BDD100k Road Segmentation Report

Sungguk Cha
UNIST
sungguk@unist.ac.kr

I. Contents

This report describes the following BDD100k experiments:

- Introduction
- Related Works
- Experiment with *Dilated Residual Network*
- Experiment about crop size and receptive area with *Deeplab V3 plus*
- Experiment about sliding window prediction
- Experiment *Deeplab v3 plus* with *Xception*
- Experiment about multi-scale prediction
- Experiment with winners methods
- Experiment with prediction analysis
- Future work

II. Introduction

Semantic segmentation is a fundamental and challenging problem in computer vision, in which each pixel is assigned with a category label. It is a key step to understand visual scene, and plays a critical roll in applications such as auto-driving.

Driving scene parsing is a semantic segmentation task on driving scenes taken from the driver's point of view, which contains roads, people, traffic sign, buildings, etc. *Convolutional neural networks*(CNNs) has made leading progresses in the driving scene parsing.

The goal of this report is to figure out the best methods to solve BDD100k drivable area semantic segmentation problem in terms of mIoU score. To accomplish this I tried (1) using various *Fully Convolutional Networks* (FCNs) models, (2) data augmentation focusing on receptive field, resolution and erroneous samples, (3) prediction methods, (4) normalization methods. I present thorough experiments demonstrating the result and value of each methods.

The goal of the experiments is to achieve the highest performance on the BDD100k drivable area segmentation test dataset [6]. Scoring is done on the evaluation server[1] and real-time leaderboard is not provided.

Mostly, I experimented using single 12GB Nvidia Titan XP, where I could not sufficiently utilize batch normalization [29] via mini-batch statistics [32]. Therefore, I preferred group normalization which can be a good strategy with small amount of GPU memory as it is irrelevant to the batch size [32].

[1]https://bdd-data.berkeley.edu/

III. Related Works

BDD100k drivable area dataset [6]. Berkeley DeepDrive(BDD) dataset is a driving scene dataset which is large-scaled (100k number of images) and diverse in the number of the weathers, the cities, the time of a day and the scene type. The dataset has annotations for road object detection, instance segmentation, drivable area and lane markings. The drivable area task is a semantic segmentation task, in which driving road and alternative roads are included. BDD100k dataset contains 2 road classes (*i.e.*, driving road and alternative road) and one background class. It contains 70,000 (*train*), 10,000 (*val*), and 20,000 (*test*) pixel-level annotated images.

Backbone	Decoder	Normalization	mIoU Score
DRN-D-105		BN	79.76
Xception	Deeplab v3+	GN8	82.20
WiderResNet38	ASPP	SyncBN	82.61
ResNet101	Deeplab v3+	IBN-a	83.53
ResNet101	Deeplab v3+	IGN-a, 16	85.12
ResNet101	Deeplab v3+	GN16	**85.33**
Unknown	Deeplab v3+	Unknown	84.01
WiderResNet38	ASPP	Inplace abn	86.04
ResNet101/152	PSP/A	IBN-a	**86.18**

Fig. 1. Experimental results on the testset public score with respect to encoder-decoder-normalization combinations. Xception [22] configuration detail in [16]. ResNet [13] and refined as [31]. WiderResNet [14]. ASPP: atrous spatial pyramid pooling module [18]. Deeplabv3+ [16]. PSP/A: ensemble of PSPNet [28] and PSANet [27] with ResNet101 and 152 for each. BN: batch normalization [29]; SyncBN: synchronized batch normalization over GPUs; GN X: group normalization [32] with grouping number X; IBN-a: instance batch normalization for some layers [33]; IGN-a, X: experimented instance group normalization for some layers as well as IBN-a, default normalization is group normalization, X denotes the numbers of grouping channel for both ign and gn. Inplace abn [34]. Last three rows are the first, second and third placed winners[3].

Fully Convolutional Networks. Models based on FCNs [8], [9] have shown significant improvements on several segmentation benchmarks [1]–[5]. There are some FCNs model variants which aim to gather contextual information for segmentation [12], [13], [22]. Using the FCNs, there are encoder-decoder structured models [11], [16], [27], including those which employ multi-scale structure (*i.e.*, pyramidal modules)

[3]CVPRW2018:Workshop on Autonomous Driving:BDD100k drivable area segmentation winners.
First placed: CUHK, SenseTime, Tencent
Second placed: Mapillary
Thrid placed: DiDi AI Lab

[16], [28]. In this work, I demonstrate how did I use those models to solve BDD100k segmentation problem.

Data augmentations. In a practice, resources such as memories and computational power are restricted in time or physically. It is one of the most critical part in deep learning engineering to crop training images appropriately and efficiently. Several methods which variate an image such as flip, scaling and blurring have been efficient and critical as much as augmented dataset. I will introduce my experiments that I have tried with various combinations of augmentations focusing on resolution, receptive field over an whole image and erroneous samples.

Normalizations. In 2015, Batch normalization [29] was published which was believed to reduce internal covariate shift between convolutional layers and showed significant improvements in performances over computer vision benchmarks (*i.e.,* classifications, object detection and segmentation) and also reducing training time to converge. After the invention, many algorithms regarding normalization are studied [32]–[34], including analysis or developments over batch normalization [37], [38].

IV. Experiment with Dilated Residual Network

Without additive head nor decoder variation, I started with Dilated Residual Network(DRN) [12], removing the last global average pooling layer which crumbles spatial features [9]. DRN-D-X models are suggested models for semantic segmentation tasks, and X denotes the number of dilated residual layers.

Backbone	Decoder	Normalization	Batchsize	Cropsize	mIoU Score
DRN-D-22		BN	8	(640, 640)	78.09
DRN-D-105		BN	8(2)	(1280, 720)	79.76

Fig. 2. Fine-tune from Cityscapes [3] pretrained model. Batch size 8(2) denotes 8 images per a mini-batch and 2 images per a GPU.

Fig. 3. Prediction results from DRN-D-22 trained with cropsize (640, 640). The red lanes are predicted *driving roads* and the blue lanes are predicted *alternative roads*.

I trained DRN-D-22 with cropsize 640 (random (640, 640) image crop out of the original (1280, 720) sized image). Many predictions by DRN-D-22 have multiple *driving road* which is non-sense (Fig. 3). I doubted the problem in two folds due to (1) *griding artifacts* that is also issued in [12], (2) a small receptive field which can make the model confused(Fig. 4). So, I trained DRN-D-105, which is deeper model than DRN-D-22, with a whole image input rather than cropping (640, 640). As a result, DRN-D-105 trained with original image, full receptive field, also had more than one driving road predictions, thus I concluded that the network has a such limitation.

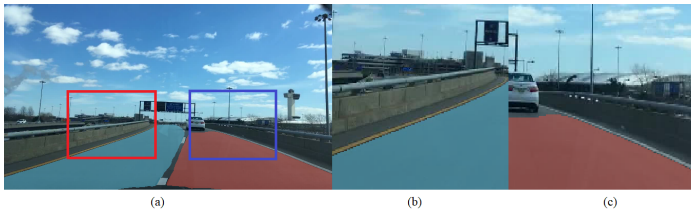


Fig. 4. Prediction sample. (b) and (c) are cropped samples from (a). Given an image (b) or (c), it is very hard to determine if the road on the scene is (1) driving road, (2) alternative road, (3) none of both, compared to given image (a). Concluding that having larger receptive area is a key in training.

V. Experiment about crop size and receptive area with Deeplab V3 Plus

Deeplabv3+ [16] has shown great successes in semantic segmentation benchmarks [1], [3]. In a sense that BDD100k dataset [6] succeeds Cityscapes dataset [3], Deeplabv3+ would work in BDD100k dataset as well as in Cityscapes dataset. So I decided to train Deeplabv3+. Referred Pytorch [7] based Deeplabv3+ implementation[4]. I used ImageNet pretrained ResNet101 as backbone network and fine-tuned. I changed the stride convolution layer of the last layer of ResNet101 into dilation convolution layer, keeping the same output stride but pursuing better receptive field.

Data augmentation. I started with the same training protocol as in [17]. In add to the protocol, I added random Gaussian blur data augmentation. In short, crop size 513, batch size 8, learning rate 0.02 (10 times larger for decoder (*i.e.,* ASPP [17] and the additional decoder) parts) with polynomial scheduling, weight decay 0.0005, momentum 0.9, training epoch 50 and output stride 16 for the backbone network.

- random scale crop: random scale 0.5 2.0 then crop.
- random Gaussian blur: by half probability, Gaussian blur with random sized radius.
- normalize: normalizing input image's RGB values by mean and std of BDD100k dataset.

Crop size and receptive area. Top models [11], [16] on the scene parsing segmentation benchmark [1] exploit crop size 513 or less. At the same time, they use *random scale* (from 0.5 to 2.0) data augmentation for training. Considering the training image shape (1280, 720), the random scaling ratio and crop size, the expected receptive area result in comparatively small, while training over *BDD100k* dataset seems to require larger receptive area than other benchmarks (Fig. 4.).

[4]https://github.com/jfzhang95/pytorch-deeplab-xception

$$ReceptiveArea(x, (a, b)) := (x/a, b/x)$$
$$E[RandomScaleCoefficient] = 1.25$$
$$RecpetiveArea(513, (1280, 720) \times 1.25)$$
$$= ReceptiveArea(513, (1600, 900)) = (0.321, 0.57)$$

Training with 513 crop size could not catch up the score of DRN did in section IV. After the experiment and the analysis above, I increased the cropsize and experimented two fold.

- Random scale (0.5 2.0) then crop (720, 720)
- No random scale and no crop.

Normalization. Batch normalization is utilizing mini-batch statistics, so it results better if it has larger number of mini-batch size. Because I incremented crop sizes for larger receptive area, batch size becomes extremely small (*e.g.,* 4 and 2). With small number of batch size, according to [32], group normalization performs better than batch normalization. Thus I replaced every batch normalization layer into group normalization with grouping 16 channels.

Backbone	Decoder	Normalization	Batchsize	Cropsize	mIoU Score
DRN-D-22		BN	8	(640, 640)	78.09
DRN-D-105		BN	8(2)	(1280, 720)	79.76
R101	Deeplabv3+	GN	8	(513, 513)	75.4
R101	Deeplabv3+	GN	4	(720, 720)	81.88

Fig. 5. Experiment V results.

VI. Experiment about sliding window predictions

It is very reasonable idea that a network in validation process must work better with a cropsize that the model is trained with. There are a few tricks based on the idea. One is reducing an image's size to the cropsize, feed it, and upsample the result. I do not think it is an appropriate approach for this task. For example, traffic lines which are thin and small, but very important in my task (as we could observe the traffic line dependency when I trained the DRN) can be non-recognizable ones if they are down sampled. So I decided to use sliding window for evaluation.

I used a model trained with 513 cropsize. As the input image is (1280, 720) shaped, I sliced an image into 6 slices and combined them. For the overlapped area, I used stacking method (*i.e.,* averaging prediction results)(Fig. 7.). The result was not improved. I concluded that each prediction with 513 cropped images has less receptive area, so the performance of the stacking could not work better.

Backbone	Decoder	Normalization	Batchsize	Cropsize	mIoU Score	Etc
DRN-D-22		BN	8	(640, 640)	78.09	
DRN-D-105		BN	8(2)	(1280, 720)	79.76	
R101	Deeplabv3+	GN	8	(513, 513)	80.5	
R101	Deeplabv3+	GN	8	(513, 513)	79.9	Sliding window
R101	Deeplabv3+	GN	4	(720, 720)	81.88	

Fig. 6. Experiment VI results.

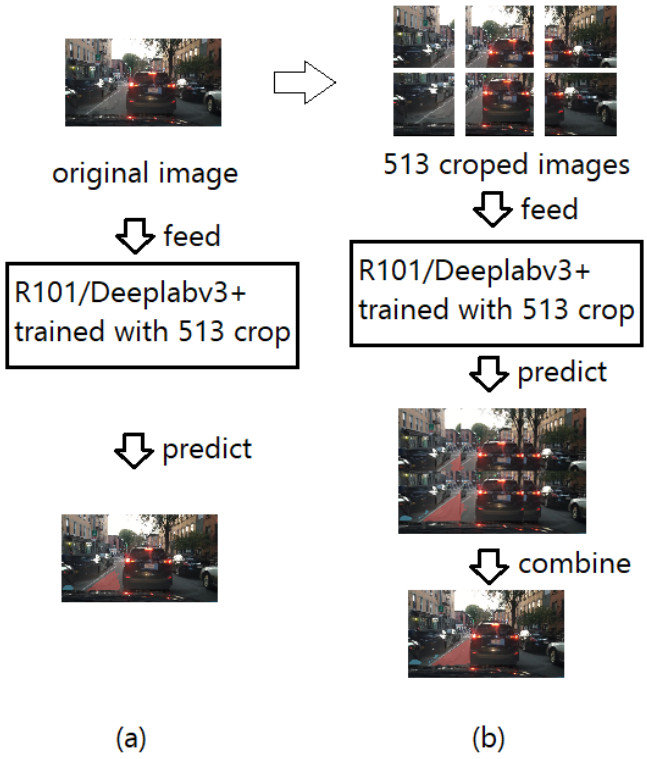


Fig. 7. Sliding window technique with cropsize 513. (a) illustrates original(1280,720)-image-fed prediction. (b) shows sliding window prediction procedure. For the overlapped regions, most confident prediction is chosen.

VII. Experiment Deeplab v3 plus with Xception

Using group normalization, I could achieve better scores. I wondered if group normalization works better with depthwise-separable convolution based networks [22]–[25]. MobileNetV2 [24] and ShuffleNet [25] shuffle channels during layer wise advance, so the cooperation between group normalization and them. Xception utilizes depthwise separable convolutions and shows great performances with Deeplab v3+ in several benchmarks [1], [3]. I used Xception pretrained in ImageNet and modified it into Deeplabv3+ adapted configuration [16]. As Xception has smaller GCD of number of channels over layers, the performance was not as good as ResNet.

Backbone	Decoder	Normalization	Batchsize	Cropsize	mIoU Score	Etc
DRN-D-22		BN	8	(640, 640)	78.09	
DRN-D-105		BN	8(2)	(1280, 720)	79.76	
R101	Deeplabv3+	GN16	8	(513, 513)	80.5	
R101	Deeplabv3+	GN16	8	(513, 513)	79.9	Sliding window
R101	Deeplabv3+	GN16	4	(720, 720)	85.15	
R101	Deeplabv3+	GN16	2	(1280, 720)	**85.33**	
Xception	Deeplabv3+	GN8	6	(640, 360)	82.20	Half scaled input

Fig. 8. Experiment VII results.

VIII. Experiment about multi-scale prediction

Multi-scaling prediction is scaling an input image and stacking the predictions. It is pouring more computation and

has shown considerable improvements over benchmarks [2], [4] like post-processing algorithms [10]. I applied multi-scale prediction with [0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0] scaling coefficients to 720 crop trained model which is trained with random scaling.

Backbone	Decoder	Normalization	Batchsize	Cropsize	mIoU Score	Etc
DRN-D-22		BN	8	(640, 640)	78.09	
DRN-D-105		BN	8(2)	(1280, 720)	79.76	
R101	Deeplabv3+	GN16	8	(513, 513)	80.5	
R101	Deeplabv3+	GN16	8	(513, 513)	79.9	Sliding window
Xception	Deeplabv3+	GN8	6	(640, 360)	82.20	Half scaled input
R101	Deeplabv3+	GN16	4	(720, 720)	84.33	Multi-scale pred.
R101	Deeplabv3+	GN16	4	(720, 720)	85.15	Random scale
R101	Deeplabv3+	GN16	2	(1280, 720)	**85.33**	

Fig. 9. Experiment VIII results.

Multi-scale prediction did not improve the result. Multi-scale prediction is regarded to be good to detect object with multi-scales in the segmentation tasks (*e.g.,* variously scaled cycle wheels in Cityscapes dataset [3]). The classes of BDD100k are hard to be seen variously scaled. The not-improved score with multi-scale prediction might be due to the comparatively constantly scaled classes.

IX. Experiment with Winners methods

The WAD2018 drivable area segmentation winners did not report their experimental details to the public. I found their model implementations roughly and tried to train with their methods.

Backbone	Decoder	Normalization	Batchsize	Cropsize	mIoU Score	Etc
DRN-D-22		BN	8	(640, 640)	78.09	
DRN-D-105		BN	8(2)	(1280, 720)	79.76	
R101	Deeplabv3+	GN16	8	(513, 513)	80.5	
R101	Deeplabv3+	GN16	8	(513, 513)	79.9	Sliding window
Xception	Deeplabv3+	GN8	6	(640, 360)	82.20	Half scaled input
WRN38	ASPP	SyncBN	16(4)	(1280, 720)	82.61	
R101	Deeplabv3+	SyncIBN-a	16(4)	(1280, 720)	83.53	
R101	Deeplabv3+	GN16	4	(720, 720)	84.33	Multi-scale pred.
R101	Deeplabv3+	IGN16-a	2	(1280, 720)	85.12	
R101	Deeplabv3+	GN16	4	(720, 720)	85.15	Random scale
R101	Deeplabv3+	GN16	2	(1280, 720)	**85.33**	

Fig. 10. Experiment IX results.

A. Instance Batch Normalization Networks(IBNNet)

IBNNet [33] has some instance normalization [35] and batch normalization combined building blocks. According to [33], [36], instance normalization has shown better performance in style transfer tasks by getting contents better than batch normalization. As the competition task seems to be more about contents than style, they tried to induce appearance invariance via adopting instance normalization mixed batch normalization.

They also proposed PSANet [27] along with PSPNet [28], but the performances of them on the benchmarks are not leading. I doubted the performances of the models, but, instead, I tried their IBN with Deeplabv3+. As a futher step, I suggested IGNNet which replaces every batch normalization with group normalization. I experimented with IGN16-ResNet101-Deeplabv3+ and IBNNet101-Deeplabv3+.

B. Inplace activation batch normalization (inplace abn)

Inplace abn [34] is activation layer and batch normalization combined layer with leaky ReLU [34] instead of ReLU for activation function. The second placed winner used inplace abn with WiderResNet38 [14]-ASPP [17]. Instead of implementing inplace abn, I tested if the model is good. I used [15] configuration for WiderResNet38, ASPP and synchronized batch normalization [30].

X. Experiment with prediction analysis

I have listed every image with its score(Fig. 11-13.). More than 90% of training set and more than 75% of validation set has "¿ 0.9" score. The number of images "¡ 0.9" score is about 10,000 (train 7k + val 3k). My hope there was that "there are cases which have less numbers to train and most cases which are scored bad should be the case." Compared to the conventional semantic segmentation tasks, BDD100k drivable dataset is very subjective in terms of 'the divisions of roads'. In the conventional tasks, a class should be the class in the annotation objectively (e.g., a bird should be annotated as a bird). However in the BDD, I could find that many annotations are subjectively labeled (e.g., "it is a beautiful bird."). I am arguing their annotations are inconsistent for confusing examples. Also there are a lot of wrongly labeled data.

Scores	Number	Percentage
>0.95	41448	59.21%
>0.90	21576	30.82%
>0.85	3285	4.69%
>0.80	811	1.16%
>0.75	304	0.43%
>0.70	156	0.22%
>0.65	403	0.58%
>0.60	1185	1.69%
>0.55	216	0.31%
>0.50	88	0.13%
>0.45	431	0.62%
>0.40	27	0.04%
>0.35	5	0.01%
>0.30	56	0.08%
>0.25	8	0.01%
>0.20	1	0.00%
>0.15	0	0.00%
>0.10	0	0.00%
>0.5	0	0.00%
>0.0	0	0.00%

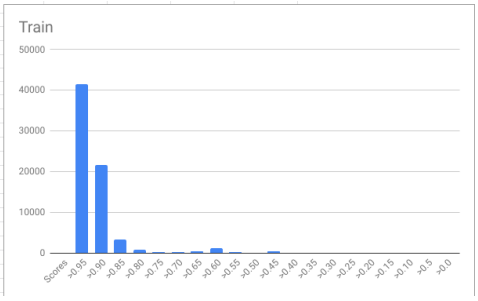


Fig. 11. Prediction scores in train set with my best model.

Scores	Number	Percentage
>0.95	3868	38.68%
>0.90	2697	26.97%
>0.85	865	8.65%
>0.80	414	4.14%
>0.75	249	2.49%
>0.70	175	1.75%
>0.65	161	1.61%
>0.60	451	4.51%
>0.55	269	2.69%
>0.50	209	2.09%
>0.45	354	3.54%
>0.40	121	1.21%
>0.35	50	0.50%
>0.30	112	1.12%
>0.25	4	0.04%
>0.20	1	0.01%
>0.15	0	0.00%
>0.10	0	0.00%
>0.5	0	0.00%
>0.0	0	0.00%
	10000	

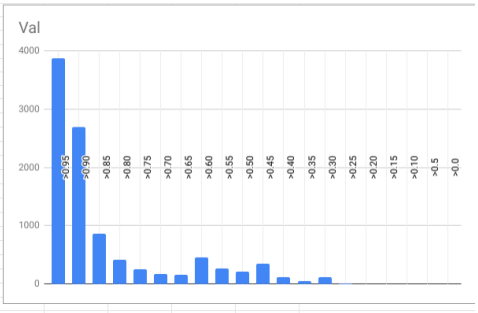


Fig. 12. Prediction scores in validation set with my best model.

train+val	Scores	Number	Percentage
	>0.95	45316	56.65%
	>0.90	24273	30.34%
	>0.85	4150	5.19%
	>0.80	1225	1.53%
	>0.75	553	0.69%
	>0.70	331	0.41%
	>0.65	564	0.71%
	>0.60	1636	2.05%
	>0.55	485	0.61%
	>0.50	297	0.37%
	>0.45	785	0.98%
	>0.40	148	0.19%
	>0.35	55	0.07%
	>0.30	168	0.21%
	>0.25	12	0.02%
	>0.20	2	0.00%
	>0.15	0	0.00%
	>0.10	0	0.00%
	>0.5	0	0.00%
	>0.0	0	0.00%
		80000	

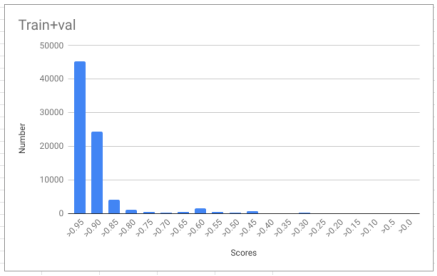


Fig. 13. Prediction scores in train and validation set with my best model.

Eventually, it was not beneficial in the final score. I assumed that the annotations are inconsistently (very subjective and in consistent). The first clues are my observation on my poorly scored inferences and the trial above. If the labeling is consistent and I was missing the low numbered cases, the final should (or would like to) get improved, while it was degraded (it is possible that my method was not good enough to improve).

Backbone	Decoder	Normalization	Batchsize	Cropsize	mIoU Score	Etc
DRN-D-22		BN	8	(640, 640)	78.09	
DRN-D-105		BN	8(2)	(1280, 720)	79.76	
R101	Deeplabv3+	GN16	8	(513, 513)	80.5	
R101	Deeplabv3+	GN16	8	(513, 513)	79.9	Sliding window
Xception	Deeplabv3+	GN8	6	(640, 360)	82.20	Half scaled input
WRN38	ASPP	SyncBN	16(4)	(1280, 720)	82.61	
R101	Deeplabv3+	SyncIBN-a	16(4)	(1280, 720)	83.53	
R101	Deeplabv3+	GN16	4	(720, 720)	84.00	Repeated erroneous 10k
R101	Deeplabv3+	GN16	4	(720, 720)	84.33	Multi-scale pred.
R101	Deeplabv3+	IGN16-a	2	(1280, 720)	85.12	
R101	Deeplabv3+	GN16	4	(720, 720)	85.15	Random scale
R101	Deeplabv3+	GN16	2	(1280, 720)	**85.33**	

Fig. 14. Experiment X results.

XI. Future work

I have a plan to ensemble models by stacking results. In order to have models to ensemble, I will train several more models with (1) architecture refinement (*e.g.,* converting dilation convolutions into stride convolutions at the last layers of ResNet), (2) training with other models such as ResNet152, NAS [39]-generated models, (3) rebuilding data augmentation strategy (*e.g.,* removing erroneous samples(Fig. 16.) in training).

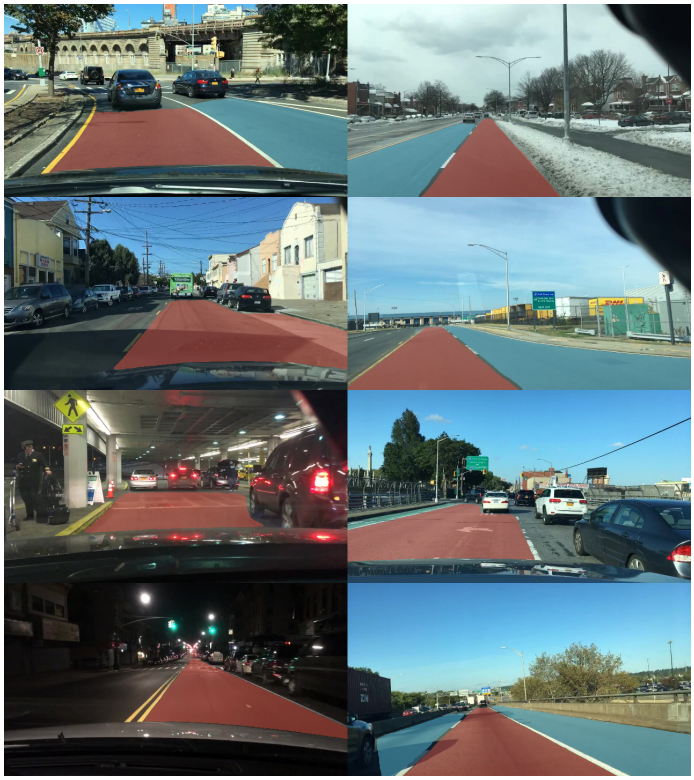


Fig. 15. Visualized predictions.

References

[1] Everingham, Mark, et al. "The pascal visual object classes challenge: A retrospective." International journal of computer vision 111.1 (2015): 98-136.

[2] Mottaghi, Roozbeh, et al. "The role of context for object detection and semantic segmentation in the wild." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014.

[3] Cordts, Marius, et al. "The cityscapes dataset for semantic urban scene understanding." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[4] Zhou, Bolei, et al. "Scene parsing through ade20k dataset." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[5] Caesar, Holger, Jasper Uijlings, and Vittorio Ferrari. "Coco-stuff: Thing and stuff classes in context." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[6] Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V. and Darrell, T., 2018. BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling. arXiv preprint arXiv:1805.04687.

[7] Paszke, Adam, et al. "Automatic differentiation in pytorch." (2017).

[8] Sermanet, Pierre, et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks." arXiv preprint arXiv:1312.6229 (2013).

[9] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[10] He, Xuming, Richard S. Zemel, and Miguel . Carreira-Perpin. "Multiscale conditional random fields for image labeling." Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.. Vol. 2. IEEE, 2004.

[11] Wu, Huikai, et al. "FastFCN: Rethinking Dilated Convolution in the Backbone for Semantic Segmentation." arXiv preprint arXiv:1903.11816 (2019).

[12] Yu, F., Koltun, V. and Funkhouser, T.A., 2017, July. Dilated Residual Networks. In CVPR (Vol. 2, p. 3).

[13] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

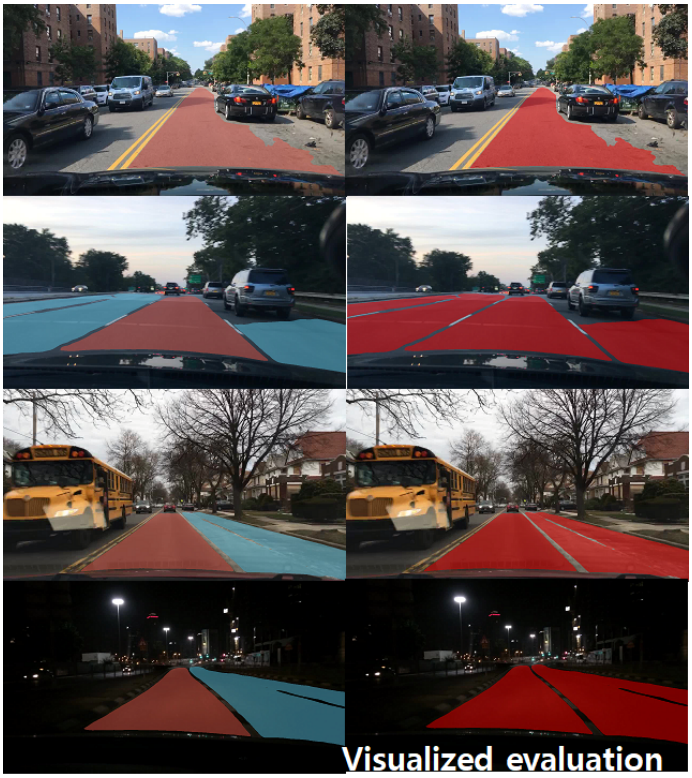
[14] Zagoruyko, Sergey, and Nikos Komodakis. "Wide residual networks." arXiv preprint arXiv:1605.07146 (2016).

[15] Wu, Zifeng, Chunhua Shen, and Anton Van Den Hengel. "Wider or deeper: Revisiting the resnet model for visual recognition." Pattern Recognition 90 (2019): 119-133.

[16] Chen, Liang-Chieh, et al. "Encoder-decoder with atrous separable convolution for semantic image segmentation." Proceedings of the European Conference on Computer Vision (ECCV). 2018.

[17] Chen, Liang-Chieh, et al. "Rethinking atrous convolution for semantic image segmentation." arXiv preprint arXiv:1706.05587 (2017).

[18] Chen, Liang-Chieh, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." IEEE transactions on pattern analysis and machine intelligence 40.4 (2017): 834-848.



(a) (b)

Fig. 16. Visualized prediction with erroneous labeled samples. (a) images are the predictions. Red pixels in (b) are visualized evaluation which means there is nothing in the labels, but I marked.

[19] B. Hariharan, P. A. Arbelaez, R. B. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In CVPR, 2015.

[20] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

[21] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In ICCV, 2015.

[22] Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. arXiv preprint, pp.1610-02357.

[23] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[24] Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[25] Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[26] Bul, S.R., Porzi, L. and Kontschieder, P., 2017. In-place activated batchnorm for memory-optimized training of DNNs. CoRR, abs/1712.02616, December, 5.

[27] Jia, J., 2018. Psanet: Point-wise spatial attention network for scene parsing.

[28] Zhao, Hengshuang, et al. "Pyramid scene parsing network." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

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

[30] Tamaki Kojima. Pytorch-syncbn. https://github.com/tamakoji/pytorch-syncbn

[31] Sam Gross and Michael Wilber. Training and investigating Residual Nets. http://torch.ch/blog/2016/02/04/resnets.html. 2016

[32] Wu, Y. and He, K., 2018. Group normalization. arXiv preprint arXiv:1803.08494.

[33] Pan, Xingang, et al. "Two at once: Enhancing learning and generalization capacities via ibn-net." Proceedings of the European Conference on Computer Vision (ECCV). 2018.

[34] Rota Bul, Samuel, Lorenzo Porzi, and Peter Kontschieder. "In-place activated batchnorm for memory-optimized training of dnns." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[35] Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Instance normalization: The missing ingredient for fast stylization." arXiv preprint arXiv:1607.08022 (2016).

[36] Huang, Xun, and Serge Belongie. "Arbitrary style transfer in real-time with adaptive instance normalization." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[37] Santurkar, S., Tsipras, D., Ilyas, A. and Madry, A., 2018. How Does Batch Normalization Help Optimization?(No, It Is Not About Internal Covariate Shift). arXiv preprint arXiv:1805.11604.

[38] Ioffe, S., 2017. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In Advances in Neural Information Processing Systems (pp. 1945-1953).

[39] Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." arXiv preprint arXiv:1611.01578 (2016).