2020 IEEE Intelligent Transportation Systems Conference (ITSC)
Rhodes, Greece. September 20-23, 2020 (Virtual)

Railroad semantic segmentation on high-resolution images

Sergey Belyaev, Igor Popov, Vladislav Shubnikov, Pavel Popov, Ekaterina Boltenkova, Daniil Savchuk

Abstract—**Recent advances in machine learning research could significantly alter the railroad industry by deploying fully autonomous trains. To achieve effective interaction between self-driving trains and the environment, an accurate long-range railway detection should be provided. In this paper, we propose a framework for the rail tracks segmentation on high-resolution images (2168x4096). The announced approach accelerates inference speed 6 times, by using two neural networks. The proposed architecture and its training approach provide a long-range railway segmentation within 150 meters, achieving 20 fps. Also, we propose an auxiliary algorithm detecting possible paths among all the found ones. To determine which data labeling approach has a higher impact, additional experiments were performed. The proposed framework provides a balanced tradeoff between computing efficiency and performance in the railroad segmentation problem.**

I. Introduction

One of the major challenges in self-driving train engineering is to design a supervising system able to detect possible obstacles occurring along a current route. However, not always the route can be exactly determined. Railways are tied up with railroad switches; thus, a train can be guided from the main track to adjacent ones. Assuming the switches states cannot always be precisely determined, all the railways, a train can proceed moving from the current location, should be examined. Such railways are referred to as *possible tracks*. (Fig. 1)

Obstacles detection accuracy implicitly depends on the possible track detection performance. Consequently, possible tracks detection is meant to be one of the base tasks in self-driving train control. Source data for the task are presented as video frames from a high-resolution camera (2168x4096) stored on a vehicle body. To detect which pixels are assigned to the rail tracks, image segmentation should be performed. Finally, among all the found tracks, possible ones are determined.

In the following paper, an approach for low-speed trains is described. An average speed of such trains is approximately 60-70 km/h. Accordingly, the braking distance tends to 118 meters at 60 km/h and 157 meters at 70 km/h with an estimated emergency braking speed of 1.2 m/s.

S. Belyaev, V. Shubnikov, D. Savchuk are with Peter the Great St. Petersburg Polytechnic University, 29, Polytechnicheskaya, 195251 St. Petersburg, Russia, email: bel_s_u@mail.ru (S. Belyaev), vlad.shubnikov@gmail.com, (V. Shubnikov) dsavchuk@itsociety.su (D. Savchuk)

I. Popov, P. Popov, E. Boltenkova are with JSC «NIIAS», 114, Naberezhnaya Obvodnogo kanala, 190005, St. Petersburg, Russia, email: ig.popov1997@gmail.com (I. Popov), pavel.niias@gmail.com (P. Popov), kitkat52@yandex.ru (E. Boltenkova)

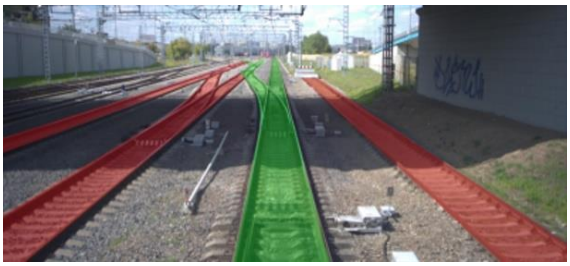

Figure 1. The possible tracks are marked with green color

Therefore, the range of a railway detection must be guaranteed to be at least 150 meters for self-driving trains of the low-speed class.

Apart from providing high performance, inference time optimization should be performed. It is crucial to achieving the segmentation result as fast as it could be to further detect and report possible obstacles properly while train proceeds moving. In practice, it should not take more than 25-50 milliseconds (10-20 fps, respectively).

To our knowledge, the current best result in the longest range covering and the best performance was presented in [1]. The authors used a convolutional neural network (CNN) to achieve 89% IoU and 20 fps speed. However, they were not able to detect the railway within 118 meters range, as was mentioned in their work.

In the following paper, we propose an approach of detecting railways that guarantees to cover the 150 meters range. The proposed model is claimed to achieve better accuracy and range, compared with the given approach in [1], although saving a competitive speed of 20 fps. The model is meant to be improving performance by using weighted IoU loss function, exploiting a foreshortening effect. Also, in this paper, it was experimentally determined that all the visible railways dataset labeling provides better accuracy than only possible tracks labeling. To extract the possible tracks among all the found ones, a special algorithm was proposed. During the research, no other alternatives were found.

Our main contributions are summarized as follows:

- Long-range railways segmentation (at least 150 meters)
- The framework outperforming the proposed model in [1] in performance and range covering
- Double network approach achieving inference speed and performance tradeoff.
- Weighted loss function exploiting a foreshortening effect
- Experimentally proved preference of all visible railways labeling
- Possible tracks identification algorithm

978-1-7281-4148-0/20/$31.00 ©2020 IEEE

II. Related Work

Traditional computer vision railway detection techniques use linear and ribbed objects to provide railway detection [2,3,4,5,6,7]. These methods are considered to achieve high computing efficiency, but their performance appears to be strongly dependent on the scene characteristics, as mentioned in [1]. Such algorithms are weakly robust to process railroad switches and railroad junctions, line gaps, obstructions and image noise.

Recent advances in deep learning have introduced convolutional neural networks (CNN) to the semantic segmentation problems. For the first time, a CNN was successfully applied to the semantic segmentation problem in [8], where UNet architecture was presented. In [1], a CNN architecture for the railroad semantic segmentation was proposed.

In terms of limited computational resources and high-performance demand, recent research was aimed to design fast and accurate real-time semantic segmentation frameworks. The first paper reporting a real-time semantic segmentation approach [9], presents ENet architecture. However, the proposed model fails to maintain high-performance results because of a small number of parameters, leading to a poor discriminative ability.

Recent real-time segmentation frameworks achieve outstanding results by using lightweight components in their structure to provide high speed and effective techniques to maintain the performance.

In [10] to enlarge the receptive field spatial pyramid pooling (SPP) module was proposed, which merges deep representation features with different granularities. PSPNet in [11] uses a convolutional adaptation of SPP module to aggregate contextual data from the feature extractor's low-level layer and upsamples its output to the original resolution. Instead of upsampling, SwiftNet [12] fuses SPP output with high-level features from lateral connections in the segmentation mask restoration process, thus achieving outstanding results on several benchmarks. Authors of DeepLab proposed Atrous SPP (ASPP) module in [13,14,15]. It exploits dilated convolution, which allows to effectively expand the receptive field to incorporate larger context without increasing the number of parameters or the number of computations. ASPP modifications were studied in [16,17,18,19]. The goal of these modifications was related to both: improving performance [16,18,19] and reducing the time spent on block evaluation [17,20]. Attention mechanism provides focusing on important features and suppressing unnecessary ones, thus strongly improving the accuracy. Different architectures of attention blocks were discussed in [21,22,23,24].

To remedy information loss, ICNet [25] employs multiscale images as an output and a cascade network structure to increase computing efficiency. BiSeNet [26] decouples the workflow into a spatial path and a context path to reduce computational complexity. The second version of SwiftNet instead of using SPP proceeds with fusing shared features at multiple resolutions to enlarge receptive field. In [27,28] various decoder schemes using lateral connections from corresponding encoder layers were discussed. In our architecture, SPP module was used on the backbone's low-level feature representations like [12] and squeeze-and-excitation attention mechanism modules on lateral connections like [26].

In [29] to achieve high computational efficiency, an auxiliary CNN was used, detecting image region containing small details. This region is processed in higher resolution comparing to the whole image. We exploit a similar technique, but in our case, the region where high-resolution should be examined is already determined. The approach provides a CNN evaluation of this region regardless of the rest parts of the images.

III. Proposed Segmentation Method

In this section, our framework for the railroad semantic segmentation is described. First, we describe a dataset, used to solve the segmentation task, following the proposed single neural network architecture. Then a special approach is presented, which significantly speeds up the inference time for the high-resolution images. Finally, we provide a training technique and *possible tracks* identification algorithm.

A. Dataset

In this paper, to solve the railway segmentation problem on a range within at least 150 meters, a special dataset was used. It consists of 2900 labeled images, captured by a high-resolution (2168x4096) camera mounted on the train body. The labeling was made on a range of at least 150 meters. Such dataset is referred to as DS2 in this paper. The dataset was split by train and test with test size 25% (725 images). All the following evaluation results were obtained on the test set.

B. Proposed architecture

In our architecture, we combine the ideas described in *Section II*.

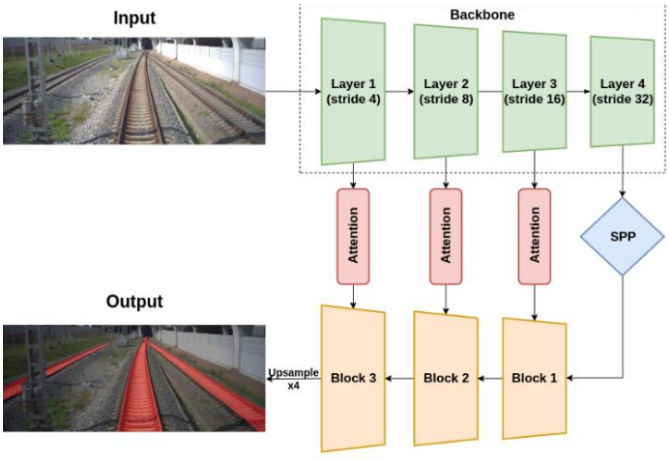

Figure 2. RailCNN Architecture. UNet-like model extended by Attention and SPP modules. Decoder block represents low-level features blending with a Conv3x3 layer, following Batch Normalization and ReLU activation. Then element-wise summarized with attention output, following by rate 2 upsample.

To improve the prediction accuracy, the model proposed in [1] is modified with SPP module and Attention blocks, as shown in Fig.2. The scheme of convolutional adaptation of SPP module (Fig. 3a) was borrowed from [12] and squeeze-and-excitation self-attention blocks (Fig. 3b) – from [26].

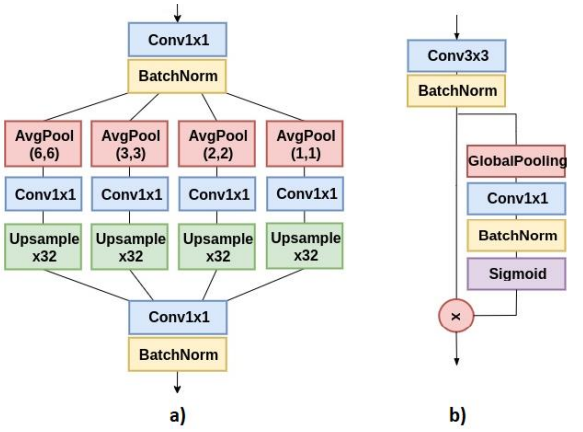

Figure 3. (a) - SPP module architecture, (b) - Attention module architecture

The provided modification is referred to as RailCNN. The proposed architecture used lightweight backbone - ResNet34 pre-trained on ImageNet to extract feature representations.

The main goal of the provided modification is to improve railway prediction performance. Table 1 reports the comparison of experimental results between two approaches: RailNet proposed in [1] and RailCNN. As far as DS3, representing the dataset used in [1], is not publicly available, RailCNN experiments were held on datasets: DS1 - public dataset RailSem19 [30] with its 4000 labeled images in 1920x1080 resolution and DS2. All the experiments were held with images converted to 1920x1080 resolution.

TABLE I. RAILNET AND RAILCNN COMPARISON. DS1 – PUBLIC RAILSEM19 DATASET, DS2 – DATASET USED IN THE CURRENT RESEARCH, DS3 - DATASET USED IN [1]

Model	Dataset	IoU	fps
RailCNN	DS1	0.931	17
RailCNN	DS2	0.940	17
RailNet	DS3	0.898	20

C. *Improving inference speed*

To process high-resolution images and not to lose much valuable information while downsampling, we propose to split the images into two pieces: A and B, as shown in Fig.4. G-level is an estimation of the horizon, considering its possible fluctuations because of the swinging effect. Thus, the level G can be approximately defined as maximum horizon level of all the images in the dataset. Thus, the level G can be approximately defined as maximum horizon level of all the images in the dataset (Y coordinate of the highest mask pixel of the train dataset).

By definition, railroads cannot appear above G-level; thus, the region above the G level is discarded and isn't used

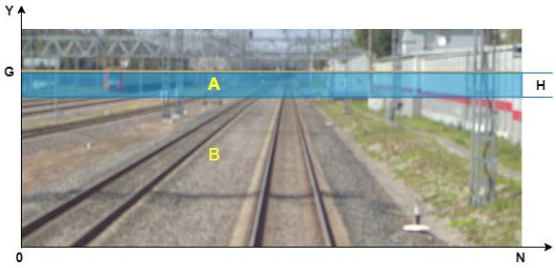

Figure 4. Regions of interest A and B. G - horizon level, H - region height, N - image width

either during the training nor inference. The rest of the region is split into two subregions: A and B. The subregion A is placed straight below the G level with height H. The rest part is taken by B subregion. While training: A-regions of the images are used in full resolution, and B-regions are downsampled with factor m. It allows to catch valuable information on a long distance and keep a suitable inference speed, achieving high segmentation accuracy.

While inference, both are used for their regions. Total inference time behaves in proportion to the joint number of pixels in A and B regions to process:

$$T = HN + (G - H)N/m^2 \quad (1)$$

Replacing $H = kG$, where $k\epsilon[0,1]$, obtain

$$T = \frac{NG(1 + k(m^2 - 1))}{m^2} \quad (2)$$

Through varying parameters k and m, we can balance performance and inference time: greater the value k, provides higher performance and lower computing efficiency. To achieve 20 fps speed, we set $k = 0.12$ and $m = 4$. The proposed double network approach provides 5.7 times speed increasing.

D. *Training aspects*

Recently, as a default loss function cross entropy is commonly used in image segmentation tasks while training:

$$bce(y, \hat{y}) = -[ylog(\hat{y}) + (1 - y)log(1 - \hat{y})] \quad (3)$$

where y – ground truth mask and $\hat{y}$ - predicted mask.

However, DS2 is unbalanced. To reduce the influence of this factor, OHEM (Online Hard Example Mining) [31] binary cross entropy loss function was used.

$$ohemce\_loss(y, \hat{y}) = top_k(bce(y, \hat{y})) \quad (4)$$

where top_k – returns K maximal of cross entropy values. Parameter K was set to $B \times W \times H/16$, where B – batch size, W and H – width and height of an image.

A deep supervision approach and Adam optimizer with 5e-3 learning rate were used during the first training step. Next training step was performed with stochastic gradient descend (SGD) with cyclic learning rate scheduler [32] and modified IoU loss function:

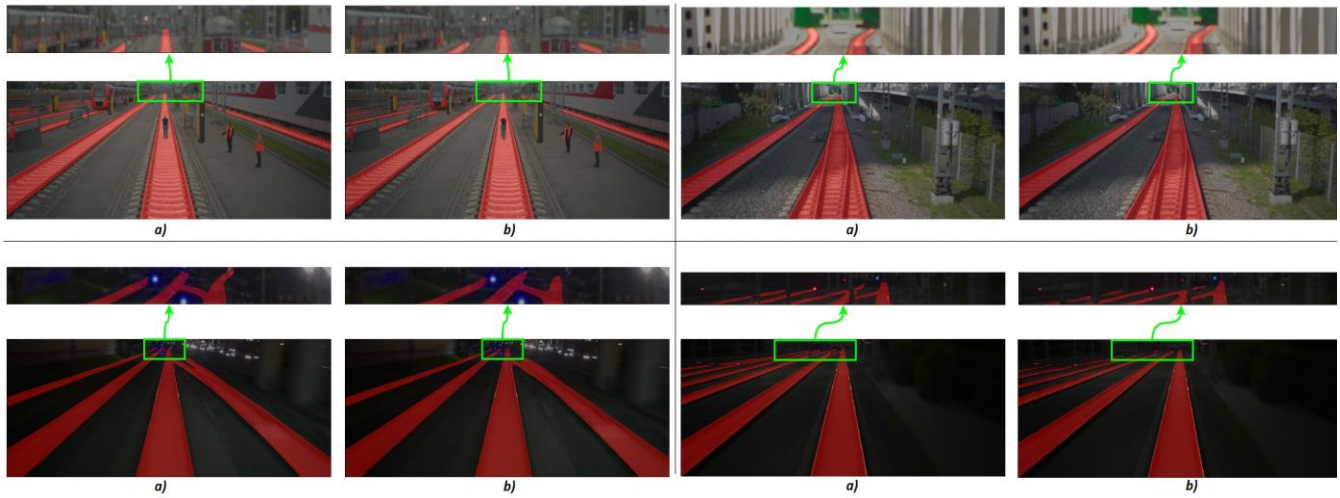

Figure 5. Example results of RailCNN. Top image strip represents A-region output (cropped for the sake of convenience), bottom image represents B-region output. (a) - ground truth, (b) - predicted mask.

$$wIoU = \frac{\sum w(Y)\, y\hat{y}}{\sum w(Y)(y + \hat{y} - y\hat{y})} \tag{5}$$

where weights $w(Y)$ are defined as

$$w(Y) = \frac{1}{1 + G - Y} \tag{6}$$

G – horizon level estimation in pixels, Y – vertical axis of an image (*Fig. 4*). The described modification was proposed to copy with a foreshortening effect in the distance. Thus, the pixels placed in a distance (higher on the image canvas) make a more significant contribution to the loss function, then closer ones.

E. Possible tracks detection

In this section, we propose an algorithm, which detects possible tracks among all ones obtained by the segmentation stage.

The algorithm starts with processing the lowest image row that contains mask pixels. Among all the line segments with the nonnull mask, the closest to the image center is chosen. The segment borders are meant to be the starting points of the possible track border. The algorithm observes all the borders points, moving from the bottom to the top.

The next border point p is computed in two stages. First, we estimate the next possible point p^* by linear extrapolation of the already computed border points.

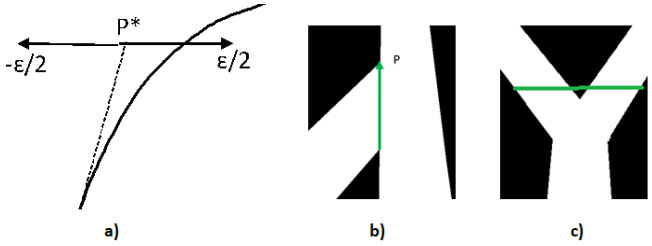

Figure 6. (a) – the next border point estimation; (b) – the track is interrupted by the incoming adjusted track from the left side; (c) – intersection detection

On the next stage, the result is elaborated by searching the real ϵ −neighborhood of the estimation p^* (*Fig. 6a*).

On the tracks crossing (*Fig. 6b*) point p wouldn't be found in the ϵ −neighborhood of the estimation p^*. In this case, the algorithm proceeds moving in the extrapolation line until an intersection is found. A track length (in pixels) in the extrapolation line is bounded by value 1 + c(G-Y), where the constant c was experimentally determined to be 0.21. If intersection was not found on the segment, the border tracing breaks.

Left and right branches are treated alongside. For every new pair of border points, railways intersection checking is made (*Fig. 6c*). If the intersection was found, the algorithm is evaluated recursively for every branch.

IV. EXPERIMENTS

A. Results

To evaluate the prediction accuracy of the proposed method, we refer to the metric from common semantic segmentation evaluation, such as IoU. Additionally, its modification wIoU is used, proposed in section 3.4. The wIoU penalizes heavier the misclassified pixels located closer to the G-level. Thus, providing more concrete information about segmentation accuracy on a long distance, compared to the vanilla IoU metric.

Table 2 reports the obtained results. The first two columns represent all rail tracks segmentation evaluation metrics. The next two columns: only possible tracks segmentation metrics.

All the evaluations are provided on NVIDIA GTX1080 Ti GPU.

TABLE II. EVALUATION RESULTS

All tracks		**Possible tracks**		**fps**
IoU	**wIoU**	**IoU**	**wIoU**	
0.944	0.912	0.983	0.955	20

B. Results comparison

Additional experiments were held to reveal the best dataset labeling approach to obtain data for possible tracks segmentation problem. The ground truth mask pixels not arranged to the possible tracks were discarded; thus, a dataset of possible tracks was obtained (using the algorithm proposed in section 3F). Our framework was trained and evaluated on such data. Table 3 reports the obtained results. For the sake of convenience, evaluation results on all tracks were duplicated from Table 2. Obtained results of all tracks and possible tracks approaches report insignificant difference in case of IoU metric comparison, but in case of wIoU metric, all tracks labeling approach outperforms significantly possible tracks approach. Thus, better railway segmentation on a long-range is achieved through all tracks labeling.

TABLE III. VARIOUS LABELING APPROACHES EVALUATION RESULTS COMPARISON

IoU		**wIoU**	
All tracks	***Possible tracks***	***All tracks***	***Possible tracks***
0.944	0.941	0.912	0.879

Table 4 represents the experimental results of the comparison between RailCNN and RailNet. The metric IoU was calculated only over possible tracks. According to the provided experiments, RailCNN outperforms RailNet in performance and range covering, providing the same computational efficiency.

TABLE IV. RAILNET VS. RAILCNN COMPARISON

	Detection range	**IoU**	**fps**
Results in [1]	< 118	0.898	20
Ours	**> 150**	**0.983**	20

V. CONCLUSION

Automotive control systems of low-speed self-driving trains with an average speed of 70 km/h, should be able to detect obstacles within the distance of 150 meters. Thus, the railroad should be detected at least with the same range. High-resolution images should be processed because of the range requirement. Such condition refers to computing efficiency problem: to maintain inference speed of at least 20 fps. Proposed in this paper framework copies with this problem using two neural networks, which process separate parts of an image. Our approach provides reliable railway detection within the required distance and inference speed. Experimental results significantly outperform the approach proposed in [1] in the range covering and accuracy with the same computing efficiency. The possible tracks detection algorithm was proposed. It was exploited to obtain the data that were used in dataset labeling methods comparison. The experiments proved all visible tracks labeling approach to be preferable, comparing to only possible tracks labeling.

VI. FUTURE WORK

Our future work will focus on improving segmentation performance and speed using temporal relations between video frames. Also, we are planning to study a more precise effect of image resolution and camera parameters on the framework results.

REFERENCES

[1] Y. Wang, L. Wang, Y. Hu, and J. Qiu, "RailNet: A Segmentation Network for Railroad Detection," *in IEEE Access*, vol. 7, pp. 143772-143779, 2019.

[2] F. Kaleli and Y. Akgul, "Vision-based railroad track extraction using dynamic programming," *in Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, 2009.

[3] Z. Qi, Y. Tian, and Y. Shi, "Efficient railway tracks detection and turnouts recognition method using HOG features," *Neural Computing and Applications*, vol. 23, no. 1, pp. 245–254, 2013.

[4] A. Purica, B. Pesquet-Popescu, and F. Dufaux, "A railroad detection algorithm for infrastructure surveillance using enduring airborne systems," *in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2187-2191, 2017.

[5] M. Karakose, O. Yaman, and E. Akinm "Detection of Rail Switch Passages Through Image Processing on Railway Line and Use of Condition-Monitoring Approach", *ICAT*, 2016.

[6] C. Tastimur, M. Karakose, and E. Akin, "A Vision Based Condition Monitoring Approach for Rail Switch and Level Crossing using Hierarchical SVM in Railways", *International Journal of Applied Mathematics, Electronics and Computers*, 2016.

[7] M. Zwemer, D. Wouw, and E. Jaspers, "A vision-based approach for tramway rail extraction", *in Proceedings of SPIE, International Society for Optical Engineering*, 2015.

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *arXiv preprint arXiv:1505.04597*, 2015.

[9] 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.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *arXiv preprint arXiv:1406.4729*, 2014.

[11] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network," *arXiv preprint arXiv:1612.01105*, 2017.

[12] M. Oršić, I. Krešo, P. Bevandić, and S. Šegvić, "In defense of pretrained imagenet architectures for real-time semantic segmentation of road-driving images," *in CVPR*, 2019.

[13] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *in Proceedings of the European Conference on Computer Vision (ECCV)*, 801-818, 2018.

[14] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A.Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834-848, 2016.

[15] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[16] Y. Yuan and J. Wang, "Ocnet: Object context network for scene parsing," *arXiv preprint arXiv:1809.00916*, 2018.

[17] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "CCNet: Criss-cross attention for semantic segmentation," *arXiv preprint arXiv:1811.11721*, 2018.

[18] K. Xiang, K. Wang, and K. Yang, "A Comparative Study of High-Recall Real-Time Semantic Segmentation Based on Swift Factorized Network," *arXiv preprint arXiv:1907.11394*, 2019.

[19] H. Wu, J. Zhang, K. Huang, K. Liang, and Y. Yu, "FastFCN: Rethinking dilated convolution in the backbone for semantic segmentation," *arXiv preprint arXiv:1903.11816*, 2019.

[20] X. Zheng, L. Huan, H. Xiong, and J. Gong, "ELKPPNet: An Edge-aware Neural Network with Large Kernel Pyramid Pooling for

Learning Discriminative Features in Semantic Segmentation," *arXiv preprint arXiv:1906.1142*, 2019

[21] J. Fu, J. Liu, H. Tian, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," *arXiv preprint arXiv:1809.02983*, 2018.

[22] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. Loy, D. Lin, and J. Jia, "PSANet: Pointwise spatial attention network for scene parsing," *in European Conference on Computer Vision, Springer,* pp. 270–286, 2018.

[23] S. Woo, J. Park, J. Lee, and I. Kweon, "CBAM: Convolutional block attention module," *in Proceedings of the European Conference on Computer Vision (ECCV),* pp. 3–19, 2018.

[24] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018.

[25] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high resolution images," *in Proceedings of the European Conference on Computer Vision,* pp. 405-420, 2018.

[26] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," *arXiv preprint arXiv:1808.00897*, 2018.

[27] H. Li, P. Xiong, H. Fan, and J. Sun, "DFANet: Deep Feature Aggregation for Real-Time Semantic Segmentation," *arXiv preprint arXiv:1904.02216*, 2019.

[28] B. Chen, L. Chen, Y. Wei, Y. Zhu, Z. Huang, J. Xiong, T. Huang, W. Hwu, and H. Shi, "Spgnet: Semantic prediction guidance for scene parsing," *in ICCV*, 2019.

[29] X. Li, Z. Jie, W. Wang, C. Liu, J. Yang, X. Shen, Z. Lin, Q. Chen, S. Yan, and J. Feng, "FoveaNet: Perspective-aware Urban Scene Parsing," *arXiv preprint arXiv:1708.02421,* 2017.

[30] O. Zendel, M. Murschitz, M. Zeilinger, D. Steininger, S. Abbasi, and C. Beleznai, "RailSem19: A Dataset for Semantic Rail Scene Understanding," *in CVPR Workshops*, 2019.

[31] A. Shrivastava, A. Gupta, and R. Girshick, "Training Region-based Object Detectors with Online Hard Example Mining," *arXiv preprint arXiv:1604.03540*, 2016.

[32] L. Smith, "Cyclical Learning Rates for Training Neural Networks," *arXiv preprint arXiv:1506.01186*, 2015.