i)) \qquad \text{(Inference)} \qquad (3\text{-}2)$$

3-1-1 Encoding with discriminative loss function

This novel discriminative loss function is very similar to a Linear Discriminant Analysis (LDA) criterion, Fisher criterion. Fisher criterion is a very intuitive way to classify two classes, see equation 3-3 [52]. Fisher criterion is designed with the object to maximize the the distance between the means of two classes while minimize the distances between different samples within the same class. In equation 3-3, m refers to the means of two classes and s represents the variance of a certain class. Variance measures the distances between samples in a class. LDA learns to maximize this Fisher Criterion to discriminate the two classes.

$$J(w) = \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2} \tag{3-3}$$

Different from Fisher criterion, the discriminative loss function in this work does not make distinguishment between two classes. The idea is rendering the semantic segmentation network to learn a n-D embeddings for different instances at pixel-level. Each pixel in the image is encoded into n dimensions. For example, if it is 2D embeddings, a RGB image of dimension $3 \times 256 \times 512$ will be encoded to $2 \times 256 \times 512$. In this 2D feature map, embeddings of different instances repulse each other while embeddings of the same instance are attracted towards their own centers as shown in Figure 3-1. Repulsion and attraction can be expressed as the result of virtual push force and pull force. Pull force pulls embeddings of all pixels in one instance to accumulate around a certain point which is similar to the variance term in Fisher criterion. Push force corresponds to the mean term in Fisher criterion where only the centers of instance embeddings are pushed away from each other instead of applying push force on all pixel embeddings. Computation and time costing in this way are much cheaper especially when large amount of instances are involved in images. This is beneficial for Cityscapes dataset where many vehicles usually appear in a sequence.

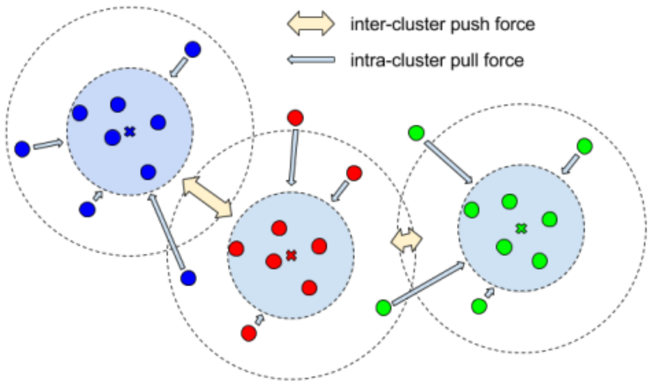


Figure 3-1: 2D Illustration for discriminative loss function
Color blue, red and green represent 2D embeddings of three different instances separately. These colors are random without any specific meaning. Small dash circle is the in-margin for pull force and large dash circle marks the out-margin for push force. In-margin and out-margin are 0.5 and 1.5 separately.[1]

Another characteristic of this loss function is that there are limitations on push and pull force realized by setting proper out-margin and in-margin. Embeddings that have already been attracted within the in-margin of its own center will not be pulled anymore. Similarly, centers will stop pushing each other if the distance between any two centers is larger than twice the out-margin, see the small and large dash circles in Figure 3-1. In-margin and out-margin are

0.5 and 1.5 for all instances. There is no regular pattern for how the network learns to arrange these embeddings. After the embeddings of different instances have been separated from each other, they can move freely in this 2D feature space. This discriminative loss function can be formulated in equations as follows [1]. It is consisted of three components, variance, distance and regularization terms. μ_c refers to different embedding centers of instances and N_C is the number of instances in a certain class. C represents different classes. δ_v and δ_d are the in-margin and out-margin separately. α, β and γ are regularization elements. Values are set as: $\alpha = \beta = 1$, $\gamma = 0.001$

$$L_{var} = \frac{1}{C}\sum_{c=1}^{C}(\frac{1}{N_C}\sum_{i=1}^{N_c}[||\mu_c - e_i||-\delta_v]_+^2) \tag{3-4}$$

$$L_{dist} = \frac{1}{C(C-1)}\sum_{c_A=1}^{C}\sum_{c_B=1}^{C}[2\delta_d - ||\mu_{C_A} - \mu_{c_B}||]_+^2 \tag{3-5}$$

$$L_{reg} = \frac{1}{C}\sum_{c=1}^{C}||\mu_c|| \tag{3-6}$$

$$L = \alpha \times L_{var} + \beta \times L_{dist} + \gamma \times L_{reg} \tag{3-7}$$

The original work choose a 8D embedding feature space which does yield better result compared to 2D embeddings according to experiments. Impact of embeddings' dimension will be discussed in Chapter 4. However, preliminary work are done with 2D embeddings which has an advantage on visualization.

This discriminative loss function is class agnostic which means it cannot classify the category of a certain instance. The only object is to arrange instances in one class as separately as possible in the feature space. However, instances from different classes can occupy the same location in the feature space. Therefore, an additional classification procedure is necessary for this method. In this case, semantic segmentation mask is required for classifying all instances. The quality of semantic segmentation mask will affect the performance to a large extent. Due to the fact that the network adopted to learn embeddings should be semantic segmentation network, it can generate semantic segmentation masks itself for this classification procedure. However, ground truth semantic segmentation presents much higher AP for the final results. Table 3-1 compares the performance of two types of semantic masks on Cityscapes [17], one is generated by resnet38 [53], the other one is ground truth semantic mask.

Two clustering methods are also compared which does not contribute to the final result as much as the quality of semantic mask does. As mentioned at the beginning of this chapter, mean-shift is an unsupervised clustering method while center threshold leverages instance-level ground truth information. Both of them will be introduced in next Section 3-1-2.

semantic mask	clustering method	AP
resnet38	mean-shift	21.4
resnet38	center threshold	22.9
ground truth	mean-shift	37.5
ground truth	center threshold	47.8

Table 3-1: Comparison of different semantic masks and clustering methods on Cityscapes dataset [17][1]
The first column lists two types of semantic masks, one is obtained by resnet38 and the other one is ground truth labeling. The second column compares mean-shift and center threshold as clustering methods. Ground truth semantic mask improves AP from 21.4 to 37.5 using mean-shift clustering method. The improvement is more significant with center threshold clustering. 24.9% increase is reported.

This classification procedure is independent of the integral training procedure. The training flow is still in an end-to-end fashion. The semantic segmentation mask generated by different semantic networks in different experiments can vary a lot even if the learning parameters are the same. So classification with self-generated semantic mask is not a controllable process. The object of this work is to compare results between single modal RGB network and multi-modal networks with depth as additional input. Under this circumstance, only ground truth semantic masks are employed in all the experiments later to control the part of fluctuation in AP caused by the quality of semantic masks.

3-1-2 Post-processing

Inference process is required to obtain instance segmentation masks from learned n-D embeddings. This can be achieved by center threshold clustering or mean-shift clustering method. **Bandwidth**, the threshold for clustering, is the most important parameter involved in this instance segmentation method. It has a significant impact on clustering results which will be examined in experiments.

Center threshold

Embeddings of pixels in one instance are accumulated around a center (mean) within the in-margin while embeddings of pixels belonging to different instances are far apart. So, ideally it is convenient to infer instances from this embedddings feature map if there is a threshold (**bandwidth**) around a certain center (mean) where embeddings with variance lower than the threshold belong to this instance while embeddings with variance larger than the threshold belong to other instances. During learning procedure, in-margin and out-margin are 0.5 and 1.5 which means embeddings should all accumulate in a circle with radius 0.5 centered at their own means. However, the learning process is stochastic that some points may lay outside the in-margin. Under this circumstance, the bandwidth should be set as a value range from in-margin to out-margin, namely 0.5 to 1.5. Centers are located by ground truth information in this clustering method.

Mean-shift

In real operation scenarios, no ground truth information is available. Therefore, unsupervised clustering approach is required to substitute center threshold clustering method. Mean-shift comes to hand in this case. Mean-shift is an iterative approach. In each round, it starts with a random point as mean in the feature map and classify all points with variance lower than a threshold (**bandwidth**) in the feature map as embeddings belonging to this instance. Then the mean of selected points is chosen as new starting point for the next iteration until it converges as shown in Figure 3-2. Converging condition is that the difference between the current mean and the new mean in next iteration is smaller than 0.0001. If iteration has exceeded 1000 times but not converged yet, mean-shift will stop this round and go to next one. Clustered points in each round will be removed from the feature map before going to the next round. Only one instance is clustered in each round. Inference is finished until the number of unclustered embeddings is less than 100. This is a criterion adopted in original work [1].

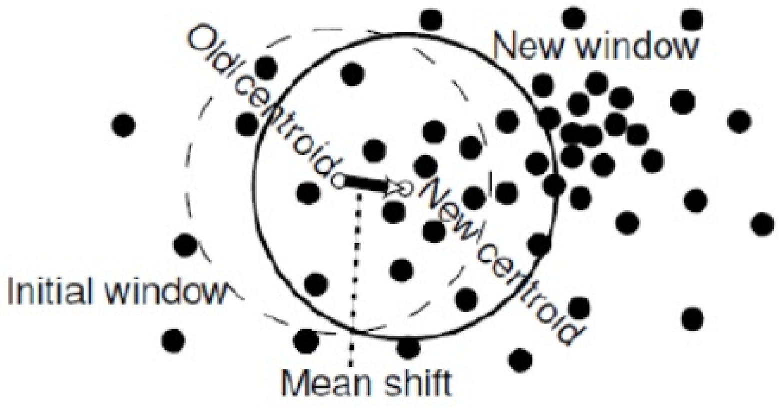


Figure 3-2: mean-shift iteration
Old centroid and new centroid refer to old mean and new mean. Initial window and new window refer to areas inside bandwidth around old mean and new mean. [16]

3-1-3 Visualization

As what mentioned above, embeddings learned by this discriminative loss function is class agnostic so instances from different classes can occupy the same place in feature map. To better visualize the learning result and post processing result, only car class, the most typical one, is discussed. If multiple classes are included in the same feature map, it is highly possible that many instances from different classes are accumulated together. For the purpose of visualization, only 2D embeddings is discussed in this Chapter.

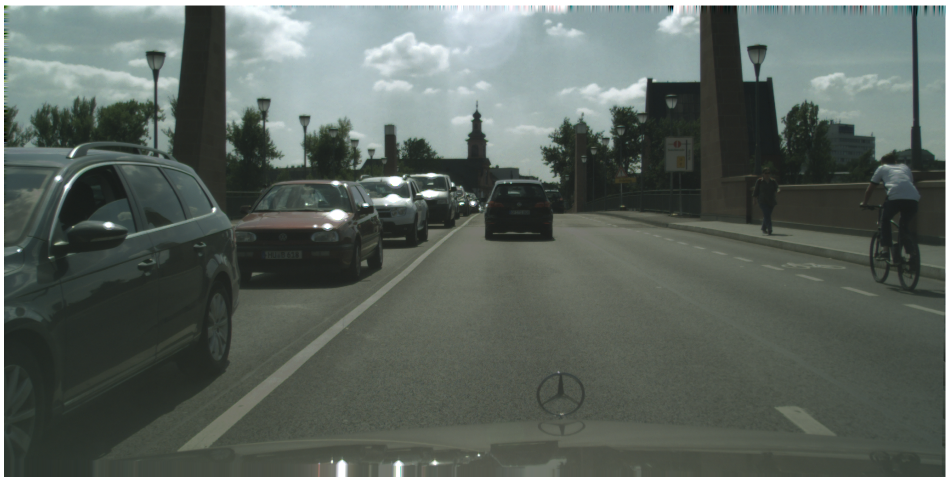


Figure 3-3: Sample image
This sample image is a frame from validation dataset

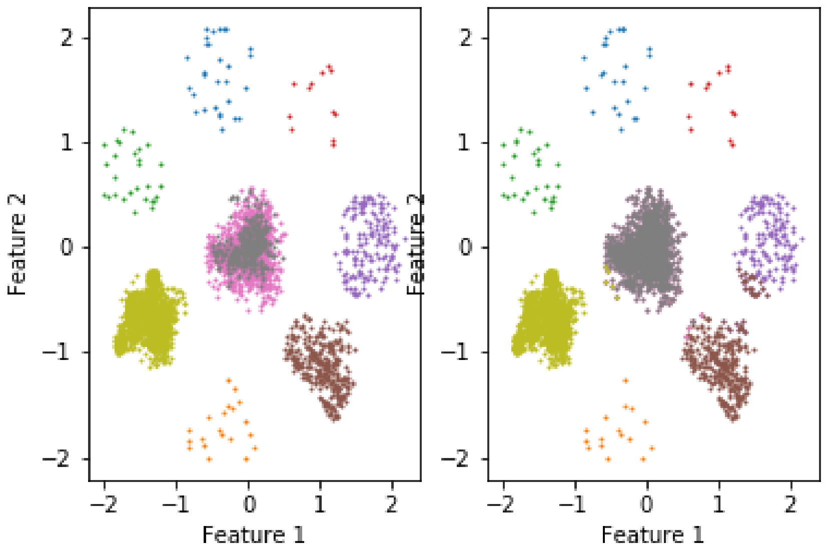


Figure 3-4: 2D embeddings for car class
The left feature map shows ground truth embeddings for all pixels in this 2D space. The sub-figure on the right is clustering result obtained by center threshold. Each color represents an instance in car class. Color are given randomly.

Figure 3-4 presents the learning result and post processing result for frame 3-3. The left sub-figure is the ground truth embeddings learned by ENet with discriminative loss function. Each instance is given a random color in this 2D feature space. In the middle, embeddings of two car instances overlap with each other. The right sub figure is the clustering result by center threshold. Though centers of the two overlapped instances can be located by the ground truth information, they cannot be separated from each other with similar embeddings in this feature map.

Mean-shift result is presented in Figure 3-5 where embeddings in the top right corner is missing due to non converged iteration with too few pixels available. In the middle, embeddings of two overlapped car instances are clustered as one. The results of misclassification in instance mask is shown in Figure 3-6. Two instances are classified as one that both of them are colored as gray in the mask (each instance is given a different color). There is no correlations between the colors used in feature map and instance mask.

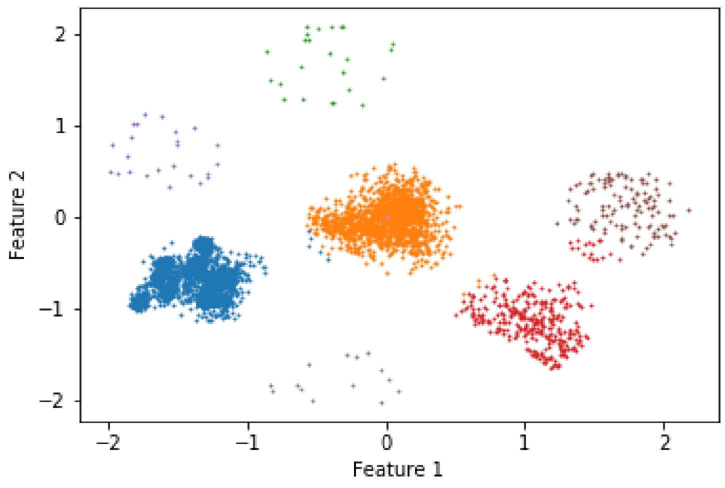


Figure 3-5: 2D mean-shift clustering results for car class
This is the clustering result obtained by mean-shift algorithm without leveraging instance-level ground truth information.

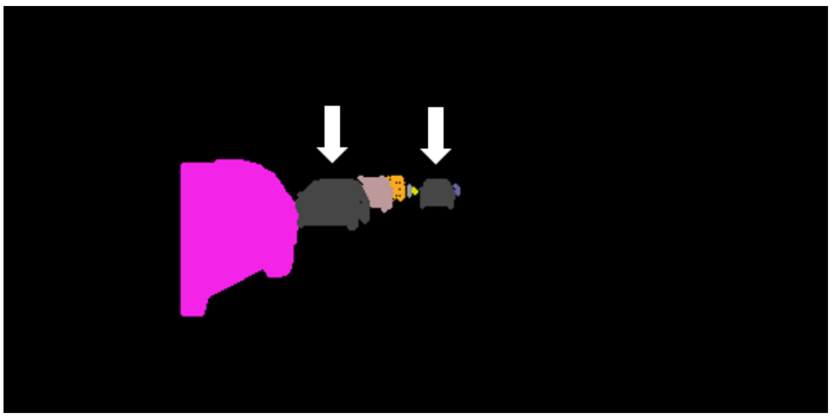


Figure 3-6: Instance mask for car class obtained by mean-shift clustering
Each instance is given a different color in this mask randomly. White arrows refer to the two instances clustered as one.

3-2 Incorporating Depth Information into RGB Embeddings

Inspired by the work of depth aware instance segmentation [2], this thesis proposes to attach depth as an additional dimension (layer) to the existing RGB n-D embeddings. Ideally, instances overlapped in the n-D embeddings (Figure 3-4) can be separated in the $(n+1)_{th}$ depth dimension. Depth image is a single channel image, so it can be scaled and added to n-D embeddings as the $(n+1)_{th}$ dimension directly. Only a scaling parameter is learned by the network during training. This method is **Scaling** which will be introduced in Section 3-2-1.

Another method is to transform depth into a 3-channel image. This adaptation renders it to be compatible with RGB network which takes in 3-channel images. Network can learn a m-D embeddings for depth in the same way how it learns n-D embeddings for RGB images. Two combination approaches, namely **Concatenation** and **Fusion**, for RGB and depth are presented in Section 3-2-2. Depth encoding procedure can be formulated in 3-8 where d_i refers to a single depth pixel which is encoded into ed_i by E with the same discriminative loss function used in RGB encoding. The complete process of scaling, concatenation and fusion can be easily expressed in 3-9, 3-10 and 3-11. The learned scaling parameter is donated as s.

$$ed_i = E(d_i) \qquad \text{(Encoding(depth))} \tag{3-8}$$
$$i_i = I(E(p_i) + s \times d_i) \qquad \text{(Scaling)} \tag{3-9}$$
$$i_i = I(E(p_i) + E(d_i)) \qquad \text{(Concatenation)} \tag{3-10}$$
$$i_i = I(E(p_i + d_i)) \qquad \text{(Fusion)} \tag{3-11}$$

Two transformation methods, **Replication** and **Coloring**, for adapting depth from single channel image to 3-channel image will be elaborated in Section 3-2-3.

3-2-1 Scaling depth image

The simplest way to incorporate depth to RGB network is scaling depth to adapt depth values into suitable range and attaching it to RGB embeddings. During inference procedure, a bandwidth is usually set to cluster regiments of embeddings. Therefore, depth values of corresponding RGB embeddings should accumulate around a mean within the same in-margin as well. For example, if instances in RGB image are encoded into 2D embeddings within an in-margin of 0.5, then corresponding depth values in the third dimension should also gather together within 0.5 to their own centers. In this way, clustering can be performed in the 3D space.

This thesis proposes to scale depth image by a single scaling parameter. All depth values are multiplied by this scaling parameter. No modification on the network is required. Only a tensor for scaling parameter is added to be learned during the training process.

3-2-2 Encode depth image into m-D embeddings

Scaling is quite simple which may not yield good performance. Different from scaling depth image, this section proposes a new method to process depth values. Similar to encoding

RGB images to n-D embeddings, depth can be encoded to m-D embeddings with the same discriminative loss function. The intuition is learning a 1D embeddings for depth image because it only contains information in longitudinal direction. So with the discriminative loss function, network learns a 1D embeddings feature map where embeddings belonging to the same instance in the depth image are accumulated together while centers of embeddings for different instances are repulsed away.

In the work of depth aware instance segmentation [2], depth is added at the end as an order index to the instance masks to avoid overlapping. Similarly, depth can be added to the n-D RGB embeddings as an order index in this work to avoid overlap in RGB embeddings as well. In this scenario, the 1D embeddings of depth is learned independently from n-D RGB embeddings and concatenated with it after training. This combination method is called **concatenation** in this thesis. However, it is also possible to train two networks together where the n-D RGB embeddings and 1D depth embeddings are fused before going to the loss function. This method is named as **fusion** in this work.

As mentioned above, two methods for incorporating depth information into RGB embeddings are involved in this work, namely concatenation and fusion. The only difference between the two methods is the stage of combination. Concatenation combines the outputs after training while fusion combines feature maps before loss function. So RGB and depth network are trained jointly in fusion approach. This part will explain the theories in detail.

- **Concatenation**

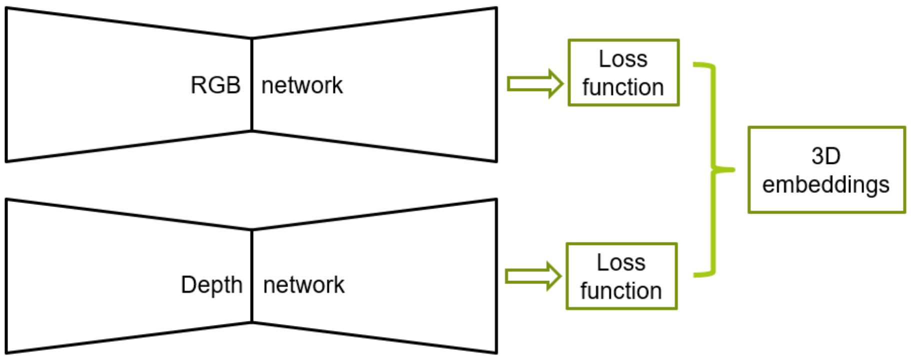


Figure 3-7: Concatenation
The top stream is RGB network where a 2D embeddings feature space is learned with the discriminative loss function. The stream in the bottom is depth network that learns a 1D embeddings for depth information. Embeddings of RGB and depth are concatenated after training two streams independently.

In concatenation method, two networks are trained independently with the same discrimininative loss function introduced in Section 2-3-4. One network learns a n-D embeddings space map for RGB images, the other one learns m-D embeddings for depth images. Figure 3-7 is an example for concatenation where RGB images is encoded into 2D feature space while depth images is encoded into 1D feature space. Feature maps from RGB network and depth network are concatenated to make a 3D feature space after training.

Due to the fact that RGB images and depth images are encoded with in-margin (0.5) and out-margin (1.5) independently by two networks. So the corresponding 3D in-margin and out-margin are adapted by equation:

$$in - margin = \sqrt{0.5^2 + 0.5^2} = 0.7071 \tag{3-12}$$

$$out - margin = \sqrt{1.5^2 + 1.5^2} = 2.1213 \tag{3-13}$$

- **Fusion**

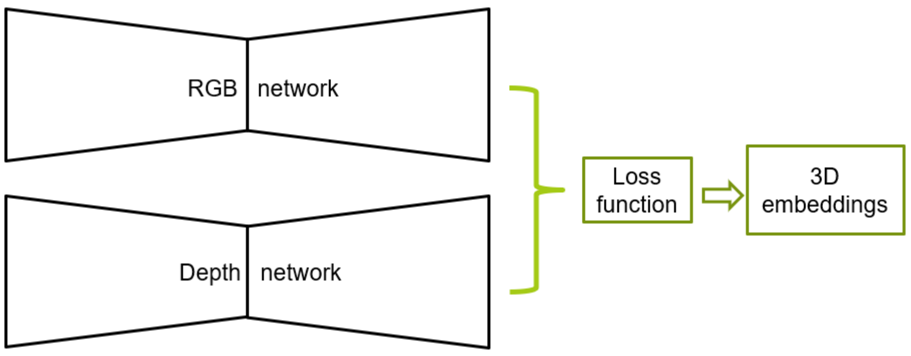


Figure 3-8: Fusion
The top stream is RGB network and the bottom one is depth network. 2D embeddings and 1D embeddings for RGB and depth are learned separately in the two steams. Outputs are fused before going to loss function. The final output of the fused network is a 3D embeddings map.

In fusion, networks for RGB and depth are trained jointly as shown in Figure 3-8. RGB network learns to encode RGB images into n-D embeddings while depth network learns to encode depth images into m-D embeddings in a similar way of concatenation. However, the outputs of two networks are fused before going to the loss function which means the two networks form a multi-modal network.

In this method, the multi-modal network learns a 3D embeddings directly. Depth feature map and RGB feature map are obtained inseparably indicating that depth here is not an order index but is fused to RGB channels as additional source information. Loss function learns a 3D feature map where 3D embeddings of the same instance are gathered while 3D embeddings of different instances are separated. It does not indicate that two dimensions of the three corresponds to RGB information while the other one represents depth information. This is also the reason why this method is named as 'fusion' while the separate training method is called 'concatenation'.

3-2-3 Depth transformation

As discussed above, depth image is a single channel image which should be adapted to 3-channel image to be fed into a network. Two popular methods for transforming depth image to 3-channel images are introduced here, namely **replication** and **coloring**. So, if the input of depth image is of size $3 \times 256 \times 512$, the learned 1D feature map is $1 \times 256 \times 512$.

To verify whether depth is a source as good as RGB images for instance segmentation, depth image is encoded into m-D embeddings. If the sole performance of depth image is better than the combination of RGB image and depth image, it means the improvement brought by incorporating depth information into RGB embeddings is not due to the combination but because of the superiority of depth channel itself. On the other hand, if the sole performance of depth image is better than the sole performance of RGB image, it demonstrates that depth information is potentially a more suitable source for instance segmentation than RGB images. Experiments are carried out and discussed in Chapter 4.

Two transformation methods for adapting depth images to 3-channel images are presented in this part. Replication is the most simple one where the single channel is repeated twice to obtain three channel. Another method is coloring [51] where a 'jet colormap' is adopted to transform the single channel gray-scale depth image into 3-channel RGB image.

Replication

The single channel of depth image is repeated to obtain a 3-channel input image in replication method. Figure 3-9 is the ground-truth 3D embeddings obtained by concatenation with replicated depth input. Due to the fact that RGB embeddings and depth embeddings are learned separately in concatenation, the 2D embeddings involved in this 3D feature space is the same as what was referred in Section 3-4. Model weights are saved in the previous experiments and reused in this 3D concatenation experiment.

The overlapped 2D embeddings in color pink and gray (Figure 3-4) correspond to the 3D embeddings (Figure 3-9) in the same colors. It is obvious that the pink embeddings and gray embeddings are separated in the z-axis although they occupy the same space in x-y plane. Clustered results by mean-shift are shown in Figure 3-10. Correspondingly, instances are separated in the mask. Compared to the mask shown in Figure 3-6 obtained by clustering RGB 2D embeddings, the two misclassified instances pointed by white arrows are correctly separated in Figure 3-11.

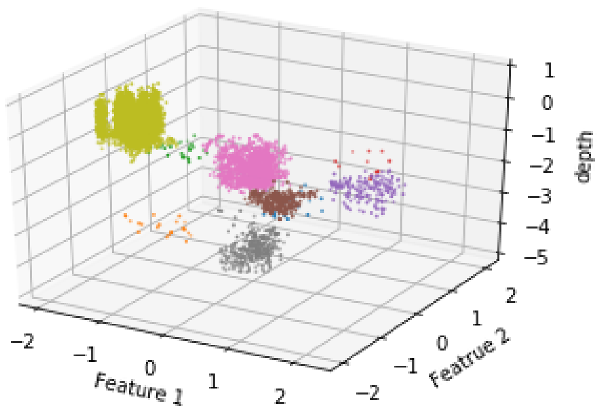


Figure 3-9: Ground-truth 2D RGB embeddings + 1D depth embeddings of car class
Feature 1 and Feature 2 are the 2D feature map for RGB information. Z-axis represents 1D depth embeddings.
Colors are given randomly to different instance embeddings.

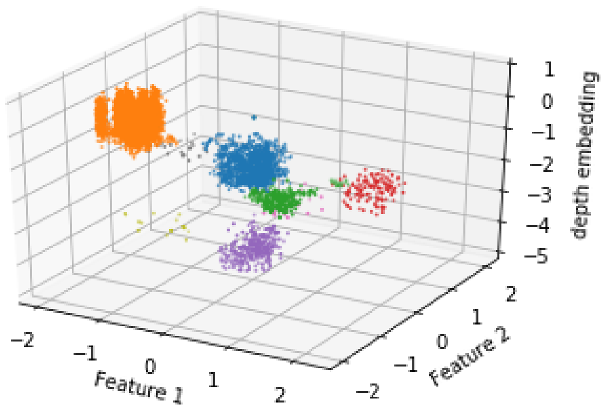


Figure 3-10: Clustering results on 2D RGB embeddings + 1D depth embeddings of car class by mean-shift
Feature 1 and Feature 2 are the 2D feature map for RGB information. Z-axis represents 1D depth embeddings. Colors are given randomly to different instance embeddings.

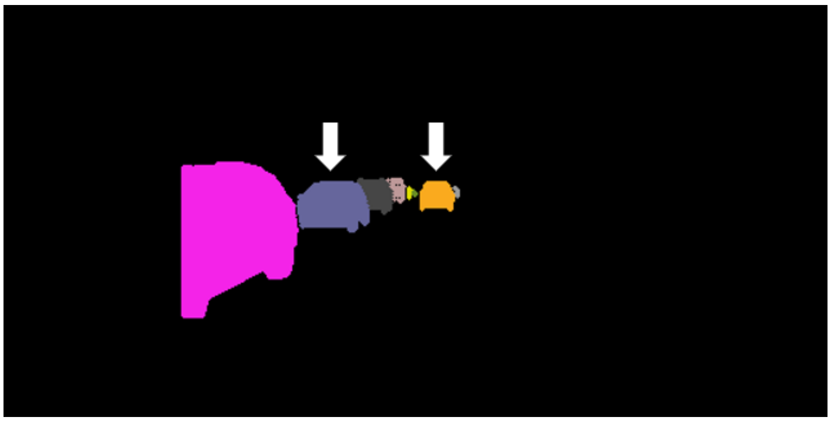


Figure 3-11: Instance mask of car class obtained by concatenating depth to 2D embeddings. Each instance is given a different color randomly. White arrows point to the misclassified two instances in 2D embeddings. They are correctly classified as two instance by concatenation of depth and RGB images.

Coloring

Rather than replication, the work in [51] proposes to color depth image according to its gray-scale value by a 'jet map'. The original single channel gray-scale depth image of frame 3-3 is Figure 3-12. After coloring with 'jet map' in *matplotlib*, depth is transformed from gray-scale image to RGB image as shown in Figure 3-13. The depth image is obtained from disparity image so the depth value is not fine grained. However, the boundaries of cars are still recognizable in the colored depth image.

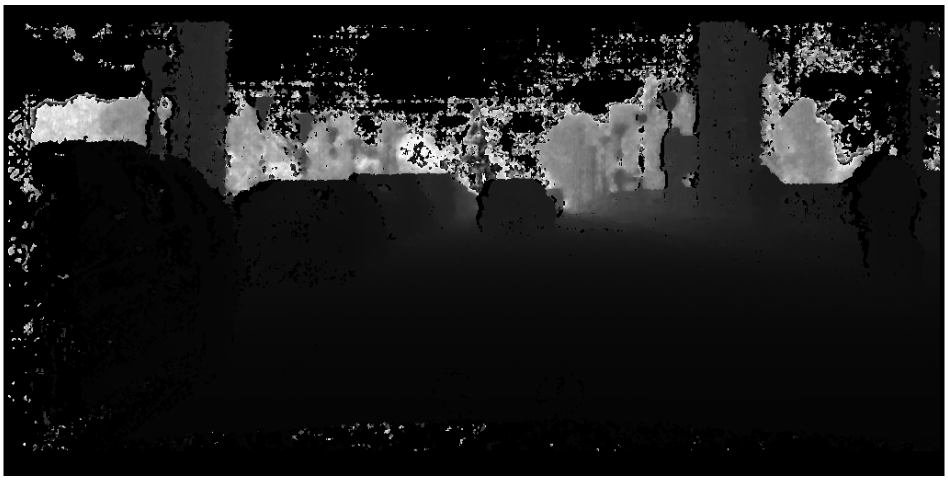


Figure 3-12: Gray-scale depth image of frame 3-3

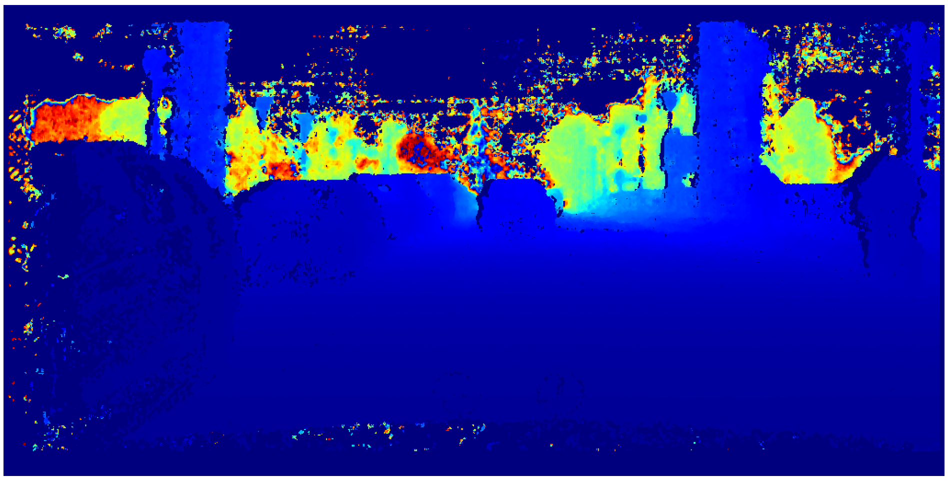


Figure 3-13: Colored depth image of frame 3-3

3-3 Evaluation Metrics

3-3-1 Accuracy

Some terminologies need to be explained before presenting accuracy evaluation metrics.

- **True Positive (TP)**: True items that are classified as false
- **True Negative (TN)**: False items that are classified as true
- **False Positive (FP)**: False items that are classified as false
- **False Negative (FN)**: True items that are classified as false.

Precision and recall

Precision and recall are commonly used evaluation metrics.

$$P = \frac{TP}{TP + FP} \tag{3-14}$$

$$R = \frac{TP}{TP + FN} \tag{3-15}$$

IoU and mAP

The commonly used evaluation metrics for semantic segmentation accuracy are mean Average Precision (mAP) and Intersection over Union (IoU).

IoU is the ratio between the overlapping area and the union area. Figure 3-14 shows the definition of IoU for detection. In this work, IoU is not used to evaluate the accuracy of bounding boxes but the segmentation. So the shapes are different but the definition is the same. If IoU is larger than a certain threshold, then this prediction is considered as positive, vice versa. Precision rate and recall rate are calculated based on the number of positive predictions.

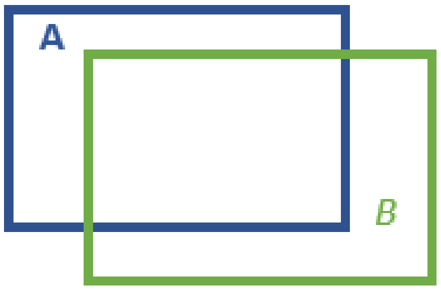


Figure 3-14: Definition of IoU
The blue bounding box is the ground truth. The green one is the predicted area.

$$IoU = \frac{A \cap B}{A \cup B} \tag{3-16}$$

Average Precision (AP) and mean Average Precision (mAP) are used to asses the performance of instance segmentation. AP is the summation of the area of precision-recall curve at a set of eleven equally spaced recall levels [54] [55]. According to Cityscapes benchmark, AP is calculated as an average of AP(s) at 10 levels IoU threshold from 0.5 to 0.95 with a step of 0.05. mAP is the mean AP over all classes.

$$AP = \frac{1}{10}\sum_{s=1}^{10}(\frac{1}{11}\sum_{n=1}^{11}(R_n - R_{n-1})P_{sn}) \tag{3-17}$$

R_n, P_n refer to recall and precision at the n_{th} recall level. The 10 steps are denoted as s.

3-3-2 Efficiency

Efficiency, or speed is usually measured by frame per second (fps)(or frequency (Hz)), time (ms). The more fps the model can handle, the shorter time is used, or the higher frequency the model has, the higher processing efficiency is.

Chapter 4

Experiments

4-1 Dataset

Large-scale dataset is indispensable for network training. This thesis only involves experiments on Cityscapes [17] dataset. Cityscapes is a stereo dataset designed for urban scene understanding. It provides a comprehensive collection and annotated samples for 3D urban street scene. Both semantic annotation and instance-level annotation are available. However, semantic annotation involves 30 classes, shown in Figure 4-2, including sky, road, etc. while instance-level annotation is only available in 8 classes, namely person, rider, car, truck, bus, train, motorcycle and bicycle. This dataset was collected in 50 cities in Germany spanning from spring through summer to fall. Weather was generally good without extreme circumstances. Frames in this dataset are manually selected from recorded videos to cover the variety of objects and street scenes.

Cityscapes is comprised of many sub datasets. These dataset are listed in Appendix A-1.

This work only involves three original datasets from Cityscapes, ***leftImg8bit_trainvaltest (4-1-1), gtFine_trainvaltest (4-1-2) and disparity_trainvaltest (4-1-3)***. Another depth dataset obtained from ***disparity_trainvaltest*** in Cityscapes will be introduced in Section 4-1-4.

4-1-1 RGB sub-dataset

RGB sub-dataset refers to the original sub-dataset ***leftImg8bit_trainvaltest*** in Cityscapes. Cityscapes is a stereo dataset where two cameras are adopted during data collection. The detailed working principal of obtaining stereo information from two cameras will be elaborated in Section 4-1-4. *LeftImg8bit_trainvaltest* contains frames recorded by the left camera. Frames recorded by the right camera are sorted in *rightImg8bit_trainvaltest* dataset. There are subtle difference (disparity) between images in the two datasets but they contain the same semantic meanings so usually only *LeftImg8bit_trainvaltest* is adopted for training. This disparity information is useful for constructing the 3D urban scene. RGB sub-dataset comprises

5,000 manually selected frames among which 3475 images are annotated for the purpose of training and validation. The annotations for the rest 1525 images are not available to the customers. This portion is the testing dataset which is used to evaluate the performance of submitted algorithms. Image 4-1 is a typical example from training RGB sub-dataset.

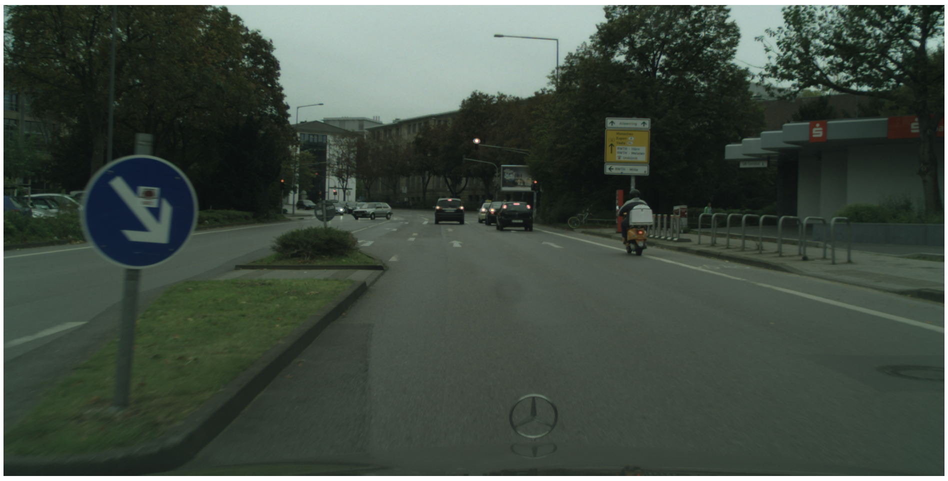


Figure 4-1: one frame captured in Aachen

4-1-2 Ground-truth sub-dataset

Ground-truth sub-dataset refers to the original sub-dataset ***gtFine_trainvaltest*** in Cityscapes. Fine annotations for the 3475 training and validation images in *leftImg8bit_trainvaltest* sub-dataset are available in ground-truth sub-dataset. Besides *leftImg8bit_trainvaltest* dataset, another dataset, *leftImg8bit_trainextra* with only coarse annotations is available. Corresponding coarse annotations are stored in *gtCoarse* dataset. The proportion of annotations for each classes are presented in Figure 4-2. Due to different occurrence frequencies of cars, bus, motorcycle etc., annotation proportion varies from 10^9 to 10^7 pixels. Moreover, as shown in Image 4-1, one frame usually contain many cars while motorcycle appears individually which will lead to a bias prediction in some algorithms. This effect will be discussed with many experiments as demonstration in section 4-3-1, Chapter 4.

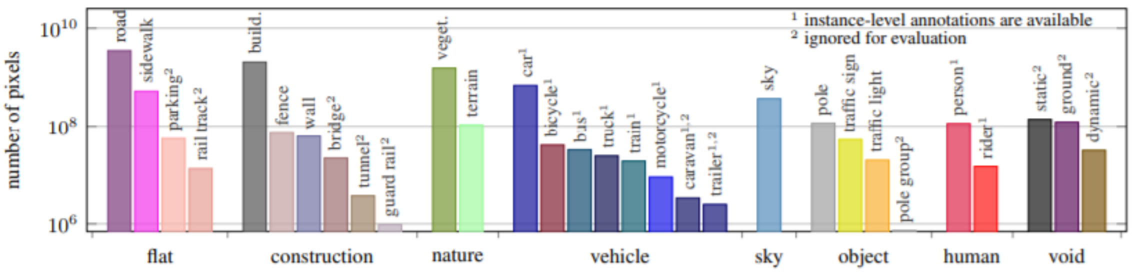


Figure 4-2: annotations in Cityscapes.
x-axis is the list of classes. y-axis is the number of annotated pixels. Classes with footnote '1' are the classes where instance-level annotations are available. [17]

4-1-3 Disparity sub-dataset

Disparity dataset is a collection of depth images in accordance with the 5,000 images in *leftImg8bit_trainvaltest*. These images are 16-bit pngs. Transformation as shown in equation 4-1 is required to obtain real disparity value from these images. Pixel value at a certain point is denoted as p.

$$d = (float(p) - 1.)/256 \tag{4-1}$$

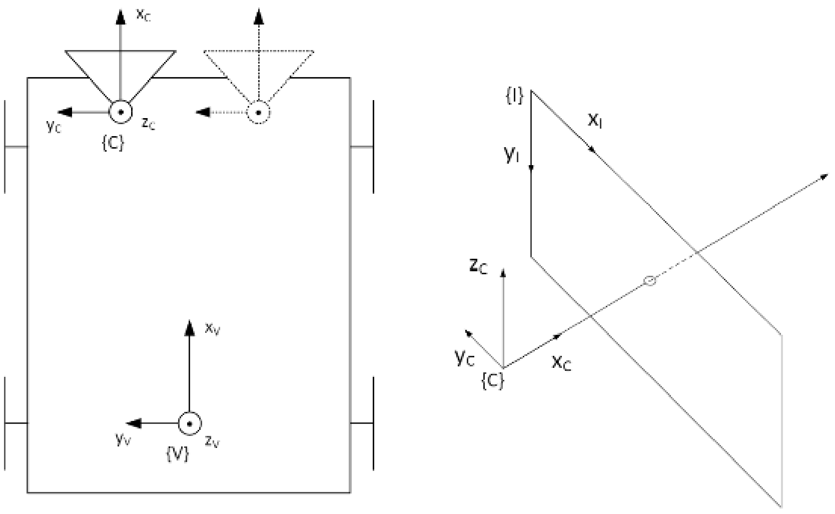


Figure 4-3: Illustration for three coordinate systems
Left figure shows camera coordinate system and vehicle coordinate system. The right one is an illustration for image coordinate system. [17]

In 3D vision field, three coordinate systems are defined as shown in Figure 4-3.

- **World Coordinate System**

  In Cityscapes, World Coordinate System is defined as the Vehicle Coordinate System. The origin of Vehicle Coordinate System is on the ground below the rear axis according to [56]. Forward driving direction is the x axis and left is the y-axis.

- **Camera Coordinate System**

  Camera Coordinate System is similar to Vehicle Coordinate System except that the original point is located at the camera's optical center as shown in top left corner in the left subfigure in Figure 4-3.

- **Image Coordinate System**

  Image Coordinate System has the pixel in the top left corner as the origin. X-axis points to right and y-axis points downward.

What obtained from the disparity images are disparities in Image Coordinate System. Intrinsic or extrinsic camera metrics are required to transfer the disparities in Image Coordinate

System to depth in Camera Coordinate System or Vehicle Coordinate System. Due to the fact that depth is only used as an additional dimension to distinguish the spatial location of different instances in this work, it is not necessary to transfer image coordinates to world coordinates. Camera coordinates can satisfy the demands. The only difference is the origin as mentioned before. Concrete method to obtain depth in Camera Coordinate System will be introduced in next section.

4-1-4 Depth sub-dataset

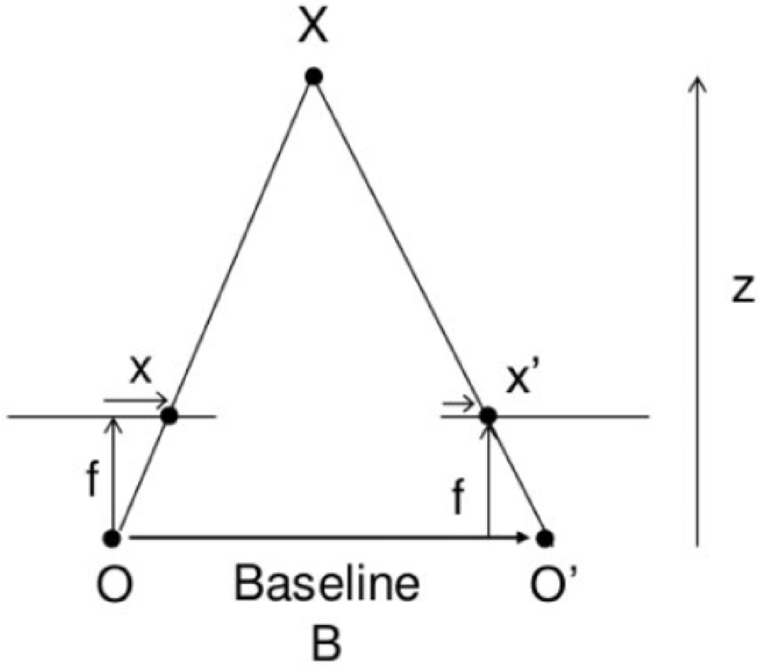


Figure 4-4: Disparity diagram [18], [19]

$$disparity = x' - x = \frac{Bf}{Z} \tag{4-2}$$

Diagram 4-4 presents the mathematical model for how to obtain depth from disparity $x' - x$. O and O' refer to the optical centers of left and right cameras. B and f are the baseline between two cameras and their focal length. Z, the depth, can be calculated out based on equivalent triangle theory. This calculation should be done pixel by pixel. Obtained depth maps are single channel 8-bit grayscale images ranging from 0 to 255.

$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-3}$$

K in equation 4-3 is the intrinsic matrix for cameras. u_0 and v_0 are the coordinates for origins. f_x and f_y refer to focal lengths. These parameters are provided by Cityscapes. With this intrinsic matrix, not only depth but also lateral distance and height information can be derived with equations 4-4, 4-5 and 4-6. $xCam, yCam$ and $zCam$ are depth, lateral distance and height in Camera Coordinate System.

$$xCam = \frac{f_x * B}{disparity} \tag{4-4}$$

$$yCam = -\frac{xCam}{f_x} * (u - u_0) \tag{4-5}$$

$$zCam = \frac{xCam}{f_y} * (v_0 - v) \tag{4-6}$$

4-2 Experiment Setup

All experiments in this thesis are performed with ENet to meet the real-time requirement. Learning rate is chosen to be 5e-4 and training batchsize is 10. Optimizer used in all experiments is Adam. Input RGB images, depth images and labels are first rescaled and cropped from center to 256×512 resolution. Depth images and labels are transformed to float tensors and long tensors separately.

4-3 Results

Some experiments are presented in this section to answer sub-level research questions proposed in Section 1-4. For the purpose of visualization, only 2D embeddings was discussed above. In the original work, RGB images are encoded into 8D embeddings which will be used as baseline in later experiments, such as fusion and concatenation.

Section 4-3-1 will elaborate the impact of bandwidth on the results. This part will be well explained to make sure readers can interpret the results of different methods discussed after this section. Section 4-3-2 explores the affect of embeddings' dimension on accuracy (AP). Whether there is huge difference or not between the two depth transformation methods, replication and coloring, will be answered in Section 4-3-3.

Performance of adding scaled depth image directly to RGB feature map will be discussed in Section 4-3-4. Comparison between fusion and concatenation will be presented in Section 4-3-5 and 4-3-6. All methods will be compared and discussed in Section 4-3-7. Runtime performance will be compared in Section 4-3-8. At the end, some examples in validation dataset are visualized for 8D embeddings and 9D fusion.

4-3-1 Impact of bandwidth and criterion for performance analysis

Impact of Bandwidth

In Cityscapes, there are only 8 classes where instance-level annotations are available, Figure 4-2. Classes car, person and bicycle usually contain lots of instances in one frame while there

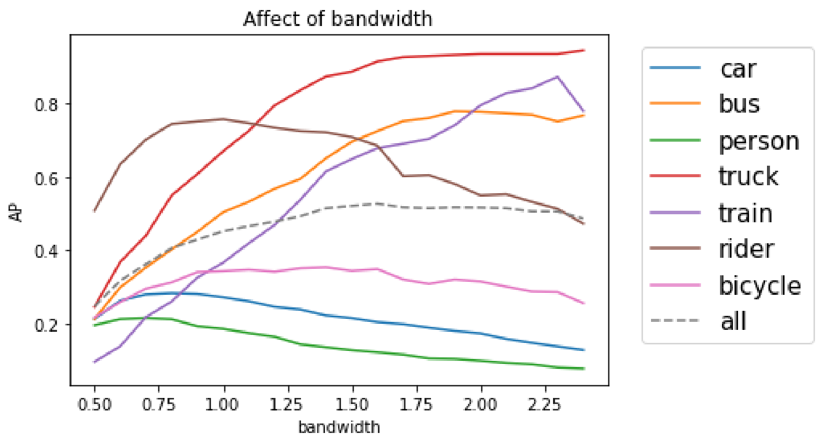


Figure 4-5: Affect of bandwidth in different classes

are only one or two instances appearing in a single frame of classes like truck, bus and train as discussed in Section 4-1. This will lead to biased result which is closely related to bandwidth, the threshold of mean-shift clustering. In this circumstance, the larger the bandwidth, the higher the accuracy in classes with few instance in a frame. For example, in truck class, non-background pixels are selected by semantic mask. If there is only one instance in a certain class and the bandwidth is very large, all the selected pixels are classified as one instance. So, the larger the bandwidth, the better the result in this class is.

Figure 4-5 compares accuracy change in all classes when bandwidth is increased from 0.5 to 2.5. This result is obtained from 2D RGB embeddings. Car and person classes usually contain many instances in one frame, corresponding to convex curves in this graph where the maximum is achieved at bandwidth 0.75. Clustering does not perform well if the bandwidth is too small or too large. If the bandwidth is too small, embeddings of the same instance may not be all clustered to the corresponding center. Reversely, embeddings of a certain instance may be clustered to centers of other instances if the bandwidth is too large. So the ideal bandwidth should be a value between in-margin and out-margin of the embeddings, which turned out to be 0.75 in car class. To the contrast, truck and bus classes first ascend with the increasing of bandwidth then reach plateaus after bandwidth 1.5. This verifies the theory proposed above. If there is only one instance in a certain class, the larger the bandwidth, the higher the accuracy is. After all embeddings are clustered to one instance, accuracy does not increase anymore by enlarging bandwidth. The dash line is the overall performance which is an average of 8 classes. This result demonstrates that there is a trade off between classes with lots of instances and classes with few similarities.

As mentioned in the original paper [1], this discriminative loss function works well on images where lots of similarities exist, e.g. cars in Cityscapes. This is because the discriminative loss function forces the embeddings of different instances to push away from each other. If there is only one instance in the training image like the motorcycle class, the push force does not exist. So the high reported AP in classes truck, bus, train and motorcycle are biased due to large bandwidth. These biased results will affect the final average accuracy as shown in Figure 4-5.

Criterion for performance analysis

To evaluate different methods, it is reasonable to compare average accuracy and car class accuracy at the same bandwidth. Average accuracy indicates the overall performance of a certain method while car class AP shows how good is this method in classes with many instances. However, the rules for concatenation is slightly different. In concatenation, embeddings from two independent networks are combined indicating that the bandwidth should be adapted to a broader range to cover a higher dimensional space. This rule has been explained in Section 3-2-2.

4-3-2 The affect of dimensions of RGB embeddings

2D feature map is convenient for visualization but may not be the best choice with regard to accuracy. In the original work of discriminative loss function [1], instances are encoded into a 8D feature space. It is usually easier to separate or classify input data in a higher dimensional feature space. For example, Support Vector Machine (SVM) [57] can separate nonlinear data linearly by transforming them to a hyper-plane with kernels. However, higher dimensional representation requires more training data for the network to learn faithfully.

In this experiment, 2D, 8D and 9D embeddings are tested. Results are compared in Figure 4-6. According to the criterion in Section 4-3-1, average AP and car class AP are compared. Dash-lines are average AP in all classes. 9D RGB embeddings yield highest average AP while 8D RGB embeddings present best performance in car class.

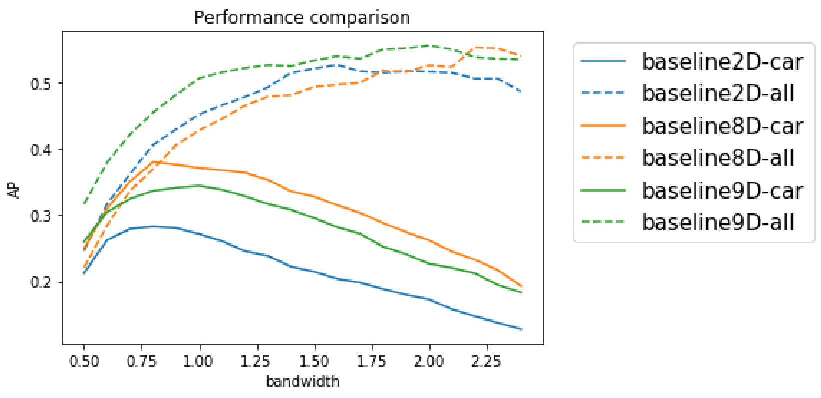


Figure 4-6: Affect of dimensions on 'car' class

8D embeddings is selected in the original work which is the baseline of this work. However, depth is incorporated into RGB embeddings as an additional dimension in later experiments which means the overall embeddings is 9D. So all methods also need to be compared with 9D RGB embeddings to eliminate the ambiguity whether the improvement is caused by adding depth or increasing dimensions. Comparison of all methods will be presented in Section 4-3-7.

4-3-3 Replication and coloring

As introduced in Section 3-2-2, replication and coloring are the two depth preprocessing approaches to prepare 3-channel depth images. Concatenation trains RGB and depth networks separately so the performance of depth transformation is not affected by RGB network. Under this circumstance, concatenation is an ideal method to compare replication and coloring. Figure 4-7 shows the results obtained by the two depth preprocessing approaches with 8D RGB and 1D depth concatenation. Coloring does not yield result as good as replication so later experiments are performed with replication only.

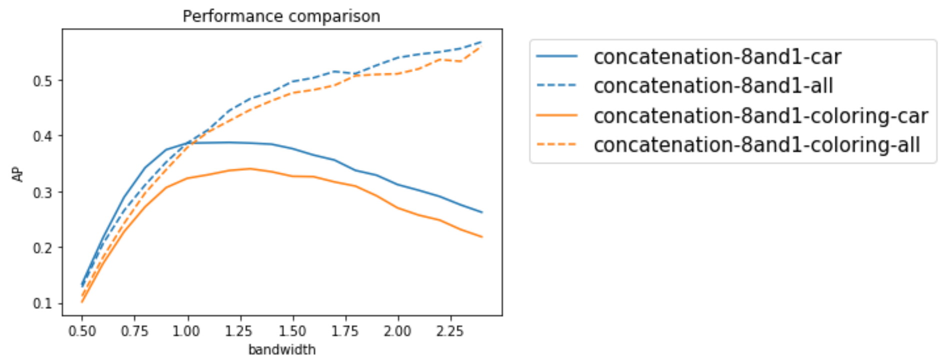


Figure 4-7: Comparison between replication and coloring

4-3-4 Scaling depth image

Experiments are carried out in this section to examine whether adding scaled depth information directly to 8D embeddings benefits the performance or not. Figure 4-8 compares the 8D baseline work and scaling method in car class and overall performance. Scaling yield a significant improvement in both car class and overall AP than 8D RGB embeddings when bandwidth is 1.5. However, the best performance of the two methods in car class are similar.

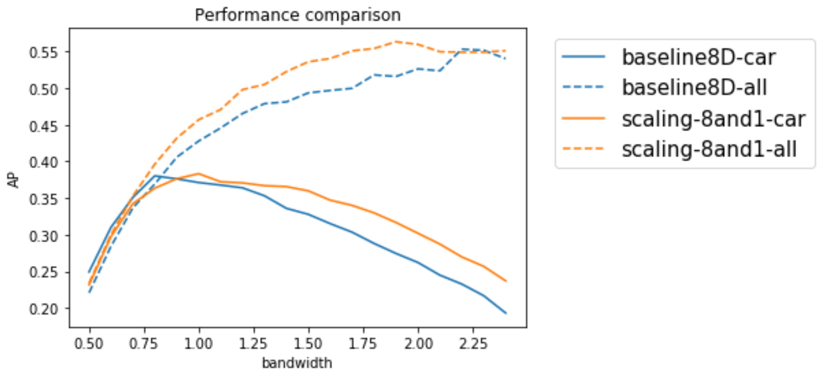


Figure 4-8: Performance of scaling in 'car' class

4-3-5 Concatenation

This section will demonstrate whether concatenation is superior than 8D embeddings or not. Depth is encoded into a 1D feature space and concatenated with 8D embeddings to form a 9D feature space. Results of concatenation is shown in Figure 4-9 and 4-10.

The bandwidth for 8D and 1D concatenation should be a value in the range from 0.7071 to 2.1213 according to the adaptation rule explained in Section 3-2-2. This explains why 8D and 1D concatenation achieves maximum value in car class at bandwidth 1 while baseline work achieves it at bandwidth 0.75 as shown in Figure 4-9.

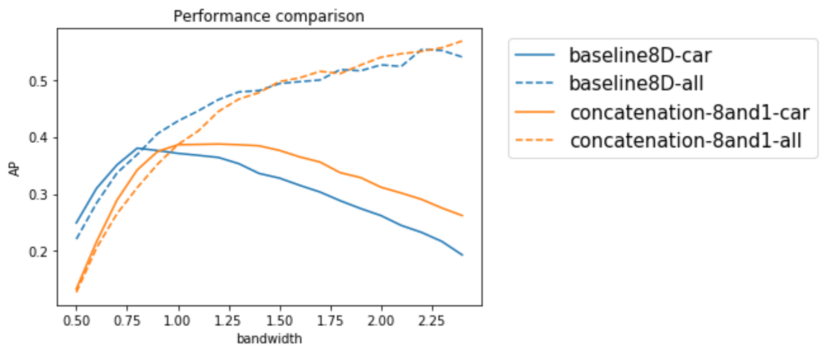


Figure 4-9: Performance of concatenation in 'car' class
Orange curves correspond to concatenation of 8D RGB embeddings and 1D depth embeddings. Blue curves represent baseline 8D RGB embeddings.

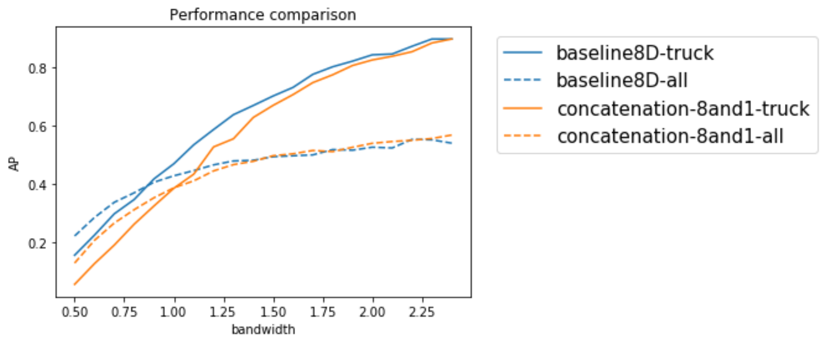


Figure 4-10: Performance of concatenation in 'truck' class
Orange curves correspond to concatenation of 8D RGB embeddings and 1D depth embeddings. Blue curves represent baseline 8D RGB embeddings.

Figure 4-9 and 4-10 compares the performance of concatenation and baseline work in car class and truck class separately. In concatenation, depth is encoded into 1D embeddings and concatenated with 8D RGB embeddings. Baseline work is 8D RGB embeddings. As discussed in Section 4-3-1, AP of classes with lots of similarities in a single frame present convex curves while AP of classes with only one or two instances in frames go up with the increase of bandwidth. Concatenation follows the same trend. According to the criterion, concatenation yields similar results to 8D baseline work. However, one interesting point is that concatenation performs better in classes with many instances as shown in Figure 4-9 where the orange curve is higher than the blue one when overall AP of the two methods are the same. To the contrast, concatenation yields lower accuracy in classes with few instances, see Figure 4-10.

4-3-6 Fusion

In 9D fusion, depth is encoded into 1D embeddings and is fused with the feature map of RGB network before loss function. The RGB network and the depth network constitute a multi-modal network. These two streams of the multi-modal network are trained jointly. So the clustering bandwidth does not need to be adapted to a broader range.

Figure 4-11 compares results obtained from 8D RGB embeddings and 9D fusion. Fusion achieves higher accuracy in both car class and overall classes than baseline work.

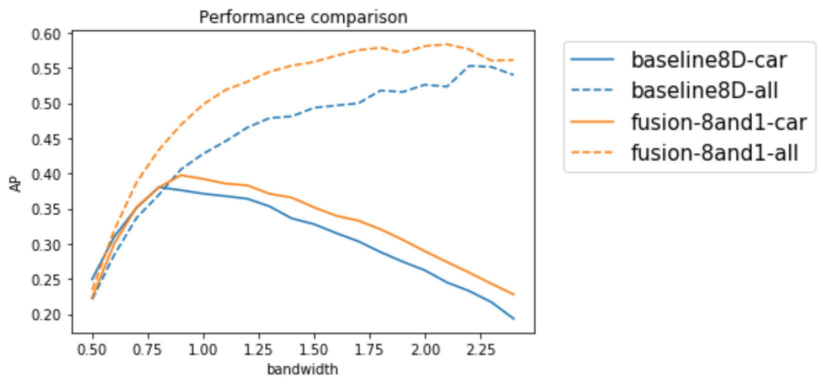


Figure 4-11: Performance of fusion

4-3-7 Comparison of all methods

Comparison with 8D baseline work

Figure 4-12 presents the results of all methods in car class and overall classes. Scaling, concatenation and fusion are not only compared to 8D RGB embeddings but also 9D RGB embeddings to eliminate the uncertainty whether the accuracy is improved by incorporating depth into RGB network or simply higher dimensions. 9D RGB embeddings is slightly better than 8D RGB embeddings but both of them are worse than scaling and fusion.

According to the experiment results, 8D RGB and 1D depth fusion obtains higher average AP than scaling. However, scaling only learns one parameter while fusion learns a 1D feature space to encode depth into embeddings indicating scaling is more efficient without sacrificing too much accuracy.

Concatenation is distinct from scaling and fusion in the way that it yields the highest accuracy in car class but lowest AP in average. This indicates that concatenation must performs worse in some other classes, for example, truck as shown in Figrue 4-13. This is already discussed in Section 4-3-5 that concatenation yields higher AP in classes with lots of similarities but lower AP in classes with few instances. In comparison with other methods, this phenomenon in concatenation is more obvious as it gives the lowest AP in truck class.

Precise results regarding to bandwidth are listed in Table 4-1. Best records of all methods are compared in Table 4-2.

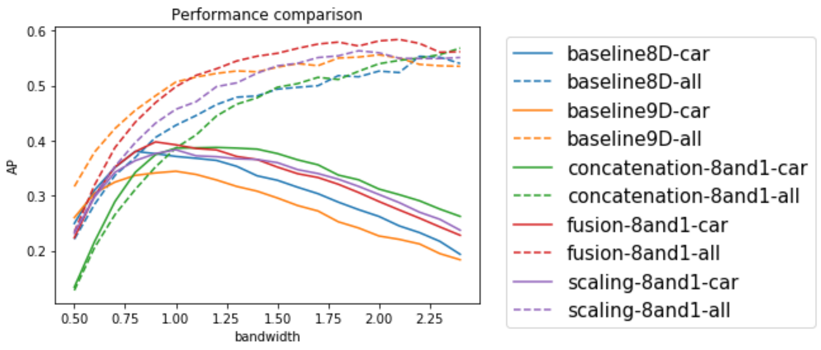


Figure 4-12: Performance comparison of all methods in 'car' class
Baseline 8D and baseline 9D refer to 8D and 9D RGB embeddings. Fusion_8and1 means 8D RGB feature map and 1D depth feature map are fused before going into loss function. Concatenation_8and1 combines the 8D RGB embeddings and 1D depth embeddings after training. Scaling attaches a single scaled depth channel to 8D RGB embeddings.

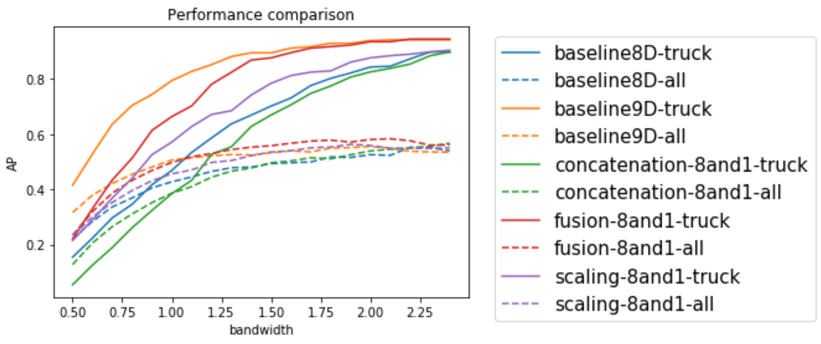


Figure 4-13: Performance comparison of all methods in 'truck' class
Baseline 8D and baseline 9D refer to 8D and 9D RGB embeddings. Fusion_8and1 means 8D RGB feature map and 1D depth feature map are fused before going into loss function. Concatenation_8and1 combines the 8D RGB embeddings and 1D depth embeddings after training. Scaling attaches a single scaled depth channel to 8D RGB embeddings.

AP	bandwidth = 1		bandwidth = 1.5		bandwidth = 1.8	
	car class	Average	car class	Average	car class	Average
8D RGB	0.372	0.422	0.327	0.495	0.285	0.516
9D RGB	0.339	0.501	0.297	0.532	0.251	0.543
Scaling	0.383	0.457	0.360	0.536	0.330	0.554
Fusion	0.393	0.500	0.351	0.558	0.322	0.578
Concatenation	0.385	0.393	0.375	0.495	0.342	0.517

Table 4-1: Performance of all Methods
The left column lists all methods. Scaling, fusion and concatenation combines 8D RGB embeddings and 1D depth information. Results are compared under different bandwidth in both car classe and average overall classes.

8D RGB	**9D RGB**	**Scaling**	**Concatenation**	**Fusion**
0.622	0.599	0.628	0.623	0.648

Table 4-2: Best records of all methods in all classes

To verify whether the improvement brought by incorporating depth into RGB network is contributed by the combination of two modalities of data or is just because depth is a superior source for instance segmentation than RGB images, depth is encoded into 1D embeddings and compared to the results obtained by 8D RGB embeddings and fusion as shown in Figure 4-14. The green curves represent the performance of unitary depth input in car class and overall classes, both of which are worse than fusion and 8D RGB embeddings. This experiment demonstrates that the improvement of fusion is due to the combination of RGB and depth instead of adding a better data source.

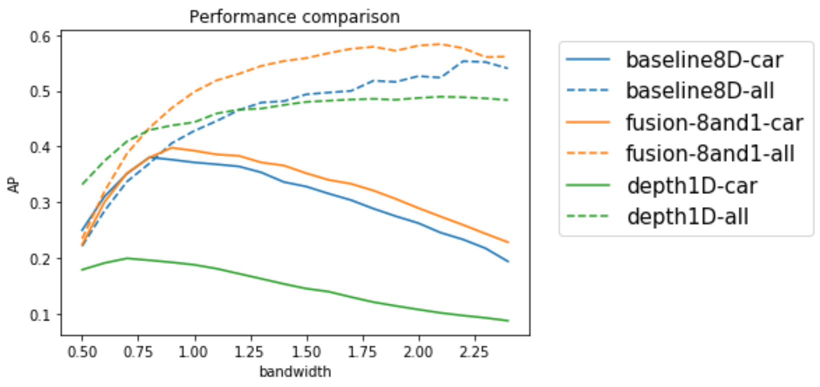


Figure 4-14: Performance comparison
8D RGB embeddings and 1D depth embeddings are fused in Fusion_8and1.

Comparison with 2D baseline work

Experiments are performed with 2D baseline as well where scaling, fusion and concatenation combines 1D depth information with 2D RGB embeddings to form a 3D space. In this scenario, concatenation is the best method among the three rather than fusion or scaling. The possible explanation is that instances are not well separated in 2D RGB embeddings so an independent depth embeddings in concatenation works well as an order index after training. In 8D RGB embeddings, instances are better separated with only RGB information. Then an independent depth layer in concatenation may not provide enough additional information to order these instances in a higher dimension. However, in fusion and scaling, depth is fused with 8D RGB feature during training. In this way, depth augments RGB source with more spatial information instead of working as an independent source.

2D baseline is significantly worse than 8D or 9D baseline so the main focus of this thesis is still on the higher dimensions. Experiment results in lower dimensions are presented in Appendix B.

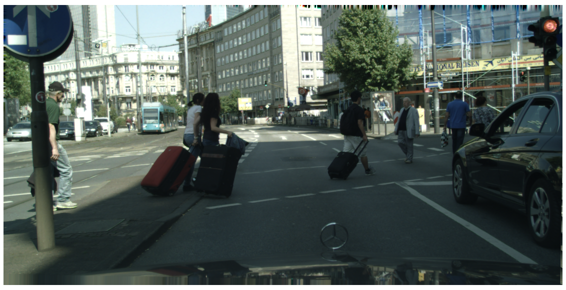
4-3-8 Runtime performance

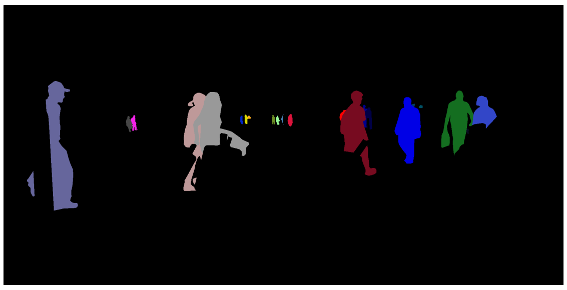
Experiments were performed with Python 2.7 on GPU Titan X. CUDA version was 8.0. For each single experiment, around 49% GPU memory was occupied.

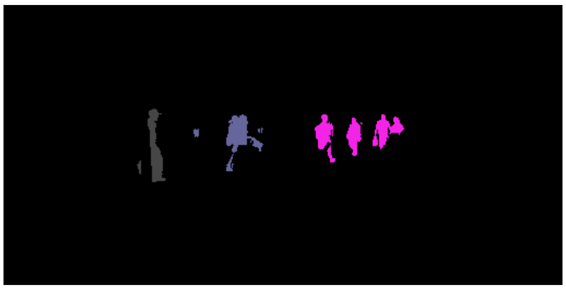
In 2D baseline work, run-time performance is 10.12 fps and 8.47fps for car class and all classes separately. The impact of adding depth to baseline work on run-time performance is negligible. In 8D baseline work, the speed is 5.55fps for all classes which is slower than 2D baseline work. Fusion has a similar performance to 8D baseline work. This runtime performance is affected by clustering which involves randomness each time so the performance is not fixed.

4-3-9 Visualization of some examples

Some visualization results for instance masks of car class and person class in Cityscapes are presented in this section. These instance masks are shown in pairs. Images (a) and (b) are the original RGB images and ground truth labels. Images (c) are inference results from 8D embeddings and images (d) are inferred from fusion of 8D RGB embeddings and 1D depth embeddings. Depth is replicated to form a 3-channel input image. Bandwidth is chosen to be 1.5 which is the original value adopted in work [1].


(a) RGB


(b) ground-truth


(c) 8D RGB embeddings

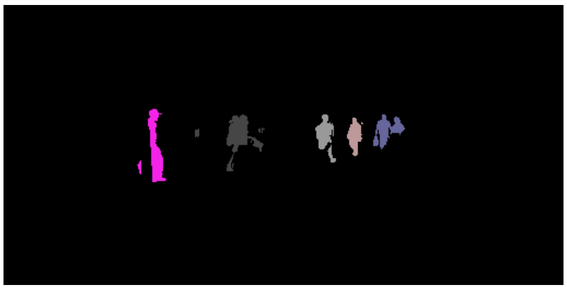
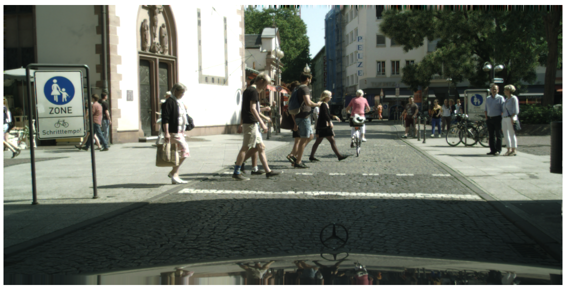
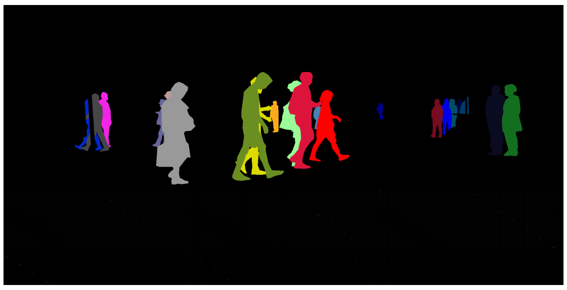
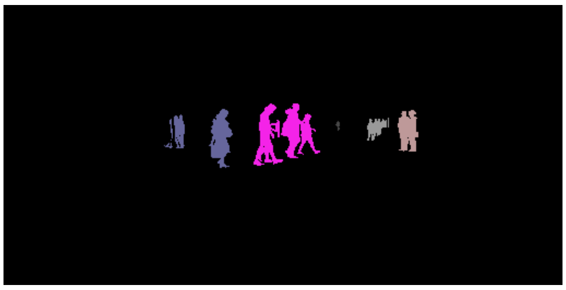

(d) Fusion

Figure 4-16: Results comparison


(a) RGB


(b) ground-truth


(c) 8D RGB embeddings

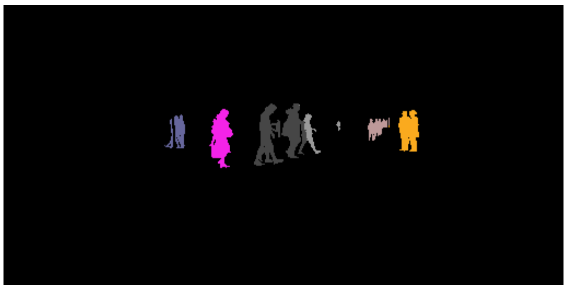
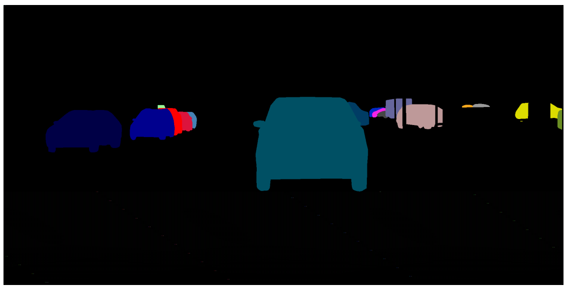
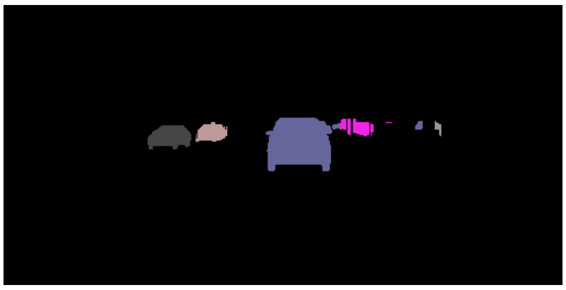

(d) Fusion

Figure 4-15: Results comparison

(a) RGB


(b) ground-truth


(c) 8D RGB embeddings

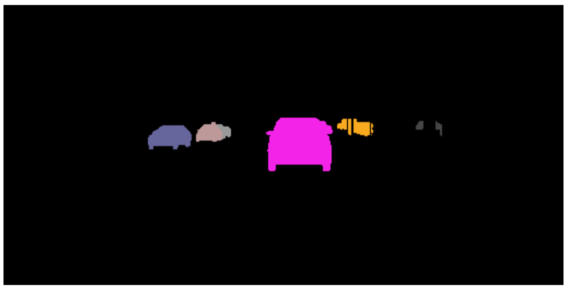
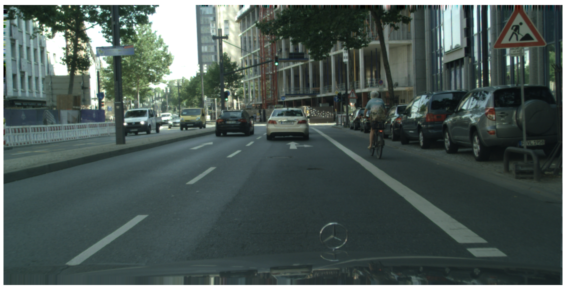
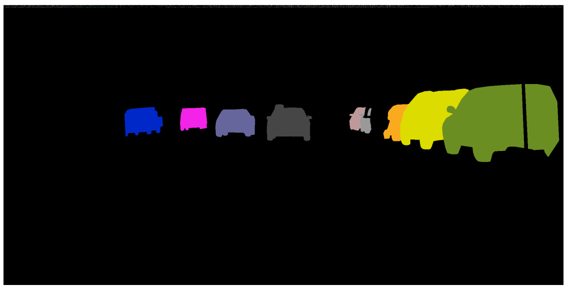
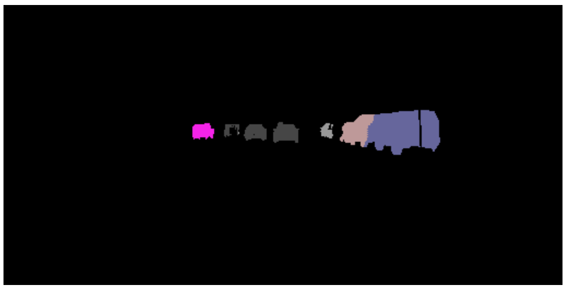

(d) Fusion

Figure 4-17: Results comparison


(a) RGB


(b) ground-truth


(c) 8D RGB embeddings

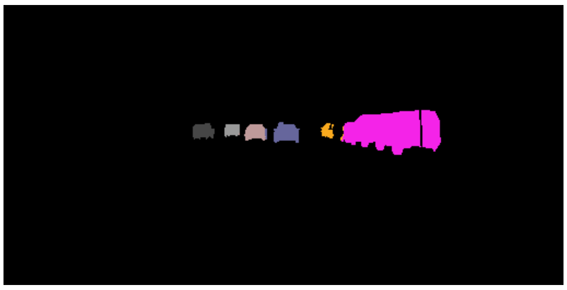
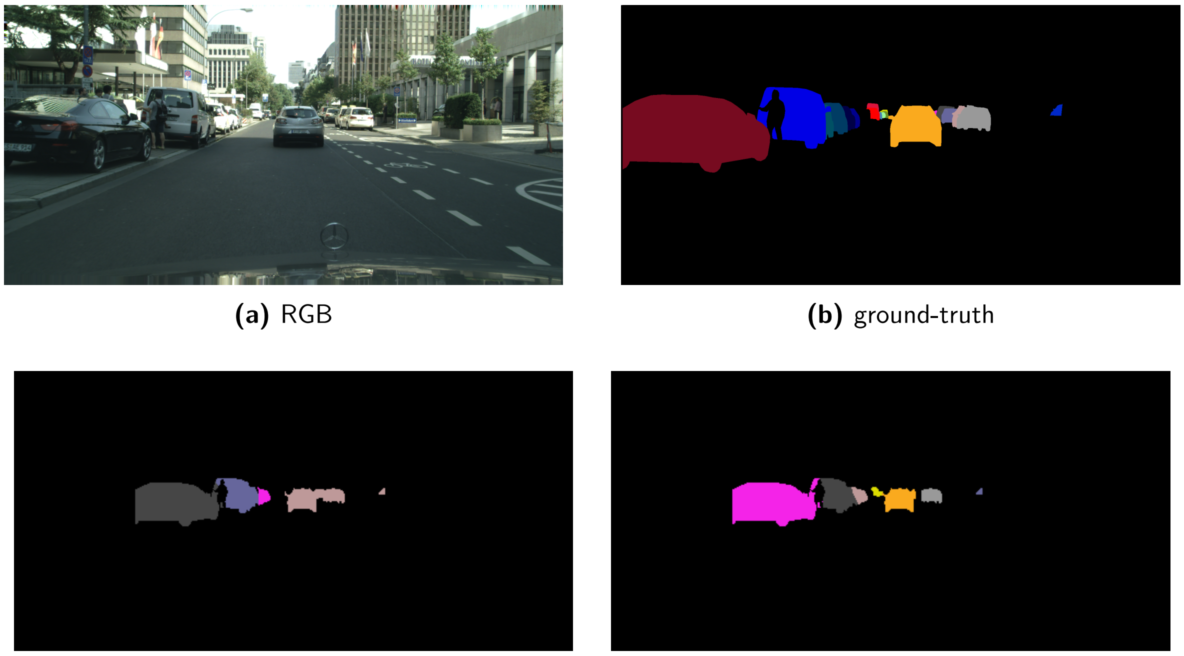
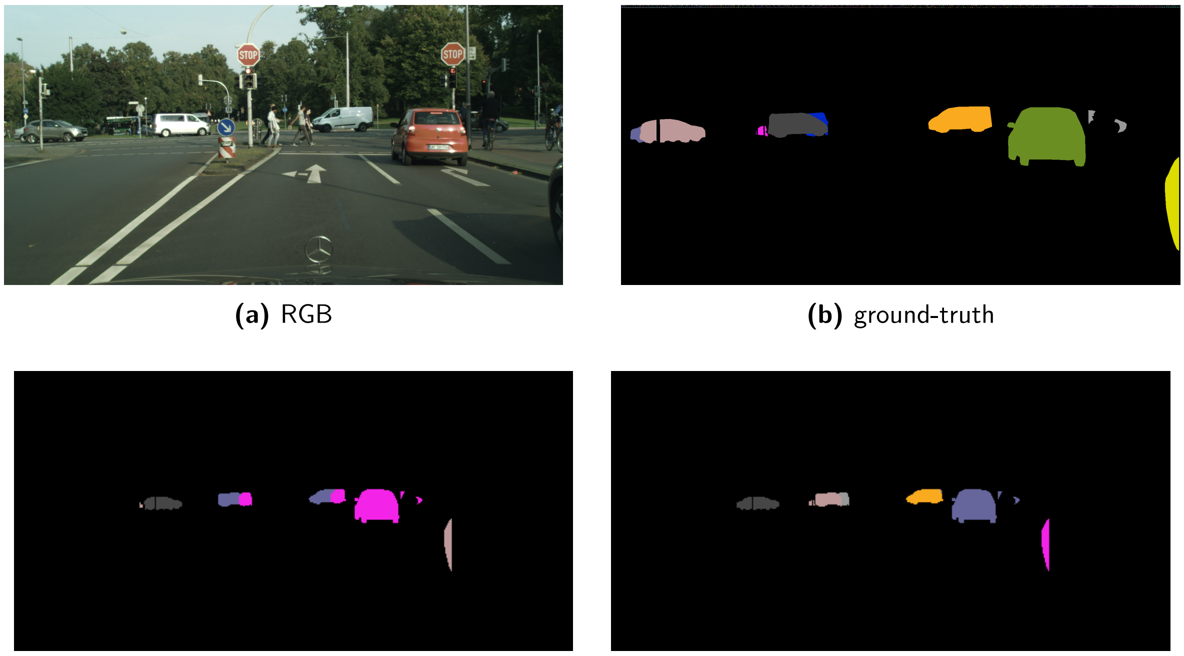

(d) Fusion

Figure 4-18: Results comparison


(a) RGB (b) ground-truth

(c) 8D RGB embeddings (d) Fusion

Figure 4-19: Results comparison



(a) RGB (b) ground-truth

(c) 8D RGB embeddings (d) Fusion

Figure 4-20: Results comparison

Chapter 5

Discussion and Conclusion

The general research question proposed in this work is whether incorporating depth information into RGB network can improve instance segmentation performance or not. The concrete theory of depth aware instance segmentation with a discriminative loss function is inspired by the work semantic segmentation with a discrimintive loss function [1] and depth-aware instance segmentation[2]. Semantic network with a discriminative loss function learns a n-D embeddings for RGB images where pixels of the same instance are encoded close to a center within an in-margin while pixels belonging to different instances repulse each other. Instance mask can be conveniently inferred from embeddings by clustering methods. Depth-aware instance segmentation is comprised of two streams, one of which predicts instance segmentation masks while the other one predicts depth information from RGB images. Predicted depth information is used as an order index to tackle occlusion problem in the predicted instance masks.

In this work, depth is encoded into 1D representation and combined with n-D RGB embeddings to form a (n+1)-D feature space which can be easily transformed into instance masks by the same inference procedure in [1]. Three methods, scaling, concatenation and fusion, are proposed to combine n-D RGB information and 1D depth information. Meanwhile, two depth transformation approaches are tested to adapt single channel gray-scale depth image to 3-channel input image.

5-1 What is the impact of parameters involved in instance segmentation in this work?

5-1-1 What is the impact of dimension of embeddings?

Experiments were performed with 2D, 8D and 9D RGB embeddings. 2D and 8D embeddings are the two dimensions involved in the baseline work [1]. Due to the curse of dimensionality, when the dataset is large enough to train the network completely, the higher the dimensions, the better the result is. According to the experiments introduced in section 4-3-2, 8D embeddings did yield significant higher accuracy than 2D embeddings and 9D embeddings performs better than 8D. So the theory is demonstrated.

5-1-2 What is the impact of clustering bandwidth? e.g. on single 'car' class and the overall AP?

Based on the experiment results shown in Figure 4-5, the larger the bandwidth, the higher the average AP is, the lower the 'car' class AP is. There is a trade-off between classes containing many instances, such as car class and classes with few instances, e.g. train, motorcycle. Increasing bandwidth will lead to biased high average AP by sacrificing car class AP. So it is reasonable to compare the performance of different methods in bandwidth range 0.75 to 1.8.

5-2 What is the best way to incorporate depth information into original RGB network?

5-2-1 Does colored depth image yield better result than simple replication of depth image? Which depth transformation approach is superior?

Based on the experiments presented in section 4-3-3, replication is slightly better than coloring so all experiments for scaling, concatenation and fusion are performed with replication. The reason could be that the depth images obtained from disparity images are of low quality so coloring does not provide more spatial information but may sacrifice some details compared to the original grayscale images.

5-2-2 Does data fusion work better or is there other superior method?

Three methods, scaling, concatenation and fusion are proposed to add depth into RGB network. Experiments show that all the three methods yield better results than 8D RGB embeddings baseline work.

According to the experiments in section 4-3-4, adding scaled depth information directly to RGB embeddings to form a (n+1)-D feature space does work better than concatenation but yields similar results to fusion. However, scaling is more efficient than fusion due to the fact that scaling only learns one additional parameter for depth whole fusion learns to encode depth into 1D embeddings with an extra network.

Concatenation is more suitable for classes with lots of instances, for example, car, person and bicycle. However, the average performance in all classes is pulled down by worse performance in classes with few instances, such as truck, train and bus. This trend is obvious in all methods but more significant in concatenation.

5-3 Future work

Depth information is a good modal of data for instance segmentation in this work. It is possible to gain advancement in the field of instance segmentation with stereo information.

One method could be encoding 3-D stereo information into 3 1D embeddings and concatenate the three feature maps with n-D RGB embeddings after training to form a (n+3)-D feature space. Each direction is encoded into one dimension. Inference is performed on this (n+3)-D feature space. In one word, exploiting depth and stereo information to benefit instance segmentation performance deserves more research devotion.

Fusion in this work refers to late-fusion which fuses the outputs of two networks. Early fusion and mid fusion deserve further research as well.

There is a trade off between efficiency and accuracy. The AP grows with higher encoding dimension while speed drops obviously. The increase of accuracy sacrifices speed. So the performance of this work is still not efficient enough to be implemented in intelligent vehicle. Real-time application is needed.

This work is only tested on Cityscapes dataset. It is possible to be implemented in other scenarios. For example, indoor instance segmentation. The best encoding dimension may vary from dataset to dataset. Another possible work to do with dataset is data augmentation. No data augmentation was carried out in this work.

Appendix A

Cityscapes

A-1 List of Cityscapes datasets

Table A-1: Cityscapes Dataset

	description
leftImg8bit_trainvaltest	left 8-bit images - train, val, and test sets (5000 images)
leftImg8bit_trainextra	left 8-bit images - trainextra set (19998 images)
gtFine_trainvaltest	fine annotations for train and val sets (3475 annotated images) and dummy annotations (ignore regions) for the test set (1525 images)
gtCoarse	coarse annotations for train and val set (3475 annotated images) and train_extra (19998 annotated images)
rightImg8bit_trainvaltest	right 8-bit images - train, val, and test sets (5000 images)
rightImg8bit_trainextra	right 8-bit images - trainextra set (19998 images)
disparity_trainvaltest	precomputed disparity maps using SGM for train, val, and test sets (5000 images)
disparity_trainextra	precomputed disparity maps using SGM for trainextra set
camera_trainvaltest	intrinsic and extrinsic camera parameters for train, val, and test sets
camera_trainextra	intrinsic and extrinsic camera parameters for trainextra set

Appendix B

Performance of different methods with 2D baseline

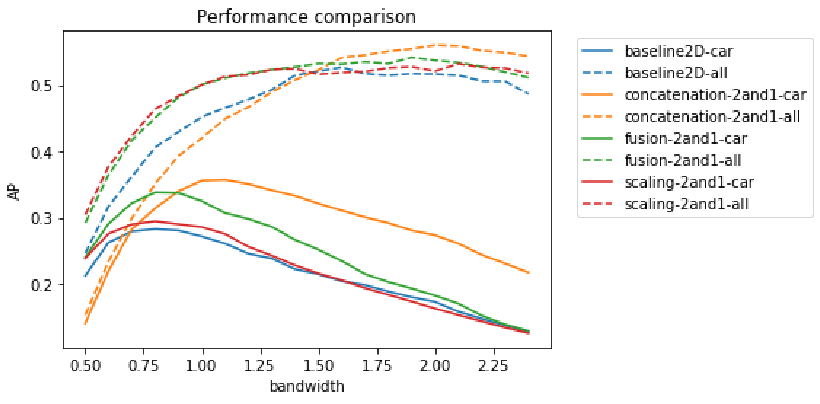


Figure B-1: Comparison of all methods in 'car' class

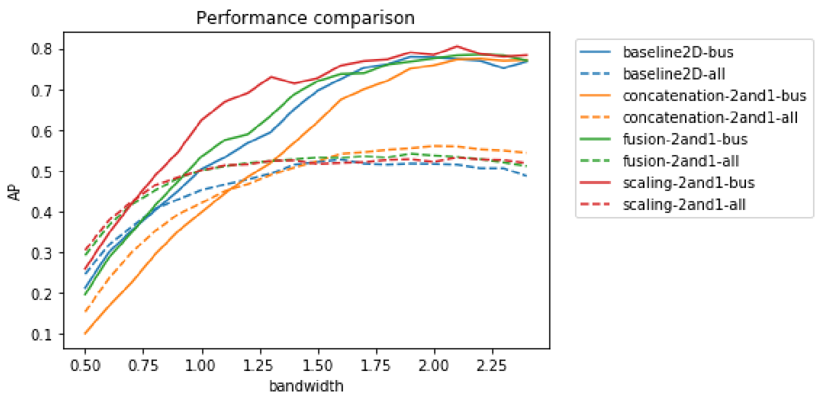


Figure B-2: Comparison of all methods in 'bus' class

Bibliography

[1] B. De Brabandere, D. Neven, and L. Van Gool, “Semantic instance segmentation with a discriminative loss function,” *Deep Learning for Robotic Vision, workshop at CVPR 2017*, 2017.

[2] L. Ye, Z. Liu, and Y. Wang, “Depth-aware object instance segmentation,” *Image Processing (ICIP), 2017 IEEE International Conference on*, pp. 325–329, 2017.

[3] S. O.-R. A. V. S. Committee *et al.*, “Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems,” *SAE International*, 2014.

[4] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” *European conference on computer vision*, pp. 740–755, 2014.

[5] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool, “Fast scene understanding for autonomous driving,” *CoRR*, vol. abs/1708.02550, 2017.

[6] A. Karpathy, “Cs231n convolutional neural networks for visual recognition,” *Neural networks*, vol. 1, 2016.

[7] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[8] V. Badrinarayanan, A. Handa, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling,” *Computer Vision and Pattern Recognition*, 2015.

[9] 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*, vol. 39, no. 12, pp. 2481–2495, 2017.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” *Computer Vision and Pattern Recognition*, pp. 2980–2988, 2017.

[11] J. Dai, K. He, Y. Li, S. Ren, and J. Sun, "Instance-sensitive fully convolutional networks," *European Conference on Computer Vision*, pp. 534–549, 2016.

[12] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2359–2367, 2017.

[13] P. O. Pinheiro, R. Collobert, and P. Dollár, "Learning to segment object candidates," *Advances in Neural Information Processing Systems*, pp. 1990–1998, 2015.

[14] B. Romera-Paredes and P. H. S. Torr, "Recurrent instance segmentation," *European Conference on Computer Vision*, pp. 312–329, 2016.

[15] L. Schneider, M. Jasch, B. Fröhlich, T. Weber, U. Franke, M. Pollefeys, and M. Rätsch, "Multimodal neural networks: Rgb-d for semantic segmentation and object detection," *Scandinavian Conference on Image Analysis*, pp. 98–109, 2017.

[16] H. Wei, H. Abrishami, X. Xiao, A. Karteek, *et al.*, "Adaptive video-based vehicle classification technique for monitoring traffic," tech. rep., Ohio Department of Transportation, Office of Statewide Planning & Research, 2015.

[17] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

[18] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[19] G. Bradski and A. Kaehler, "Opencv," *Dr. Dobb's journal of software tools*, vol. 3, 2000.

[20] 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.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[22] P. I. Labuhn and W. J. Chundrlik Jr, "Adaptive cruise control," Oct. 3 1995. US Patent 5,454,442.

[23] D. Belgiovane and C.-C. Chen, "Micro-doppler characteristics of pedestrians and bicycles for automotive radar sensors at 77 ghz," *Antennas and Propagation (EUCAP), 2017 11th European Conference on*, pp. 2912–2916, 2017.

[24] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[25] H. Cho, Y.-W. Seo, B. V. Kumar, and R. R. Rajkumar, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 1836–1843, 2014.

[26] L. Spinello, R. Triebel, and R. Siegwart, "Multiclass multimodal detection and tracking in urban environments," *The International Journal of Robotics Research*, vol. 29, no. 12, pp. 1498–1515, 2010.

[27] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[28] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *Computer Vision and Pattern Recognition*, 2016.

[29] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*, pp. 267–285, Springer, 1982.

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[32] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," *International Conference on Computer Vision*, pp. 1520–1528, 2015.

[33] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[34] R. Girshick, "Fast r-cnn," *Computer Vision and Pattern Recognition*, 2015.

[35] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, pp. 91–99, 2015.

[36] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," *Advances in neural information processing systems*, pp. 379–387, 2016.

[37] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[38] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," *European Conference on Computer Vision*, pp. 75–91, 2016.

[39] S. Dehaene and L. Cohen, "Dissociable mechanisms of subitizing and counting: Neuropsychological evidence from simultanagnosic patients.," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 20, no. 5, p. 958, 1994.

[40] G. Porter, T. Troscianko, and I. D. Gilchrist, "Effort during visual search and counting: Insights from pupillometry," *Quarterly journal of experimental psychology*, vol. 60, no. 2, pp. 211–229, 2007.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[43] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[44] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pp. 273–278, 2013.

[45] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," *Advances in Neural Information Processing Systems*, pp. 737–744, 1994.

[46] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 539–546, 2005.

[47] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2, pp. 1735–1742, 2006.

[48] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[49] M. Koestinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof, "Large scale metric learning from equivalence constraints," *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2288–2295, 2012.

[50] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on information theory*, vol. 21, no. 1, pp. 32–40, 1975.

[51] L. Schneider, M. Jasch, B. Fröhlich, T. Weber, U. Franke, M. Pollefeys, and M. Rätsch, "Multimodal neural networks: Rgb-d for semantic segmentation and object detection," *Scandinavian Conference on Image Analysis*, pp. 98–109, 2017.

[52] M. Brown, "Fisher's linear discriminant," *web document*, 1999.

[53] Z. Wu, C. Shen, and A. v. d. Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *arXiv preprint arXiv:1611.10080*, 2016.

[54] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," *European Conference on Computer Vision*, pp. 297–312, 2014.

[55] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[56] I. . 2011, "Road vehicles–vehicle dynamics and road-holding ability–vocabulary," 2011.

[57] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.