Enhancement of Semantic Segmentation with Edge Networks Using Wavelets and Adaptive Canny Thresholding

Kuan-Lin CHEN[a] and Jian-Jiun DING[a,1]
[a] *Graduate Institute of Communication Engineering, National Taiwan University, Taipei 10617, Taiwan*

Abstract. Semantic segmentation is a core technology in computer vision. It plays a pivotal role in various real-world applications, including street scene recognition and medical image analysis. Current approaches of semantic segmentation predominantly leverage deep learning techniques and employ pixel-level decision making. However, these methods often introduce distortions in edge regions, which compromises segmentation accuracy. To address this issue, this study presents an efficient network architecture designed to extract edge features from images and incorporate them as the auxiliary input. This is further integrated with state-of-the-art models for joint training. Experimental results show significant improvements in accuracy and edge handling for semantic segmentation.

Keywords. semantic segmentation, wavelet transform, deep learning, Canny algorithm, convolution neural network

1. Introduction

Semantic segmentation is often required in various domains, such as street scenes, biomedical imaging, and space imagery, to annotate key regions for subsequent processing. Similarly, in scene understanding, segmenting different objects is frequently necessary to analyze information about various entities in the image. A significant amount of research has been conducted in this field. Traditional algorithms include methods like watersheds [1], which segment grayscale images into multiple catchment basins and then further divide the image into distinct regions using specific algorithms. There has been significant research on this technique, including numerous traditional algorithms. For instance, the watershed algorithm segments grayscale images into multiple catchment basins and then uses additional algorithms to divide the image into different regions. Another example is graph-based image segmentation algorithms [2], where the image is represented abstractly using graph theory. Pixels with similar characteristics are grouped into the same category using clustering algorithms, and adaptive thresholds are applied to control the size and number of resulting super-pixels. While these algorithms can segment various objects in an image, they often face limitations or errors when it comes to accurately classifying different categories. With the popularization of deep learning, many networks have been proposed for precise

[1] Corresponding author, Graduate Institute of Communication Engineering, National Taiwan University, Taipei 10617, Taiwan; E-mail: jjding@ntu.edu.tw.

classification tasks. For example, Fully Convolutional Networks (FCN) [3] marked a significant breakthrough in semantic segmentation, laying the foundation for deep learning in this field and making substantial contributions. Additionally, several networks based on Convolutional Neural Networks (CNNs) have been developed, such as U-Net [4], which features a U-shaped architecture. In U-Net, feature maps from each down-sampling stage are used as auxiliary information for the corresponding up-sampling stages, allowing the network to leverage early-stage data for more accurate predictions. The DeepLab series [5] employs dilated convolutions to expand the receptive field of convolutional layers and incorporates conditional random fields (CRFs) to enhance boundary details. In addition, the Temporal Memory Attention Network (TMANet) [6], based on the self-attention mechanism, captures relationships between the current frame and previously stored information from prior frames. Another recently powerful architecture is PIDNet [7], which is based on a PID controller. It divides the input image into three branches to process different levels of detail and integrates features from these branches for enhanced decision-making. PIDNet not only delivers accurate results but also achieves high computational efficiency. Semantic segmentation is widely applied in scenarios such as surveillance, dashcams, and autonomous vehicles. However, to achieve real-time segmentation of various objects in the real world, overly complex networks are impractical. Consequently, research tends to focus on CNN-based networks, which, while possibly inferior to Vision Transformer (ViT) [8] and other transformer-based architectures [9] in performance, provide sufficient accuracy for most applications. Illumination and shadows can sometimes affect content judgment, such as causing imprecise segmentation due to shadow interference. Recent studies have considered these factors, and [10] proposed a new dataset along with the Segmented Refinement Removal Network as a preprocessing approach to remove shadows from scenes. This technique unifies the light source and filters out shadow effects, thereby reducing their impact on segmentation performance.

This study proposes an idea that enhances the input image by performing certain transformations to obtain feature maps. These features are derived from a hybrid network combining traditional algorithms and CNNs. Using these features as auxiliary inputs provides the network with additional information, enabling it to refine the output map and improve performance.

- **Wavelet-Based Edge Feature Extraction**: We propose combining Discrete Wavelet Transform (DWT) and Synchronized Wavelet Transform (SWT) with CNNs to extract edge and low-frequency features as auxiliary inputs to support the main network during training.
- **Adaptive Canny Thresholding**: Canny algorithm edge features, combined with other feature maps, serve as auxiliary data for the main network. The algorithm is adaptively modified via a lightweight CNN for adjustable feature extraction.
- **Efficient Network Integration**: These feature extraction networks are integrated as auxiliary inputs to the main network, and training is employed to further enhance network performance.

2. Network Architecture

2.1. 2D Discrete Wavelet Transform

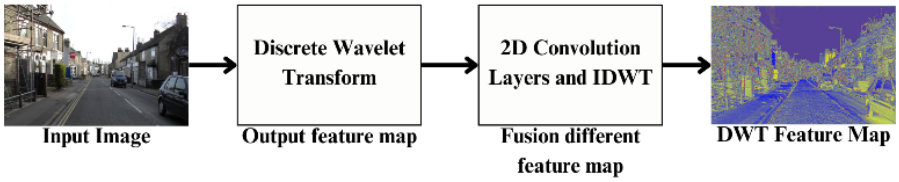

Figure 1. The 2D-DWT feature extractor

2D Discrete Wavelet Transform (2D-DWT) is a classical multi-scale decomposition algorithm designed for analyzing 2D signals such as images. It decomposes input images into multiple frequency component while retaining frequency information in the spatial domain, enabling precise extraction of high-frequency elements (e.g. edge information) at various scales. Specifically, 2D-DWT employs a pair of low-pass and high-pass filters to perform filtering along the horizontal and vertical directions, decomposing the signal X into four sub-bands: LL, LH, HL, and HH. To enhance computational efficiency, the decomposition process is reformulated into convolution operations, as expressed in equations Eq. (1) and Eq. (2), to quickly derive the four components. In this study, the 2D-DWT layer is implemented using the Wavelet Convolution proposed in [11]. By employing two levels of wavelet decomposition combined with trainable convolution layers, edge features are extracted from input images to effectively support training for semantic segmentation tasks.

$$f_{LL} = \frac{1}{2}\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}, f_{LH} = \frac{1}{2}\begin{bmatrix}1 & -1\\ 1 & -1\end{bmatrix}, f_{HL} = \frac{1}{2}\begin{bmatrix}1 & 1\\ -1 & -1\end{bmatrix}, f_{HH} = \frac{1}{2}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix} \quad (1)$$

$$[X_{LL}, X_{LH}, X_{HL}, X_{HH}] = [f_{LL}, f_{LH}, f_{HL}, f_{HH}] * X \quad (2)$$

The study utilizes two layers of 2D-DWT as a feature extraction branch. A schematic diagram of a single layer is shown in Figure 1. Initially, the input image is normalized and then processed through DWT to extract the original wavelet feature maps. These maps are subsequently passed through a CNN architecture and inverse DWT to reconstruct an output image with the same resolution as the input, serving as the feature map of the DWT layer.

2.2. 2D Synchronized Wavelet Transform

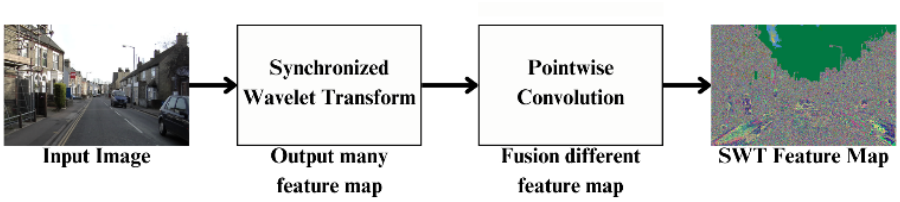

Figure 2. The 2D-SWT feature extractor

The 2D synchronized wavelet transform (2D-SWT) is another variant of wavelet transform. Unlike the DWT, the SWT processes signals in a translation-invariant manner, and it does not perform down-sampling like the DWT, allowing the resolution to remain the same as the input. The mathematical formulation is similar to the 2D DWT; however, in each layer, the coefficients of the high-pass and low-pass filters are processed using interpolation to ensure that the output resolution is unaffected by the transformation.

In this study, we apply the traditional SWT to process the input image X and use trainable pointwise convolutions to integrate information across different channels. This allows for scaling of the channels, reducing both the number of parameters and computational costs, while also combining cross-channel feature information. We define one layer as a combination of SWT followed by pointwise convolution, and the model uses two such layers to extract features. The architecture for this part is shown in Figure 2. First, the input image is normalized, then processed through SWT to extract frequency components in various directions. After that, pointwise convolution is applied to fuse the features between different components, and the output is a feature map with the same number of channels as the original image.

2.3. Self-Adaptive Canny Feature Extractor

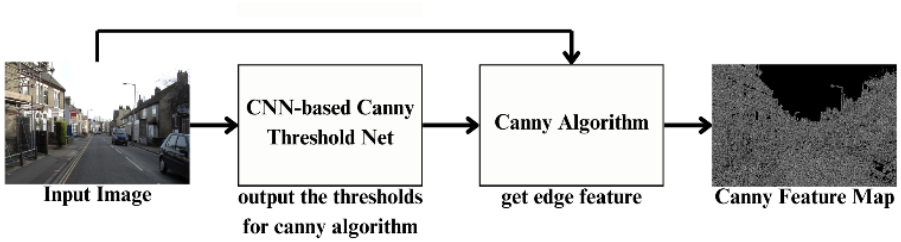

Figure 3. The Canny edge feature extractor.

The Canny algorithm is a traditional method used for edge feature extraction in images. It's capable of reliably detecting edges through a series of processing steps. First, the algorithm uses a Gaussian filter for noise reduction, then applies the Sobel operator to

compute the gradient of the image in both the horizontal and vertical directions. To precisely locate edges, non-maximum suppression is applied. If the gradient value at a given pixel isn't the local maximum in its gradient direction, it is set to zero. Next, two threshold values, T_{low} and T_{high}, are used to classify edges. If the gradient value higher then T_{high}, it is marked as a hard edge; if the gradient value is between T_{low} and T_{high}, it's marked as a weak edge; if the gradient value is less than T_{low}, it's considered non-edge. Finally, edge connectivity is performed. If a weak edge is connected to a hard edge, it is retained as part of the edge. If a weak edge does not connect to a hard edge, it is discarded as non-edge.

In this study, we parameterize the T_{low} and T_{high}, making them trainable parameters. Given that the Canny algorithm already has strong edge detection capabilities, we can adaptively adjust T_{low} and T_{high} for each input, thereby extracting the most effective edge information for training. The schematic diagram is shown in Figure 3. We designed a simple network consisting of two blocks of [2D Convolution, Batch Normalization, ReLU, followed by a flatten layer and a fully connected layer to output the values for T_{low} and T_{high}. The number of filters in the two CNN layers is 3 and 1, respectively, with a kernel size of (3, 3) and a stride of (4, 4). The output values of T_{low} and T_{high} are used as the two thresholds in the Canny algorithm, and the Canny feature map is computed based on these values.

2.4. Backbone Network

In this study, we use the current state-of-the-art PIDNet [7] to training, a model that employs three branches to process global structure (P branch), edge details (I branch), and spatial details (D branch), respectively. This model is primarily based on CNN architecture, rather than the more commonly used transformer architecture, which results in fewer parameters and faster inference, making it suitable for real-time semantic segmentation. Building on this foundation, we input the original image with a resolution of 960 × 720, along with the 2D-DWT, 2D-SWT, and Canny features of the original image, into this model and retrain it. After each epoch, we evaluate the model using the testing dataset. The model comes in different sizes, and in this study, we use the small architecture (PIDNet-S) for experiment and used the pretrained ImageNet [12] as the backbone. There are no significant changes to the network, except for the input layer, which is modified to accommodate the multiple input features by adjusting the number of input channels.

3. Network Architecture

3.1. Training Details

This study uses the CamVid dataset [13] as the training dataset, It contains 701 images, 367 images for training, 101 images for validation, and 233 images for testing. For the pixel-wise segmentation classification loss, we use Online Hard Example Mining (OHEM) cross-entropy. This loss is designed to address the imbalance in the number of labels commonly encountered in segmentation tasks. For example, the number of pixels belonging to a person and the number of background pixels usually differ significantly. OHEM calculates the loss by selecting labels with higher difficulty and ignoring simpler

ones, in order to reduce the negative impact of easy labels on training. In this study, following the original settings of PIDNet, the difficulty threshold for OHEM is set to 0.9, meaning the labels with the highest loss and difficulty are selected for training. The model is trained 200 epochs and using the SGD optimizer with a learning rate of 0.005, a batch size of 8, and a weight decay of 0.0005.

The calculation of OHEM typically starts by computing the loss for all samples, where *N* represents the total number of input image pixels, the loss for each pixel is computed and then sorted in descending order. After sorting, the pixel-wise losses are represented as shown in Eq. (3). And a sample set I_{OHEM} of size 64×64×32 is selected, consisting of all samples whose loss exceeds 0.9. If no loss exceed 0.9, the 64×64×32 samples with the highest losses are selected, the I_{OHEM} refers to the set where all L_i satisfy the given conditions, and these samples are used for loss computation Eq. (4).

$$L_1 \geq L_2 \geq \cdots \geq L_N \tag{3}$$

$$L_{OHEM} = \frac{1}{|I_{OHEM}|} \sum_{i \in I_{OHEM}} L_i \tag{4}$$

In addition to this, PIDNet also uses other components for loss calculation. These losses are computed using weighted binary cross-entropy loss to handle the imbalance of boundary pixels, which helps strengthen boundary features. Furthermore, boundary awareness cross-entropy loss is used to improve the performance of segmentation and boundary detection.

3.2. Hyper-parameter Setting

In this study, the hyperparameters for model training are fixed during the training process. For the experiment, each training consists of 120 epochs. The optimizer we used is SGD optimizer, with an initial learning rate set to 0.005, weight decay is 0.0005, and momentum is 0.9. The batch size is 6.

3.3. Experiment Result and Discussion

Table 1. Comparison of the proposed method and the original method on the CamVid dataset

Model Name	mIoU	Resolution	Training Parameters
DFANet A [14]	64.7	960×720	7.8M
LCTNet [15]	73.2	960×720	**1.4M**
PIDNet-S	73.8	960×720	<u>7.6M</u>
PP-LiteSeg-B [16]	75.0	960×720	-
DDRNet-23 [17]	<u>76.3</u>	960×720	-
Proposed Wavelet-EdgeNet	**77.4**	960×720	7.7M

This study conducted a series of tests on features generated by different algorithms, revealing that different features have varying impacts on the training process. Furthermore, many traditional algorithms were discarded due to prolonged computation times or because the features they generated did not effectively aid the training process.

Initially, we fully trained the original method and recorded the results. Subsequently, additional images with various features were incorporated as inputs to assist in training. The results show a noticeable improvement with the inclusion of these extra features, as demonstrated in Table 1.

After training, we also found that this method improves certain details. As shown in Figure 4, for some areas of the test dataset, we observed that using additional features to extract edge and detail information sometimes allows for more accurate judgment of smaller objects and ensures that the objects are properly connected. However, the overall performance is still quite similar to the original model.

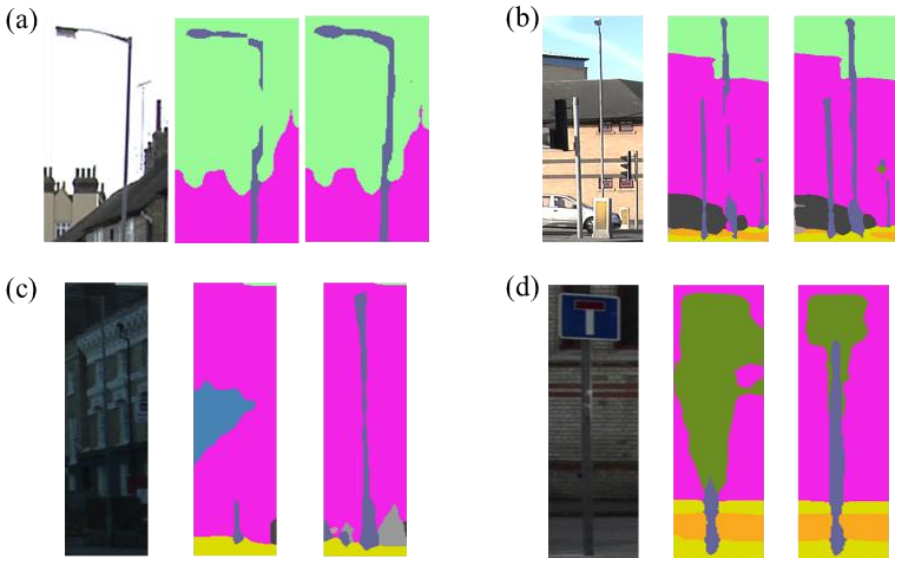

Figure 4. Comparative examples of segmentation results. The three subfigures within (a)(b)(c)(d) are (Left) the input image; (Middle): results with the original settings; (Right): results with the proposed Wavelet-EdgeNet method.

Further research is needed on how to extract more suitable features for model training and how to adjust the main model architecture in response to the increase in features.

After each epoch of training, an evaluation is performed on the testing data. As shown in Figure 5, the evaluation results indicate that our method converges quickly, typically reaching a better position in the early stages of training. We think this is because the input images, after being processed with some edge feature extraction methods, already have a preliminary segmentation, allowing the subsequent training to converge faster.

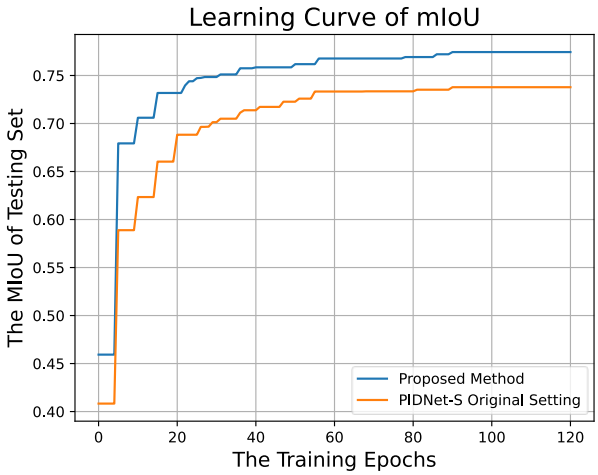

Figure 5. Learning curve of testing mIoU during training.

However, after testing, we found that this method is less compatible with simpler model architectures. Adding more features led to some difficulties during model training, resulting in poor performance. Additionally, when training on PIDNet, we discovered that adjusting the architecture for traditional algorithms is also quite challenging. First, it's necessary to ensure that these methods can perform backpropagation to compute gradients and optimize the model. Furthermore, we need to consider whether the features extracted by these methods will have a negative impact on the training process. These factors require extensive experimentation for validation. Our method currently shows preliminary results on this network, but it may not be compatible with other architectures. Future research should focus on developing more robust feature extraction methods and compatible architectures.

3.4. Ablation Study

Table 2. Ablation study of different feature.

Input Feature			Testing set mIoU	Training time (hr)
DWT	**SWT**	**Canny**		
✓			75.72	6.1
	✓		76.46	5.6
		✓	73.33	5.9
✓	✓		77.06	6.0
	✓	✓	<u>77.28</u>	**6.7**
✓		✓	76.20	5.8
✓	✓	✓	**77.44**	<u>6.2</u>

In this study, we analyzed the impact of various features on training performance, as shown in Table 2. All experiments were conducted on an RTX 3080. The results demonstrate that incorporating these features consistently improves training outcomes. Among individual feature trials, we observed that using only the SWT-derived features

yielded the most significant performance gains, while features derived solely from Adaptive Canny provided the least improvement, even underperforming compared to the baseline model without additional features. We think that in more complex street scenes or environments with insufficient lighting (due to poor weather or nighttime conditions), low-light environments make it challenging for Canny to provide accurate segmentation results. Even when using a CNN-based module to assist in determining Canny's two thresholds, the resulting feature maps may still introduce some negative interference to the training. However, through experimentation, we found that using all three features together yields the best results. While training time inevitably increases, there is a noticeable improvement in the final outcomes.

4. Conclusion

This study combines various edge extraction techniques and traditional image processing methods, such as wavelet transforms, with deep learning architectures as input features for model training. The experimental results reveal notable performance improvements over the baseline, particularly in the mIoU metric. Although the proposed method shows satisfactory compatibility with certain models, it still exhibits considerable limitations in some scenarios, such as when paired with simpler architectures or under complex lighting conditions, incorporating additional image features, such as edges, into the input of deep learning networks has proven to supply valuable data that can enhance model performance. However, the further experimentation is required to thoroughly understand the positive and negative impacts of these features on model training, as well as to enhance the generalizability and compatibility of this approach across diverse datasets and architectures.

Acknowledge

The authors thank for the support of National Science and Technology Council, Taiwan, under the contract of 113-2221-E-002-146.

References

[1] L.Vincent andP.Soille, “Watersheds in digital spaces: an efficient algorithm based on immersion simulations,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 6, pp. 583–598, Jun.1991, doi: 10.1109/34.87344.

[2] P. F.Felzenszwalb andD. P.Huttenlocher, “Efficient Graph-Based Image Segmentation,” *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, Sep.2004, doi: 10.1023/B:VISI.0000022288.19776.77.

[3] J.Long, E.Shelhamer, andT.Darrell, “Fully Convolutional Networks for Semantic Segmentation,” Nov.2014, [Online]. Available: http://arxiv.org/abs/1411.4038

[4] O.Ronneberger, P.Fischer, andT.Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” May2015, [Online]. Available: http://arxiv.org/abs/1505.04597

[5] L.-C.Chen, Y.Zhu, G.Papandreou, F.Schroff, andH.Adam, “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation,” Feb.2018, doi: arXiv:1802.02611.

[6] H.Wang, W.Wang, andJ.Liu, “Temporal Memory Attention for Video Semantic Segmentation,” Feb.2021, [Online]. Available: http://arxiv.org/abs/2102.08643

[7] J.Xu, Z.Xiong, andS. P.Bhattacharyya, “PIDNet: A Real-time Semantic Segmentation Network Inspired by PID Controllers,” Jun.2022, [Online]. Available: http://arxiv.org/abs/2206.02066

[8] A.Dosovitskiy *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” Oct.2020, [Online]. Available: http://arxiv.org/abs/2010.11929

[9] A.Vaswani *et al.*, “Attention Is All You Need,” Jun.2017, [Online]. Available: http://arxiv.org/abs/1706.03762

[10] C.-Y.Hsieh andJ.-J.Ding, “ADSP: Advanced Dataset for Shadow Processing, Enabling Visible Occluders via Synthesizing Strategy,” in *Computer Vision -- ACCV 2024*, M.Cho, I.Laptev, D.Tran, A.Yao, andH.Zha, Eds., Singapore: Springer Nature Singapore, 2025, pp. 329–347.

[11] S. E.Finder, R.Amoyal, E.Treister, andO.Freifeld, “Wavelet Convolutions for Large Receptive Fields,” Jul.2024, [Online]. Available: http://arxiv.org/abs/2407.05848

[12] O.Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, 2014, [Online]. Available: https://api.semanticscholar.org/CorpusID:2930547

[13] G. J.Brostow, J.Fauqueur, andR.Cipolla, “Semantic object classes in video: A high-definition ground truth database,” *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, Jan.2009, doi: 10.1016/j.patrec.2008.04.005.

[14] H.Li, P.Xiong, H.Fan, andJ.Sun, “DFANet: Deep Feature Aggregation for Real-Time Semantic Segmentation,” *2019 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pp. 9514–9523, 2019, [Online]. Available: https://api.semanticscholar.org/CorpusID:102354774

[15] Y.Shen andX.Ma, “Real-Time Semantic Segmentation Model Combining CNN and Transformer to Detect Belt Runout,” in *2024 5th International Conference on Machine Learning and Computer Application (ICMLCA)*, IEEE, Oct. 2024, pp. 451–456. doi: 10.1109/ICMLCA63499.2024.10754444.

[16] J.Peng *et al.*, “PP-LiteSeg: A Superior Real-Time Semantic Segmentation Model,” *ArXiv*, vol. abs/2204.0, 2022, [Online]. Available: https://api.semanticscholar.org/CorpusID:247996837

[17] Y.Hong, H.Pan, W.Sun, andY.Jia, “Deep Dual-resolution Networks for Real-time and Accurate Semantic Segmentation of Road Scenes,” *ArXiv*, vol. abs/2101.0, 2021, [Online]. Available: https://api.semanticscholar.org/CorpusID:231627652