Efficient Semantic Segmentation Using Multi-Path Decoder

Xing Bai $^{1,2,†}$ and Jun Zhou $^{1,2,*,*}$

$^1$ Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China; baixing@hccl.ioa.ac.cn

$^2$ University of Chinese Academy of Sciences, Beijing 100190, China

$^*$ Correspondence: zhoujun@hccl.ioa.ac.cn

† These authors contributed equally to this work.

Received: 8 July 2020; Accepted: 7 September 2020; Published: 14 September 2020

Abstract: Benefiting from the booming of deep learning, the state-of-the-art models achieved great progress. But they are huge in terms of parameters and floating point operations, which makes it hard to apply them to real-time applications. In this paper, we propose a novel deep neural network architecture, named MPDNet, for fast and efficient semantic segmentation under resource constraints. First, we use a light-weight classification model pretrained on ImageNet as the encoder. Second, we use a cost-effective upsampling datapath to restore prediction resolution and convert features for classification into features for segmentation. Finally, we propose to use a multi-path decoder to extract different types of features, which are not ideal to process inside only one convolutional neural network. The experimental results of our model outperform other models aiming at real-time semantic segmentation on Cityscapes. Based on our proposed MPDNet, we achieve 76.7% mean IoU on Cityscapes test set with only 118.84GFLOPs and achieves 37.6 Hz on 768 $\times$ 1536 images on a standard GPU.

Keywords: deep learning; semantic image segmentation; convolutional neural network

1. Introduction

The purpose of semantic segmentation is to predict the category label of each pixel in an image, which has always been a basic problem in computer vision. In recent years, with the deepening of research, the performance of semantic segmentation models has been greatly improved. This has also promoted the development of many practical applications such as autonomous driving [1], medical image analysis and virtual reality [2]. Following the fully convolutional network (FCN) [3], various architectures and mechanisms were introduced to capture contextual information and generate high-resolution representations. However, semantic segmentation models not only need to be accurate but also should be efficient in order to apply them to real-time applications.

To achieve high accuracy, state-of-the-art semantic segmentation models modified the downsampling layers in their backbones. This makes the feature maps output by backbones usually 1/8 of the original image size. Such models need huge time and GPU memory during training and inference. This makes it difficult to apply these models to scenes that require real-time segmentation. In order to make semantic segmentation more widely used, many real-time models [4–9] have been proposed. But results of models which are not pretrained on ImageNet are not satisfactory. Therefore, we choose a light-weight pretrained on ImageNet classification model as our backbone in order to realize real-time inference. But unlike the classification task, which only need to extract semantic information, models need to extract semantic, shape and location information for objects and stuff in the semantic segmentation task. Meanwhile, previous models process color, shape, location and texture information together.
inside only one convolutional neural network. Because they are very different kinds of information, this approach may not be optimal. Therefore, we propose a multi-path decoder. Different types of features are extracted by different branches in the decoder. Then, feature maps output by different branches are fused to generate segmentation results.

In recent years, some studies have used edge detection as auxiliary information to help intensify the prediction of pixels on the edge of things and stuff. In [10], Towaki et al. proposed to process shape information in a separate branch, resulting in a two-stream convolutional neural network (CNN) architecture for semantic image segmentation. [11] treated the edges as another semantic category to enable the network to be aware of the boundary layout. [12] extended the CNN-based agnostic edge detector proposed in [13] and allows each edge pixel to be associated with multiple categories. Although the shape branch helps to improve the accuracy of the semantic segmentation model, we think it is unnecessary to incorporate an edge detection branch to the semantic segmentation model during inference. Among them, we propose a dual-stream CNN structure for semantic segmentation. We explicitly links shape information to a single processing branch, namely the shape stream, but the edge detection task and the semantic segmentation task in our model are independent. We do not use the edge branch during inference, which reduces computational complexity of our model. As an additional semantic class, edge learning enables the network to understand the layout of boundaries.

In this paper, we present an effective lightweight architecture for semantic segmentation on datasets of few annotated images, based on ResNet features and multi-path decoder. The results of our proposed approach cannot match the state-of-the-art models, but smaller and faster than them. To our best knowledge, our model surpasses any other models aiming at real-time inference. By experimenting and evaluating on two public datasets, Cityscapes and ADE20K, we prove the effectiveness of our approach. Our MPDNet achieves 76.7% mean IoU (Intersection over Union) on the Cityscapes test set and 43.14% on ADE20K validation set with the single scale input.

The paper is structured as follows. In Section 2, we introduce two kinds of semantic segmentation models. In Section 3, we demonstrate our approach in detail. We give an overview of the whole architecture, elaborate the core module in our model and describe the computation process of the framework. In Section 4, the ablation experimental results and the comparison with other semantic segmentation methods are shown to prove the effectiveness of our approach. Finally, our conclusions are shown in Section 5.

2. Related Works

Image classification networks are used as the backbone in semantic segmentation task for extracting rich features. However, for yielding accurate segmentation, semantic segmentation models need semantic information and spatial information that backbone trained for classification cannot afford. Meanwhile, to segment objects with different scales and classify ambiguous pixels, multi-scale features are needed. To handle these problems, multiple modifications are made for pixel-level prediction. Most modifications focus on how to get multi-scale context information, which can be summarized to two kinds.

The first one is the modifications made for feature maps of same size but different receptive fields. Generally, models use feature maps with different receptive field sizes to represent different context information and then connect these feature maps to yield new feature maps which encode multi-scale context information. Many schemes are designed for combining feature maps of different receptive fields. Generally, many works first feed the last convolutional feature map of the backbone into a module that concatenates feature maps with different receptive fields, such as SPP(Spatial Pyramid Pooling) [14] or upgraded SPP, and then feed the feature maps output by this kind of module into the pixel-wise classifier [15]. DeepLab [16] use atrous convolution with different dilation rates to represent feature maps with different receptive fields. Combined with atrous convolution, DeepLab develops SPP to Atrous Spatial Pyramid Pooling (ASPP) module. By using different neurons to represent
sub-regions with different sizes, pyramid scene parsing network (PSPNet) develops SPP to Pyramid Pooling module.

The second one is modifications made for feature maps of different size. After copious operations of convolution with stride and pooling, the spatial size of feature maps put out by the backbone is very small compared to the original image and the receptive field of the neurons in the last layer is larger than the neurons in the shallower layers. By the same token, the neurons in the shallower layers have smaller receptive fields and the shallower layers encode less semantic information and more spatial information. This kind of models want to use feature maps in the shallower layers with high resolution to compensate the low resolution of high-level features and yield accurate segmentation results. To encode multi-scale context information, the collection of feature maps come very naturally. Specifically, when got feature maps from each layer, a “U-shape” architecture is built and feature maps from deep layer to shallow layer are gradually fused [17–20].

Besides, many works combine these two kind of modifications. Ref. [21] uses an “U-shape” architecture equipped with an atrous spatial pyramid pooling(ASPP) module. UPerNet [22] combines Feature Pyramid Network(FPN) with a pyramid pooling module (PPM) from PSPNet. Our proposed MPDNet also combines these two kind of modifications.

3. Method

The framework of our model is demonstrated in Figure 1, termed as MPDNet. All semantic segmentation methods face a contradiction between accuracy and speed.

![Figure 1](image)

**Figure 1.** An overview of the MPDNet. Feature maps of different sizes are extracted by the encoder. The main decoder has four identical paths to extract different types of features. Feature maps output by four paths in the main decoder are fused to generate segmentation results. The edge decoder learns edge information and generates boundary segmentation.

The backbone used in our framework is ResNet. There are four stages, Res1, Res2, Res3, Res4 in ResNet. The spatial resolution of feature maps put out by each stage of ResNet is 1/4, 1/8, 1/16, 1/32 of the original image. These feature maps are denoted as \( R_1, R_2, R_3, R_4 \), respectively. The feature maps from four different stages have different sizes. Many models use feature maps with different receptive field sizes to represent different context information and then connect these feature maps to yield new feature maps which encode multi-scale context information. The DeepLab series is one of the best. Their proposed ASPP module contains one \( 1 \times 1 \) convolutional layer and three \( 3 \times 3 \) convolutional layers with dilation rates of 6, 12, 18 respectively. The feature maps output by the encoder of DeepLab is 1/16 of the original image during training and 1/8 during inference. To save training and inference time, we do not follow their method. This makes that the height and width of feature maps output by the encoder of our model is 1/32 of the original image. Thus, we use feature maps in the shallower layers with high resolution to compensate the low resolution of high-level features and yield accurate segmentation results. To encode multi scale context information, we concatenate feature maps output by each block of the backbone.

At the same time, we found that under such circumstances, ASPP does not need to be as large as the original version in DeepLab to achieve good results. All the information is contained in the bottom feature maps in DeepLab (except DeepLabv3+). We do not need to be like this. Before inputting the
bottom feature map into each atrous convolution branch, we input the bottom feature maps into a 1 × 1 convolutional layer to reduce the number of channels to 512. This makes the number of FLOPs used in the modified ASPP in our model at least 77% lower than that of ASPP in DeepLab during inference. And when we use four ASPPs with 64 output channels instead of one with 256 output channels, the segmentation result is better and the model is smaller.

In semantic segmentation task aiming at real-time inference, a powerful and efficient upsampling datapath is very important. The most efficient upsampling datapath is to use a 1 × 1 convolutional layer as lateral connection and fuse enlarged feature maps with feature maps delivered by lateral connection. Then send the largest feature maps to a classifier. Efficient but powerless. Many previous “U-shape” models use several 3 × 3 convolutions as the lateral connection. Then upsample these feature maps to a quarter of the original image, concatenate them and reduce the channel dimension to the number of object categories. We find that the concatenation operation improve the segmentation results a lot. But the concatenation is operated on the largest feature maps, which requires many FLOPs. Therefore, we use a new lateral connection method.

In order to extract different type of features, we use four identical ASPP modules and four identical decoders for four ASPPs. In each decoder, we denote the feature maps output by the ASPP module as $L_C^i$. $L_C^i$ encodes the highest level of semantic information. Then the number of channels of $R_3$ is reduced to 64 by a lateral connection module which contains a $64 \times C \times 1 \times 1$ convolutional layer, one $64 \times 64 \times 3 \times 3$ convolutional layer with 64 groups and one $19 \times 64 \times 1 \times 1$ convolutional layer. By adding upsampled $L_C^4$ and reduced $R_3$, new feature maps are generated and are denoted as $L_C^3$. In the same way, we gradually enlarge feature maps from bottom to top and use lateral connections to fuse features encoded by shallower layers of ResNet. These feature maps are denoted as $L_C^3$, $L_C^2$, $L_C^1$, $L_C^4$, respectively. They have different sizes, gradually decreasing from $L_C^1$ to $L_C^4$ by a ratio of 2. To fuse them, we enlarge them up to the size of $L_C^1$ by bilinear interpolation and concatenate these resized feature maps. And then a convolutional layer is followed to reduce the channel dimension of these concatenated feature maps to the number of object categories. Then we concatenate feature maps output by four decoders and another convolutional layer reduce the channel dimension of these concatenated feature maps to the number of object categories. In order to supervise each path of the decoder, we calculate loss for feature maps output by each path of the decoder. Four classifiers are applied after four paths of the main decoder. The above process is summarized in Algorithm 1. Where $L_C^{ij}$ represents the $j$-th level feature maps in the $i$-th branch of the main decoder. $C_i$ represents the concatenation of four level feature maps in the $i$-th branch. $S_i$ represents the prediction of the $i$-th branch. $S$ represents the final prediction of our model.

Algorithm 1 Main Decoder in the Multi-Path Decoder Network.

**Input:** The image need to be segmented, Image;

**Output:** The segmentation result, $S$, the main loss, $L_m$, and the loss for each decoder, $L_{d_i}$.

1: $R_i = Encoder(Image)$, where $j \in [1, 4]$;
2: for each $i \in [1, 4]$ do
3: \hspace{1cm} $L_C^{ij} = ASPP(R_i)$;
4: \hspace{1cm} for each $j \in [1, 3]$ do
5: \hspace{1.5cm} $L_C^{ij} = RN(R_i) + Upsample(L_C^{ij+1})$;
6: \hspace{1cm} end for
7: \hspace{1cm} $C_i = Concat(L_C^{ij})$, where $j \in [1, 4]$
8: \hspace{1cm} $S_i = Conv(C_i)$
9: \hspace{1cm} $L_{d_i} = NLLLoss(S_i, Label)$
10: end for
11: $S = Conv(Concat(S_i))$, where $i \in [1, 4]$
12: $L_m = NLLLoss(S, Label)$
13: $L_d = \sum L_{d_i}$
Considering the importance of edge information in semantic segmentation, we use another decoder to learn edge information in addition to the main decoder. Like deep supervision proposed in PSPNet for their model, what the edge decoder learns is a supervision for our model. In the edge decoder, we also use ASPP module and gradually recover edge information by combining information encoded by the shallow layers. The steps of generating the predictions of edges in the edge decoder are the same as main decoder. The entire process for edge decoder is summarized in Algorithm 2.

Algorithm 2 Edge Decoder in the Multi-Path Decoder Network.

Input: Feature map output by the encoder, \( R_j \), where \( j \in [1, 4] \);
Output: The edge loss, \( L_e \).

1. \( LC_4 = \text{ASPP}(R_4) \);
2. for each \( j \in [1, 3] \) do
3. \( LC_j = \text{RN}(R_j) + \text{Upsample}(LC_{j+1}) \);
4. end for
5. \( C = \text{Concat}(LC_j) \), where \( j \in [1, 4] \)
6. \( S = \text{Conv}(C) \)
7. \( L_e = \text{NLLLoss}(S, \text{EdgeLabel}) \)

In our proposed model, we need to calculate three kind of losses in total, the main loss \( L_m \), the supervision loss for each decoder \( L_d \) and the edge supervision loss \( L_e \). Thus, the integrated loss function \( L_{\text{total}} \) is finally formulated as below:

\[
L_{\text{total}} = L_m + \alpha L_d + \beta L_e,
\]

where \( \alpha \) and \( \beta \) are weights for balancing the losses.

4. Results

All extra non-classifier convolutional layers have batch normalization [23]. ReLU (rectified linear unit) is applied after batch normalization [24]. Same as many works, we use the “poly” learning rate policy where the learning rate at current iteration equals to the initial learning rate multiplying \((1 - \frac{\text{iter}}{\text{max iter}})\)^{\text{power}}\) with \( \text{power} = 0.9 \). We set initial learning rate to 0.01 for Cityscapes, and 0.02 for ADE20K. Momentum and weight decay are set to 0.9 and 0.0001 respectively. To apply data augmentation, we adopt random resize between 0.4 and 1.5 and random cropping. And we also use other data augmentation schemes such as mean subtraction and horizontal flip. We set the batchsize to 16 during training.

The standard metrics used to evaluate semantic segmentation tasks include Pixel Accuracy (P.A.), which represents the proportion of correctly classified pixels, and mean IoU, which represents the intersection over union between the prediction and ground truth, averaged over all object categories. Generally, we use the mean IoU to evaluate the model on the Cityscapes dataset. We use the mean IoU and pixel accuracy to evaluate the model on the ADE20K dataset.

4.1. Cityscapes

Cityscapes segmentation dataset contains 19 foreground object categories and one background class. It contains 5000 high quality pixel-level finely annotated images collected from 50 cities in different seasons and is divided into 2975, 500, 1525 images for training, validation and testing. In addition, the dataset also includes 20,000 coarsely annotated images. Unlike the state-of-the-art models, we do not use coarse data in our experiments. The final results of our model are obtained by evaluating on 768 × 1536 images. Therefore, we set the cropsize to 512 during training. The number of training iteration is set to 40k.
Table 1 shows the results when we set the number of paths to different numbers. We find that when the decoder has four paths, the model works best. Table 2 shows the detailed results with different settings. Table 3 shows the per-class results on Cityscapes validation set. The lateral connection used in the baseline is a conventional $256 \times 3 \times 3$ convolutional layer. When we use ResNet-50 as the encoder and set cropszie to 768, we evaluate our model on $1024 \times 2048$ images and achieve 78.06\% mIoU under single scale test. ResNet-101 and multi scale test scheme promote the result to 79.99\% mIoU. But this is contrary to our original intention of real-time prediction. Therefore, in order to pursue efficiency, we set the cropszie to 512 and evaluate our model on $768 \times 1536$ images. When using ResNet-50 as the encoder, the baseline yields a result of 74.45\% in Mean IoU. While MPDNet yields a result of 76.84\% in Mean IoU, which brings 2.39\% improvement in terms of Mean IoU. In our experiments, we use 64 dilated convolutions in each branch of deeplab in the edge decoder. Because we find that when the size of the edge decoder is a quarter of the main decoder, the model works best. If the size of the edge decoder is larger, the edge detection task will have a negative effect on the semantic segmentation task. Experimental results with single-scale testing are listed in Table 4. Our model yields a result of 76.7\% on Cityscapes test set. Several examples are shown in Figure 2.

![Figure 2](image.png)

**Figure 2.** Visual improvements on Cityscapes val set. MPDNet produces more accurate and detailed results.

**Table 1.** Investigation of MPDNet with different number of paths on the validation set of Cityscapes.

| Encoder    | Number of Paths | Number of Channels in Each Path | mIoU(\%) |
|------------|----------------|-------------------------------|----------|
| ResNet-50  | 1              | 256                           | 76.25    |
| ResNet-50  | 2              | 128                           | 76.43    |
| ResNet-50  | 4              | 64                            | 76.84    |
| ResNet-50  | 8              | 32                            | 76.56    |
| ResNet-50  | 4              | 128                           | 76.99    |
| ResNet-50  | 8              | 64                            | 76.75    |
**Table 2.** Investigation of MPDNet with different settings on the validation set of Cityscapes. ‘RB’ denotes the lateral connection used in our model. The column ‘Resolution’ shows the image resolution on which the inference were performed. ‘MS’ denotes multi-scale testing. The column ‘FLOPs’ shows the number of floating point operations.

| Encoder   | Resolution | RB | Edge Decoder | Multi-Path Decoder | MS | mIoU(%) | FLOPs   |
|-----------|------------|----|--------------|-------------------|----|---------|---------|
| ResNet-50 | 768 x 1536 | ✓  |              |                    |    | 74.45   | 174.59G |
| ResNet-50 | 768 x 1536 | ✓  | ✓           |                    |    | 75.11   | 174.59G |
| ResNet-50 | 768 x 1536 | ✓  | ✓           | ✓                  |    | 76.25   | 116.1G  |
| ResNet-50 | 1024 x 2048| ✓  | ✓           | ✓                  |    | 76.84   | 118.84G |
| ResNet-101 | 1024 x 2048| ✓  | ✓           | ✓                  |    | 78.06   | 211.27G |
| ResNet-101 | 1024 x 2048| ✓  | ✓           | ✓                  | ✓  | 78.68   | 366.33G |
| ResNet-101 | 1024 x 2048| ✓  | ✓           | ✓                  | ✓  | 79.99   | -       |

**Table 3.** Per-class results on Cityscapes validation set. Training is done using the finely annotated train set. Scores are measured by %. ‘BL’ denotes baseline. ‘ED’ denotes edge decoder. ‘MPD’ denotes multi-path decoder.

| Method | Resolution | Val mIoU | Test mIoU | FLOPs | FPS |
|--------|------------|----------|-----------|-------|-----|
| BL     | 98.3       | 84.8     | 92.2      | 50.6  | 63.0| 68.6| 74.1| 92.2| 71.8| 94.3| 84.4| 66.2| 95.1| 60.4| 72.4| 59.5| 61.8| 72.4| 74.5|
| ERNet  | 98.3       | 84.2     | 92.0      | 53.4  | 53.5| 62.6| 68.1| 73.0| 92.4| 71.8| 94.6| 83.1| 65.6| 94.7| 65.0| 77.5| 69.0| 59.8| 71.2| 75.2|
| ESPNet | 98.4       | 84.2     | 92.1      | 57.0  | 55.6| 60.0| 60.5| 72.2| 92.8| 69.6| 95.2| 83.4| 65.4| 95.1| 70.7| 84.7| 82.0| 58.3| 71.5| 76.3|
| SwiftNet | 98.5     | 85.4     | 92.3      | 51.9  | 56.1| 61.2| 67.6| 74.4| 92.9| 71.2| 95.2| 83.9| 66.7| 95.4| 72.7| 83.0| 76.0| 62.8| 72.6| 76.8|

**Table 4.** Comparison to state-of-the-art on Cityscapes dataset. ‘-’ means that they are not reported in these papers. The column ‘Resolution’ shows the image resolution on which the inference were performed. The column ‘Val mIoU’ shows results on Cityscapes validation set. The column ‘Test mIoU’ shows results on Cityscapes test set. The column ‘FLOPs’ shows the number of floating point operations. The column ‘FPS’ shows the number of processed frames per second.

**4.2. ADE20K**

Unlike Cityscapes dataset, images in ADE20K dataset have different sizes. To prove the robustness of our model, we evaluate our model on the ADE20K validation set. ADE20K segmentation dataset contains 150 foreground object categories and is divided into 20,210, 2000, 3000 images for training, validation and testing. To evaluate our model, we conduct experiments with several settings. The cropsize is set to 480 for ADE20K. We set maximum iteration number to 125K. If inputting single-scale images during evaluation, our model yields a result of 43.14/80.91 on ADE20K val set with only 246.9 GFLOPs@1Mpx. If inputting multi-scale images during evaluation, the performance of our model can be promoted to 44.01/81.44. Experimental results are listed in Table 5. Table 6 shows the comparison with other models. Several examples are shown in Figure 3.
Table 5. Investigation of MPDNet with different settings on the validation set of ADE20K.

| Encoder    | RB | Edge Decoder | Multi-Path Decoder | MS | mIoU(%) | PA(%) |
|------------|----|--------------|---------------------|----|---------|-------|
| ResNet-50  | ✓  | ✓            | ✓                   |    | 38.88   | 78.34 |
| ResNet-50  | ✓  | ✓            | ✓                   |    | 39.56   | 79.65 |
| ResNet-50  | ✓  | ✓ ✓         | ✓                   |    | 40.98   | 79.93 |
| ResNet-50  | ✓  | ✓ ✓ ✓      | ✓                   |    | 42.06   | 80.42 |
| ResNet-101 | ✓  | ✓ ✓ ✓     | ✓                   |    | 43.14   | 80.91 |
| ResNet-101 | ✓  | ✓ ✓ ✓ ✓  | ✓                   |    | 44.01   | 81.44 |

Table 6. Comparison to state-of-the-art on the validation set of ADE20K. “-” represents that it is not reported in the related work.

| Method     | Single Scale Test mIoU(%)/PA(%) | Multi Scale Test mIoU(%)/PA(%) | SS FLOPs@1Mpx |
|------------|--------------------------------|-------------------------------|---------------|
| PSPNet     | 41.96/80.64                     | 43.29/81.39                   | 722G+         |
| UPerNet    | 42.00/80.79                     | 42.66/81.01                   | 370.74G       |
| DSSPN      | -                              | 43.68/81.13                   | -             |
| PSANet     | 42.75/80.71                     | 43.77/81.51                   | 722G+         |
| OCNet      | -                              | 45.45/-                       | 722G+         |
| MPDNet     | 43.14/80.91                     | 44.01/81.44                   | 246.9G        |

Figure 3. Visual improvements on ADE20K val set. MPDNet produces more accurate and detailed results.

5. Conclusions

In this work, we present a novel network, MPDNet, for fast and accurate semantic segmentation. The network has an encoder-decoder structure to encode the rich contextual information and recover the object boundaries. We use a powerful lateral connection to fuse the semantic information in deep layers with the spatial information in early layers. These connections help feature maps in deep layers with encoding abstract contextual information regardless of low-level details and small objects. Compared with various dilated architectures, this design considerably decreases computational complexity while achieving competitive results. Considering the difference between shape information and other information, we use two decoders to recover shape information and other information, respectively. The edge decoder affects the main decoder by adding the edge loss to the main loss.
There may be a more effective way to use the edge decoder to improve the main decoder. Meanwhile, we divide the main decoder to four paths to extract different type features. In each path, the enriched feature maps are concatenated to produce a high-resolution feature map. We supervise each branch of the main decoder by calculating loss for them. As an FPN-based model, MPDNet is relatively faster than many state-of-the-art models. Our experimental results show that our proposed model achieves good results on semantic segmentation benchmark Cityscapes (76.7%) with only 119GFLOPs.

Same as other semantic segmentation models, it requires a large number of finely annotated images to train our model. How to train an effective model with a small number of annotated images is still a problem to be solved. Due to the lack of high-resolution representations, it is still difficult for models to segment small objects, especially for models with large output stride. A model that can segment small objects well will greatly improve the benchmark. Meanwhile, we do not use information between frames in video segmentation. We look forward to a weakly supervised model for semantic video segmentation.

Author Contributions: All authors contribute equally. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the National Natural Science Foundation of China (Nos. 11590770, 61650202, 11590772, 11722437, 61671442) and IACAS Young Elite Researcher Project (No. Y854141431).

Conflicts of Interest: The authors declare there is no conflicts of interest regarding the publication of this paper.

References
1. Teichmann, M.; Weber, M.; Zoellner, M.; Cipolla, R.; Urtasun, R. MultiNet: Real-time joint semantic reasoning for autonomous driving. *arXiv* 2016, arXiv:1612.07695.
2. Hong, Z.W.; Yu-Ming, C.; Su, S.Y.; Shann, T.Y.; Chang, Y.H.; Yang, H.K.; Ho, H.L.; Tu, C.C.; Chang, Y.C.; Hsiao, T.C. Virtual-to-real: Learning to control in visual semantic segmentation. *arXiv* 2018, arXiv:1802.00285.
3. Long, J.; Shelhamer, E.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
4. Romera, E.; Alvarez, J.M.; Bergasa, L.M.; Arroyo, R. ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation. *IEEE Trans. Intell. Transp. Syst.* 2018, 19, 263–272. [CrossRef]
5. Zhao, H.; Qi, X.; Shen, X.; Shi, J.; Jia, J. ICNet for real-time semantic segmentation on high-resolution images. In *Computer Vision—ECCV*; Springer: Cham, Switzerland, 2018; pp. 418–434.
6. Mehta, S.; Rastegari, M.; Caspi, A.; Shapiro, L.G.; Hajishirzi, H. ESPNet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *Computer Vision—ECCV*; Springer: Cham, Switzerland, 2018; pp. 561–580.
7. Siam, M.; Gamal, M.; Abdel-Razek, M.; Yogamani, S.; Jagersand, M.; Zhang, H. A comparative study of real-time semantic segmentation for autonomous driving. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Salt Lake City, UT, USA, 18–22 June 2018; pp. 700–70010.
8. Orsic, M.; Kresco, I.; Bevandic, P.; Segvic, S. In defense of pre-trained ImageNet architectures for real-time semantic segmentation of road-driving images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019.
9. Kresco, I.; Krapac, J.; Segvic, S. Efficient ladder-style DenseNets for semantic segmentation of large images. *arXiv* 2019, arXiv:1905.05661.
10. Takikawa, T.; Acuna, D.; Jampani, V.; Fidler, S. Gated-scn: Gated shape CNNs for semantic segmentation. *arXiv* 2019, arXiv:1907.05740.
11. Ding, H.; Jiang, X.; Liu, A.Q.; Thalmann, N.M.; Wang, G. Boundary-aware feature propagation for scene segmentation. *arXiv* 2019, arXiv:1909.00179.
12. Yu, Z.; Feng, C.; Liu, M.Y.; Ramalingam, S. CASENet: Deep category-aware semantic edge detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 June 2017.
13. Xie, S.; Tu, Z. Holistically-nested edge detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1395–1403.
14. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Trans. Pattern Anal. Mach. Intell. 2015, 37, 1904–1916. [CrossRef] [PubMed]
15. Lazebnik, S.; Schmid, C.; Ponce, J. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), New York, NY, USA, 17–22 June 2006; Volume 2, pp. 2169–2178.
16. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. arXiv 2014, arXiv:1412.7062.
17. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Medical Image Computing and Computer-Assisted Intervention; Springer: Cham, Switzerland, 2015; pp. 234–241.
18. Gholami, G.; Fowlkes, C.C. Laplacian Pyramid Reconstruction and Refinement for Semantic Segmentation. In Computer Vision—ECCV 2016; Springer: Cham, Switzerland, 2016; pp. 519–534.
19. Islam, M.A.; Rochan, M.; Bruce, N.D.B.; Wang, Y. Gated Feedback Refinement Network for Dense Image Labeling. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4877–4885.
20. Peng, C.; Zhang, X.; Yu, G.; Luo, G.; Sun, J. Large Kernel Matters—Improve Semantic Segmentation by Global Convolutional Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1743–1751.
21. Chen, L.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 833–851.
22. Xiao, T.; Liu, Y.; Zhou, B.; Jiang, Y.; Sun, J. Unified Perceptual Parsing for Scene Understanding. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 432–448.
23. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 448–456.
24. Nair, V.; Hinton, G.E. Rectified Linear Units Improve Restricted Boltzmann Machines. In Proceedings of the International Conference on Machine Learning, Haifa, Israel, 21–24 June 2010; pp. 807–814.
25. Vallurupalli, N.; Annamaneni, S.; Varma, G.; Jawahar, C.V.; Mathew, M.; Nagori, S. Efficient semantic segmentation using gradual grouping. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 598–606.
26. Chaurasia, A.; Culurciello, E. LinkNet: Exploiting encoder representations for efficient semantic segmentation. In Proceedings of the 2017 IEEE Visual Communications and Image Processing (VCIP), St. Petersburg, FL, USA, 10–13 December 2017; pp. 1–4.
27. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid Scene Parsing Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6230–6239.
28. Chen, L.; Papandreou, G.; Schroff, F.; Adam, H. Rethinking Atrous Convolution for Semantic Image Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
29. Yu, C.; Wang, J.; Peng, C.; Gao, C.; Yu, G.; Sang, N. Learning a Discriminative Feature Network for Semantic Segmentation. arXiv 2018, arXiv:1804.09337.
30. Liang, X.; Zhou, H.; Xing, E. Dynamic-Structured Semantic Propagation Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018.
31. Zhao, H.; Zhang, Y.; Liu, S.; Shi, J.; Loy, C.C.; Lin, D.; Jia, J. PSANet: Point-wise Spatial Attention Network for Scene Parsing. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
32. Yuhui, Y.; Jingdong, W. Ocnet: Object context network for scene parsing. arXiv 2018, arXiv:1809.00916.