ISSN: 2710-0588

Real Time Semantic Segmentation Algorithm based on Encoder-decoder

Shibo Hou[1, a], Wenwen Zong[2, *]

[1] Henan Polytechnic University, College of Computer Science and Technology, Jiaozuo 454000, China

[2] Shangqiu Institute of Technology, Shangqiu 476000, China

[a]2986681361@qq.com, [*]wenwen_zong@163.com

Abstract

Encoder-decoder networks have demonstrated significant advantages in semantic segmentation tasks. PP-LiteSeg as a light-weight encoder-decoder network, has gained a lot of attention because it can strike a good balance between accuracy and speed. However, the encoder and decoder in the networks have obvious limitations in extracting semantic information and recovering detailed information, respectively. To address these limitations, we carry out some improvements on PP-LiteSeg and propose MFF-LiteSeg. The improvements include a new encoder based on a parallel dilated convolution short-term dense cascade module and a multi-scale context module. The former module are exploited to help the encoder capturing rich features while maintaining low parameter counts. The latter module is integrated into the encoder to capture multi-scale contextual information, enhancing the ability of the encoder to understand objects of different scales. The improvements also include an attention fusion module which is applied to the decoder to optimize the feature fusion, enabling the decoder to more accurately restore detailed feature information. Experimental results on public datasets demonstrate that MFF-LiteSeg achieves higher segmentation accuracy than PP-LiteSeg while maintaining a fast inference speed of 160 FPS, striking a good balance between inference speed and segmentation accuracy.

Keywords

Real Time Semantic Segmentation, Convolutional Neural Net-Work, Lightweight Algorithm, Attention Fusion, Context Information.

1. Introduction

Semantic segmentation[1] is a fundamental task in computer vision, aiming to assign a semantic category label to each pixel in an image. Real-time semantic segmentation requires semantic segmentation models to maintain segmentation accuracy while improving inference speed to meet real-time demands. Therefore, real-time semantic segmentation often needs to strike a good balance among segmentation accuracy and inference speed. It has broad application prospects in fields such as autonomous driving[2], medical image analysis[3], and video surveillance[4].

Since Badrinarayanan et al.[5] first applied the encoder-decoder networks to semantic segmentation task and achieved significant results, a series of similar networks have been proposed[6][7][8]. However, these networks are only suitable for segmentation tasks prioritizing accuracy. They cannot meet the requirements of real-time tasks. To satisfy the requirements of real-time tasks, some light-weight encoder-decoder networks, e.g., E-Net[9], MSCFNet[10], PP-LiteSeg[11], RTFormer[12], and LETNet[13], have been proposed sequently. In these light-weight networks, PP-LiteSeg has gained a lot of attention because it can strike a good balance between

ISSN: 2710-0588

accuracy and speed. Nevertheless, PP-LiteSeg has obvious limitations in feature extraction and fusion. Specifically, the encoder struggles to capture rich semantic information especially in complex scenes. The decoder cannot effectively fuse the semantic features from encoders with the detail features from itself, which limits its ability to recover high-resolution features. These shortcomings negatively impact the performance of PP-LiteSeg.

To enhance the feature extraction capability of the encoder and the feature fusion capability of the decoder, we conducted a series of improvements on PP-LiteSeg, and proposed a new network named MFF-LiteSeg. In MFF-LiteSeg, the encoder is constructed based on a Short-Term Dense Concatenate Module with Parallel Dilated Convolution(STDC-PDC) and a Multi-Scale Context Module(MSCM). These two modules are proposed to enhance the feature extraction capability of the encoder, especially the capability for extracting multi-scale features. The decoder is improved by adopting an Attention Fusion Module(AFM) which is designed to enhance the feature fusion capability of the decoder, thereby improving its ability to restore high-resolution feature maps.

The contributions of this paper are as follows:

(1) We introduce a Short-Term Dense Concatenate Module with Parallel Dilated Convolution(STDC-PDC) and a Multi-Scale Context Module(MSCM) to enhance the feature extraction ability of the encoder.

(2) We design an Attention Fusion Module(AFM) to optimize the feature fusion between the decoder and encoder, thus improving the ability to recover high-resolution feature maps.

(3) We propose MFF-LiteSeg, a light-weight encoder-decoder network suitable for real-time semantic segment

2. Related Work

2.1. Encoder-decoder Structure Related Work

The encoder-decoder[14] structure extracts deep semantic and contextual information through convolution and downsampling of the encoder, and then the decoder gradually restores feature resolution through convolution and upsampling to achieve pixel-level prediction.

SegNet adopts a symmetric convolution upsampling structure and utilizes max-pooling indices to assist decoding. DeconvNet achieves end-to-end reconstruction through deconvolution layers, without relying on pooling indexes. ResSegNet introduces residual modules in the encoder section to enhance feature expression. To meet the growing demand for real-time semantic segmentation, various lightweight networks based on codecs have emerged: ENet significantly reduces computational load by simplifying decoders. RTFormer combines CNN and Transformer to balance local features and global context. MSCFNet and LETNet achieve a balance between accuracy and efficiency through multi-scale fusion and context enhancement. However, none of these networks can achieve excellent performance in both speed and accuracy simultaneously.

2.2. Heoretical Basis of PP-LiteSeg Network

PP-LiteSeg is a lightweight real-time semantic segmentation model based on an encoder decoder structure, which adopts an asymmetric architecture with lightweight encoder and efficient decoder, significantly reducing computational complexity while maintaining accuracy. The network consists of an encoder, a pyramid pooling module, and decoder.

The encoder consists of five stages: the first two stages reduce resolution through downsampling to reduce computation. The last three stages consist of Short-Term Dense Concatenate Modules (STDC), where STDC with a step size of 2 further downsamples and STDC with a step size of 1 captures semantic information. A lightweight pyramid pooling module is introduced at the end of the encoder to obtain multi-scale context. The decoder adopts a flexible

ISSN: 2710-0588

and lightweight structure (FLD), it utilizes a unified attention fusion module (UAFM) to fuse shallow and deep features, and gradually reducing channels during upsampling to improve efficiency.

However, PP-LiteSeg still has shortcomings: cascading downsampling in STDC may weaken deep semantic expression. The large-scale pooling of pyramid pooling module can easily result in the loss of local details. UAFM only uses a single attention mechanism, which makes it difficult to fully model the complex relationship between high-level semantics and low-level details.

3. MFF-LiteSeg

3.1. Architecture of MFF-LiteSeg

In order to improve the feature extraction capability of encoder and the feature fusion capability of decoder, we propose several improvements and construct MFF-LiteSeg by applying these improvements to PP-LiteSeg. These improvements include: improving the feature extraction capability by constructing encoder based on the proposed Short-Term Dense Concatenate Module with Parallel Dilated Convolution (STDC-PDC) and the proposed Multi-Scale Context Module (MSCM), and optimizing decoder by applying the proposed Attention Fusion Module (AFM). Figure 1 shows the structure of MFF-LiteSeg.

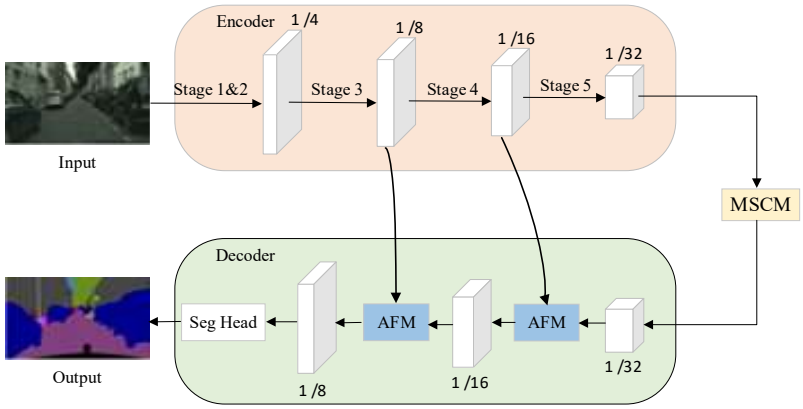

Figure 1. Structure of MFF-LiteSeg

3.2. Short-Term Dense Concatenate Module with Parallel Dilated Convolution (STDC-PDC)

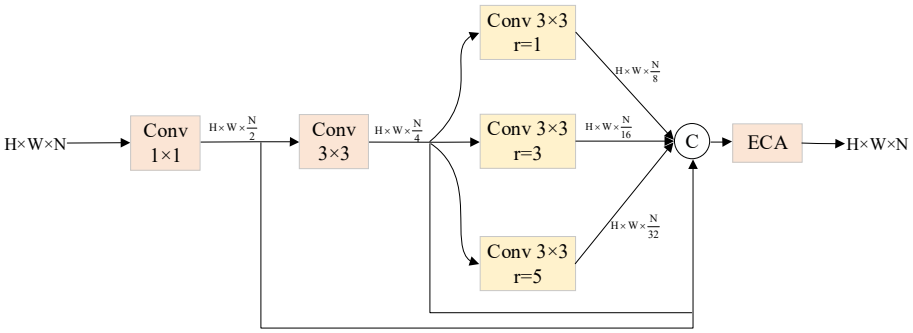

Figure 2. Structure of STDC-PDC

In PP-LiteSeg, the encoder is composed of STDC modules. However, STDC creates more feature maps in shallow layers while fewer feature maps in deep layers. The number of feature maps at the deepest layer is only 1/8 of the number of feature maps at the shallowest layer. This situation limits encoder's ability to extract deep semantic information. Furthermore, the STDC module only utilizes concatenation operations and 1x1 convolutions to fuse feature maps,

ISSN: 2710-0588

which fails to highlight the importance of different feature maps, thereby limiting the feature expressive capability of the models. To alleviate these issues, we improve STDC by introducing parallel dilated convolution and attention mechanism, and name the improved version as STDC-PDC. Figure 2 shows the structure of STDC-PDC.

We improve STDC in two aspects. First, we replace the last two convolutions in the STDC with a parallel dilated convolution block to extract rich features. The parallel dilated convolution block consists of three parallel dilated convolution branches, with dilation rates of 1, 3, and 5, respectively. Second, we exploit the Efficient Channel Attention module (ECA) to process the concatenated feature maps, highlighting the importance of different feature maps during the feature fusion process and enhancing the effectiveness of feature map fusion.

Given the input X, STDC-PDC first reduces the dimensionality of X with a 1x1 convolution operation, decreasing the number of channels to N/2, where N is the number of channels in X. Then, STDC-PDC extracts features with a 3x3 convolution and further reduces the number of the channels to N/4. Subsequently, STDC-PDC utilizes the parallel dilated convolution block to obtain feature maps with three different scales. Finally, STDC-PDC concatenates the output of all the convolutions, and processes the concatenated results with the ECA module.

3.3. Multi-Scale Context Module (MSCM)

Pyramid Pooling Module (PPM)[15] is widely used in semantic segmentation models due to its prominent advantage in extracting multi-scale contextual information. PPM first employs multiple pooling operations separately with varying kernels and strides to extract multi-scale context features. It then performs upsampling on the pooling outputs using bilinear interpolation to generate feature maps with the same resolutions, followed by feature fusion.However, pooling operations can lead to the loss of details, while upsampling operations can introduce noise into the feature maps. These factors limit the performance of segmentation models. To address this issue, we propose MSCM. Figure 3 shows the structure of MSCM.

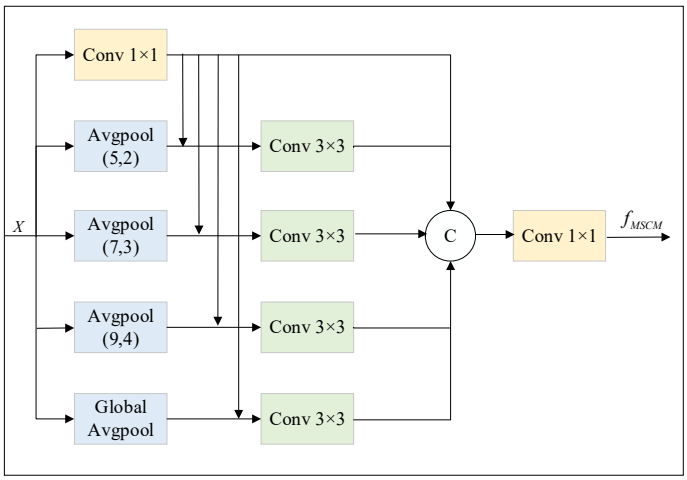

Figure 3. Structure of MSCM

MSCM consists of five branches. The first branch is composed of a 1x1 convolution, used for extracting local detail information from the input feature maps. The other four branches are parallel pooling branches, used for extracting context information of different scales. Among the four pooling branches, the first three branches are average pooling branches, with kernel sizes of 5x5, 7x7, and 9x9, padding values of 2, 3, and 4, respectively. Since the three branches output feature maps at the same scale, upsampling is not executed before feature fusion, thus avoiding the introduction of noise. The fourth pooling branch employs global average pooling to capture global context information. To further enrich the multi-scale context information

ISSN: 2710-0588

extracted by the pooling branches, MSCM fuses the results of the 1x1 convolution branch into the four pooling branches, respectively. Then, 3x3 convolutions are performed on the fused results for feature extraction. Finally, the feature maps from all five branches are concatenated along the channel dimension and fused via a 1x1 convolution, resulting in the output of MSCM. Given X as the input of MSCM, $f_{MSCM}(X)$ represents the output of MSCM. It is calculated as equation 1. In the equation, $Conv_{1\times1}$ represents 1×1 convolution, and $branch_i(\cdot)$ denotes the results of the ith branch. $branch_i(\cdot)$ is calculated according to equation 2. In the equation, $P_{2i+1}(\cdot)$ describes a 5×5 average pooling operation, $P_{global}(\cdot)$ notes the global average pooling operation.

$$\boldsymbol{f}_{MSCM}(X)=Conv_{1\times1}(Concat(branch_i(X))) \tag{1}$$

$$branch_i(X)=\begin{cases} Conv_{1\times1}(X) & i=1 \\ Conv_{3\times3}(P_{2i+1}(X)+branch_i(X)) & 1<i<n-1 \\ Conv_{3\times3}(P_{global}(X)+branch_i(X)) & i=n \end{cases} \tag{2}$$

3.4. Attention Fusion Module (AFM)

The encoder extracts deep semantic feature maps through multiple downsampling operations, while the decoder performs multiple upsampling operations on the deepest semantic feature maps to output high-resolution feature maps. To enrich the detail information in the output feature maps, the decoder progressively fuses the results of the upsampling with the encoder-generated feature maps of the same resolution. To highlight important contextual and detail information in the fused feature maps, we propose an Attention Fusion Module (AFM). Figure 4 shows the structure of AFM.

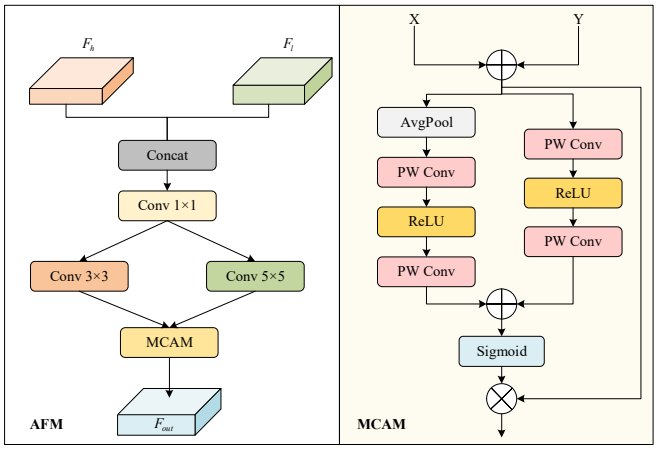

Figure 4. Structure of AFM

AFM first concatenates feature maps including rich detail features and those including context features along the channel dimension, and fuses the concatenated results with a 1 1 convolution. Subsequently, AFM exploits a 3 3 convolution and a 5 5 convolution to separately process the fused results to extract multi-scale features. Finally, AFM employs a proposed Multi-Scale Channel Attention Module (MCAM) to fuse the multi-scale features, highlighting the significant features in the outputs.

Given high-resolution feature maps- X_h and low-resolution feature maps- X_l, the output of AFM can be noted as $f_{AFM}(X_h, X_l)$ which is calcuated according to equation 3. In the equation, $Conv_{3\times3}(\cdot)$

ISSN: 2710-0588

and $Conv_{5\times5}(\cdot)$ seperately describe the results of 3 × 3 convolution and 5 × 5 convolution. $f_{fuse}(\cdot)$ and $f_{MCAM}(\cdot)$ represent the results of the fusion operation with a concatenation operation and a 1 × 1 convolution, and the results of MCAM, $f_{fuse}(\cdot)$ respectively. is calculated according to equations 4. In Equation 4, $Up(\cdot)$ denotes the upsampling operation, and $Concat(\cdot)$ denotes concatenation.

$$\boldsymbol{f}_{AFM}(X_h, X_l) = \boldsymbol{f}_{MCAM}(Conv_{3\times3}(f_{fuse}(X_h, X_l)), Conv_{5\times5}(f_{fuse}(X_h, X_l))) \tag{3}$$

$$\boldsymbol{f}_{fuse}(X_h, X_l) = Conv_{1\times1}(Concat(Up(X_l), X_h)) \tag{4}$$

MCAM first fuses the input feature maps using element-wise addition, and then utilizes two parallel branches to extract global features and local features from the fused results, respectively. The global branch consists of average pooling followed by two consecutive point-wise convolutions. The purpose of these two convolutions is to learn the relationships between different channels through cross-channel interaction. The local branch is composed of two consecutive point-wise convolutions. After that, MCAM fuses the results from the two parallel branches with element-wise addition followed by a sigmoid function to generate weights. Finally, the weights are multiplied with the fused feature maps to generate the output of MCAM. Given input feature maps X' and Y', the output of MCAM is denoted as $\boldsymbol{f}_{MCAM}(X', Y')$ which is calculated according to equation 5. In the equation, σ and $\oplus$ describe sigmoid function and element-wise addition, $G(X'+Y')$ and $L(X'+Y')$ separately represent the results of the Global branch and the local branch. $G(X'+Y')$ and $L(X'+Y')$ are calculated according to equations 6 and 7. In the two equations, $PWConv(\cdot)$ denotes point-wise convolution, and $AvgPool(\cdot)$ describes average pooling.

$$\boldsymbol{f}_{MCAM}(X', Y') = \sigma(L(X'+Y') \oplus G(X'+Y')) \otimes (X'+Y') \tag{5}$$

$$G(X'+Y') = PWConv(\delta(PWConv(AvgPool(X'+Y')))) \tag{6}$$

$$L(X'+Y') = PWConv(\delta(PWConv(X'+Y'))) \tag{7}$$

4. Experiments

In this section, we conduct experiments on two public benchmark datasets to evaluate our improvements. Firstly, we present the details of datasets, training, and evaluation metrics in Section 4.1. Secondly, we analyze the ablation experiments in Section 4.2. Finally, we discuss comparative experiments in Section 4.3.

4.1. Experimental Setting

Dataset.We exploit two public datasets to evaluate MFF-LiteSeg. The details of the datasets are described as follows:

Cityscapes is a high-resolution urban street scene dataset with 5,000 finely annotated images across 19 semantic segmentation categories. It is divided into 2,975 training, 500 validation, and 1,525 test images.

CamVid is another dataset for scene understanding with 701 annotated images, including 11 semantic categories, split into 367 training, 101 validation, and 233 test images.

ISSN: 2710-0588

Experiment. All the experiment is carried out on the same server which the server configuration is as follows: The processor adopts Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz, the memory capacity is 504GB, the graphics card uses NVIDIA GeForce RTX 4090, and the operating system is Ubuntu 20.04. Our improvements and all the comparative solutions are implemented in Python and run on PaddlePaddle framework.

During the experiment, we trained the network using a stochastic gradient descent optimizer (SGD) with an initial momentum of 0.9. We set a weight decay of for the Cityscapes dataset and for the CamVid dataset. The initial learning rate was set to 0.01 for Cityscapes and 0.001 for CamVid. The batch size was set to 16 for the Cityscapes dataset and 24 for the CamVid dataset, with the maximum number of iterations set to 160,000 for the Cityscapes dataset and 1,000 for the CamVid dataset. We applied multiple data augmentations, including random cropping, scaling, horizontal flipping, color jittering, and normalization, to both datasets.

Evaluation metrics. Mean Intersection over Union (mIoU) and Frames Per Second (FPS) are exploited to evaluate the accuracy and inference speed of all solutions, respectively. MIoU is calculated according to Equation 8 in which TP denotes true positives, FP represents false positives, and FN represents false negatives. Additionally, Params and GFLOPs are utilized to evaluate complexity.

$$mIoU = \frac{1}{k+1}\sum_{i=0}^{k}\frac{TP}{TP+FP+FN} \tag{8}$$

4.2. Ablation Experiments

To verify the effectiveness of the improvements applied in MFF-LiteSeg, we conducted a series of ablation experiments on Cityscapes dataset, and recorded the corresponding results in table 1.

In the table, Model0 represents the original version of PP-LiteSeg, Model1, Model2 and Model3 respectively represent replacing STDC in Model0 with STDC-PDC, applying MSCM to Model1, and replacing UAFM in Model2 with AFM to construct the improved models. Model1, Model2 and Model3 increased mIoU by 0.5%, 1%, and 0.6% compared to Model0, Model1, and model2, respectively. Model3, the model including all the improvements, improved mIoU by 2.1% compared to Model0. While improving segmentation accuracy, Model3 performed semantic segmentation tasks at 160 FPS, quite meeting the demands of real-time scenarios. At the same time, Model3 only increased 2.8M Params and 3.2 GFLOPs compared to Model0. These results fully verified the effectiveness of all the improvements in this work.

Table 1. Effectiveness of all the improvements.

Models	PA-STDC	MSCM	AFM	mIoU(%)	FPS	Params(M)	GFLOPs
Model0				75.3	195.3	11.7	19.3
Model1	√			75.8	190	12.3	20.22
Model2	√	√		76.8	173	13.8	21.1
Model3	√	√	√	**77.4**	**160**	14.5	22.5

4.3. Performance Comparison

We conducted a series of experiments on the Cityscapes dataset and CamVid dataset. In the experiment, we compared our solution, i.e., MFF-LiteSeg, with several lightweight semantic segmentation networks, i.e., BiSeNet1, BiSeNet2, BiSeNetV2, BiSeNetV2-L, STDC1-Seg75, STDC2-Seg75, PP-LiteSeg-T1 and PP-LiteSeg-B1, respectively.

ISSN: 2710-0588

Results on Cityscapes dataset: Table 2 shows the experiment results of all the networks on Cityscapes dataset. According to the table, MFF-LiteSeg achieved a mIoU of 77.4%, which is on average 4.15% higher than that of the series of BiSeNet, 1.65% higher than that of the series of STDC-Seg75, and 3.2% higher than that of the PP-LiteSeg series. MFF-LiteSeg also achieved a relatively high inference speed at 160 FPS. It outperformed the series of BiSeNet and the series of STDC-Seg75. Although it was slightly slower than the PP-LiteSeg series. It is still fast enough to meet real-time scene requirements. STDC2-Seg75 achieved a higher mIoU than that of each of the other networks except MFF-LiteSeg. The mIoU of STDC2-Seg75 is about 0.4\% lower than that of MFF-LiteSeg. Moreover, it ran semantic segmentation task at a slower inference speed, about 97FPS, compared with MFF-LiteSeg. STDC2-Seg75 also showed higher complexity in terms of Params (M) and GFLOPs. The above results show that MFF-LiteSeg achieves a better balance among speed, accuracy and complexity on Cityscapes dataset.

Table 2. Results on Cityscapes dataset

Methods	mIoU(%)	FPS	Params(M)	GFLOPs
BiSeNet1	69.0	105.8	5.8	14.8
BiSeNet2	74.8	65.5	14.6	55.3
BiSeNetV2	73.4	156	3.4	24.8
BiSeNetV2-L	75.8	47.3	—	118.5
STDC1-Seg75	74.5	126.7	11.4	38.1
STDC2-Seg75	77.0	97	15.4	53.0
PP-LiteSeg-T1	73.1	273.6	8.2	12.6
PP-LiteSeg-B1	75.3	195.3	11.7	19.3
OURS	**77.4**	**160**	14.5	22.5

Table 3. Comparative experiments on CamVid dataset

Methods	mIoU(%)	FPS
BiSeNet1	65.6	175
BiSeNet2	68.7	116.3
BiSeNetV2	72.4	124
BiSeNetV2-L	73.2	33
STDC1-Seg75	73.0	197.6
STDC2-Seg75	73.9	152.2
PP-LiteSeg-T1	73.3	222.3
PP-LiteSeg-B1	75.0	154.8
OURS	**76.7**	**130.7**

Results on CamVid dataset: Table 3 shows the results of all the networks on CamVid dataset. According to the table, MFF-LiteSeg achieved a mIoU of 76.7%, averagely 6.74% higher than that of the series of BiSeNet, 3.25% higher than that of the STDC-Seg75 series, and 2.55% higher

ISSN: 2710-0588

than that of the series of PP-LiteSeg. The running speed of MFF LiteSeg on the CamVid dataset is 130.7 FPS, which is lower than on the Cityscapes dataset. However, it is quite enough for real-time scenarios. PP-LiteSeg-B1 achieved the mIoU lower than MFF-LiteSeg but higher than the other solutions. However, PP-LiteSeg-B1 demonstrated higher inference speed and lower computational complexity compared to MFF-LiteSeg. It sacrifices accuracy for higher inference speed and lower complexity. Different from PP-LiteSeg-B1, MFF-LiteSeg moderately sacrifices inference speed and complexity for accuracy.

4.4. Analysis of Visualization Results

We selected 5 pictures featuring different scenes from Cityscapes dataset, and depicted the corresponding results processed separately by PP-LiteSeg and MFF-LiteSeg in figure 5. By comparing the results of PP-LiteSeg and those of MFF-LiteSeg with the corresponding labeled results, we found that MFF-LiteSeg achieved more complete and accurate segmentation of targets, especially for small ones. As indicated by the red rectangles in (d), MFF-LiteSeg can more completely capture the target boundaries, achieving higher-quality segmentation.The bottom image in (a) represents a dark scene in which the grass has a similar color to the road, a very small portion of the pixels of grass are misclassified by MFF-LiteSeg while a large portion of the pixels by PP-LiteSeg.These results fully demonstrate the accuracy and robustness of MFF-LiteSeg.

Figure 5. Comparison of segmentation results. (a), (b), (c) and (d) represent original images, labeled images, the results of PP-LiteSeg and those of MFF-LiteSeg, respectively.

5. Conclusion

In this work, we improved the encoder-decoder network, PP-LiteSeg, and proposed MFF-LiteSeg. In MFF-LiteSeg, we designed a short-term dense cascade module with parallel dilated convolution, and a multi-scale context module. We constructed the encoder of MFF-LiteSeg with both modules to enhance the feature extraction capability of the encoder. In addition, we designed an attention fusion module and applied it to the decoder to optimize the feature fusion between the encoder and the decoder. The experiment results on Cityscapes dataset and CamVid dataset demonstrated the effectiveness and efficiency of MFF-LiteSeg.

ISSN: 2710-0588

References

[1] Gao Y, Jiang Y, Peng Y, et al. Medical Image Segmentation: A Comprehensive Review of Deep Learning-Based Methods[J]. Tomography, 2025, 11(5): 52.

[2] Elhassan M A M, Zhou C, Khan A, et al. Real-time semantic segmentation for autonomous driving: A review of CNNs, Transformers, and Beyond[J]. Journal of King Saud University-Computer and Information Sciences, 2024, 36(10): 102226.

[3] Yang G, Wang Y, Shi D, et al. Golden Cudgel Network for Real-Time Semantic Segmentation[C]//Proceedings of the Computer Vision and Pattern Recognition Conference, 2025: 25367-25376.

[4] Li B, Tang X, Ruan C, et al. A Survey on Real-Time Semantic Segmentation Based on Deep Learning[C]//International Conference on Big Data and Security, 2023: 51-62.

[5] Badrinarayanan V, Kendall A, Cipolla R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(12): 2481-2495.

[6] Zhang Y, Yao T, Qiu Z, et al. Lightweight and progressively-scalable networks for semantic segmentation[J]. International Journal of Computer Vision, 2023, 131(8): 2153-2171.

[7] Noh H, Hong S, Han B. Learning deconvolution network for semantic segmentation[C]//Proceedings of the IEEE International Conference on Computer Vision, 2015: 1520-1528.

[8] Zhao H, Qi X, Shen X, et al. Icnet for real-time semantic segmentation on high-resolution images[C]//Proceedings of the European Conference on Computer Vision (ECCV), 2018: 405-420.

[9] Paszke A, Chaurasia A, Kim S, et al. Enet: A deep neural network architecture for real-time semantic segmentation[J]. arXiv preprint arXiv:1606.02147, 2016.

[10] Gao G, Xu G, Yu Y, et al. MSCFNet: A lightweight network with multi-scale context fusion for real-time semantic segmentation[J]. IEEE Transactions on Intelligent Transportation Systems, 2021, 23(12): 25489-25499.

[11] Peng J, Liu Y, Tang S, et al. PP-LiteSeg: A superior real-time semantic segmentation model[J]. arXiv preprint arXiv:2204.02681, 2022.

[12] Wang J, Gou C, Wu Q, et al. RTFormer: Efficient design for real-time semantic segmentation with transformer[J]. Advances in Neural Information Processing Systems, 2022, 35: 7423-7436.

[13] Xu G, Li J, Gao G, et al. Lightweight real-time semantic segmentation network with efficient transformer and CNN[J]. IEEE Transactions on Intelligent Transportation Systems, 2023, 24(12): 15897-15906.

[14] Chen S, Tang M, Dong R, et al. Encoder--Decoder Structure Fusing Depth Information for Outdoor Semantic Segmentation[J]. Applied Sciences, 2023, 13(17): 9924.

[15] Zhao H, Shi J, Qi X, et al. Pyramid scene parsing network[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017: 2881-2890.