Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

SPSSNet: a real-time network for image semantic segmentation*

Saqib MAMOON[1], Muhammad Arslan MANZOOR[1], Fa-en ZHANG[2], Zakir ALI[1], Jian-feng LU‡[1]

[1]*School of Computer Science and Engineering, Nanjing University of Science & Technology, Nanjing 210094, China*

[2]*AInnovation, Beijing 100080, China*

E-mail: saqibmamoon@njust.edu.cn; arsalaan@njust.edu.cn; zhangfaen@ainnovation.com; alizakir@njust.edu.cn; lujf@njust.edu.cn

Received Dec. 14, 2019; Revision accepted Feb. 21, 2020; Crosschecked July 22, 2020

Abstract: Although deep neural networks (DNNs) have achieved great success in semantic segmentation tasks, it is still challenging for real-time applications. A large number of feature channels, parameters, and floating-point operations make the network sluggish and computationally heavy, which is not desirable for real-time tasks such as robotics and autonomous driving. Most approaches, however, usually sacrifice spatial resolution to achieve inference speed in real time, resulting in poor performance. In this paper, we propose a light-weight stage-pooling semantic segmentation network (SPSSN), which can efficiently reuse the paramount features from early layers at multiple stages, at different spatial resolutions. SPSSN takes input of full resolution 2048×1024 pixels, uses only 1.42 $\times 10^6$ parameters, yields 69.4% mIoU accuracy without pre-training, and obtains an inference speed of 59 frames/s on the Cityscapes dataset. SPSSN can run directly on mobile devices in real time, due to its light-weight architecture. To demonstrate the effectiveness of the proposed network, we compare our results with those of state-of-the-art networks.

Key words: Real-time semantic segmentation; Stage-pooling; Feature reuse
https://doi.org/10.1631/FITEE.1900697 **CLC number:** TP39

1 Introduction

Semantic segmentation is essential to comprehend an image because it can provide detailed information about all pixels in the image, thus known as dense image classification. In the past, convolutional neural network (CNN) architectures built for image classification have been applied directly to segmentation tasks (Cheng et al., 2018). Such architectures can achieve better results than traditional methods for semantic segmentation. Nevertheless, there still exist some challenges to be addressed, such as the uncertainty of appearance and location of objects, different shapes in images, lens flare distortion, and object occlusions. Conditional random fields (CRFs), Markov random fields (MRFs), Gaussian CRFs, and other variants have been proposed to solve the problems mentioned above. Although these methodologies can improve performance, they make the networks computationally complex.

A major breakthrough was the fully convolutional network (FCN) introduced by Long et al. (2014), which can be trained end to end and accept images of arbitrary sizes as input. Later, a vast number of deep neural network (DNN) architectures based on FCN were proposed (Christ et al., 2016; Dai et al., 2016a, 2016b; Sherrah, 2016; Lee et al., 2019), and demonstrated exceptional progress in this

‡ Corresponding author

* Project supported by the National Key R&D Program of China (No. 2017YFB1300205)

ORCID: Saqib MAMOON, https://orcid.org/0000-0002-8392-5118; Jian-feng LU, https://orcid.org/0000-0002-9190-507X

field; however, they are often implemented offline due to their low inference speed and high-dimensional feature vectors. In recent years, the application of DNNs in mobile systems has attracted more attention, and research increasingly focused on building light-weight, fast networks with an acceptable accuracy for segmentation in real time (Paszke et al., 2016; Mazzini, 2018; Poudel et al., 2018; Romera et al., 2018; Yu CQ et al., 2018; Zhao et al., 2018; Türkmen and Heikkilä, 2019). The most commonly used techniques to make a network fast and light include the following:

1. Network pruning

The strategy of eliminating redundancies in a pre-trained network can make the network more efficient, increase inference rates, and use fewer parameters and less memory (Han et al., 2016; Li H et al., 2016; Wen et al., 2016; Li C and Shi, 2018).

2. Weight quantization

Such methods encode the representation of trained weights in a few bits rather than high-precision floating points (Soudry et al., 2014; Hubara et al., 2016, 2018; Rastegari et al., 2016; Wu et al., 2018).

3. Convolutional factorization

This technique employs factorization of computationally expensive convolution operations to reduce computation requirements and memory consumption (Szegedy et al., 2015; Chollet, 2016; He et al., 2016; Howard et al., 2017; Huang et al., 2017; Ma et al., 2018; Mehta et al., 2018, 2019). To keep the model light-weight and fast, small feature maps and low-resolution images are often used, which can significantly reduce the accuracy. Additionally, in encoder-decoder architectures, the image goes through abundant down-sampling and up-sampling operations, thereby losing much of the finer image structure. Consequently, the critical information from early and intermediate layers is completely neglected, leading to the reduction in the overall performance of the network.

In this paper we propose a novel stage-pooling architecture to reuse the early and intermediate layer features at a high spatial resolution. Our model uses the bottom architecture of fast-SCNN (Poudel et al., 2019), which consists of the learning to downsample module, stage-pooling module, deep branch, shallow branch, and feature fusion module. We test our model on the Cityscapes (https://www.cityscapes-dataset.com/method-details/?submissionID=5369) standard benchmark with 2048×1024 pixels input, and the CamVid dataset with 720×720 pixels input. The proposed light-weight stage-pooling semantic segmentation network (SPSSN) yields 69.4% and 64.3% mean intersection-over-union (mIoU) with 59 and 105 frames/s on the Cityscapes and CamVid datasets, respectively.

Our main contributions are as follows:

1. A novel stage-pooling technique is introduced to reuse early and intermediate features for a light-weight model.

2. Our results show that compared with recent light-weight networks, our model achieves a better trade-off between accuracy and speed.

2 Related work

There is a lot of work on semantic segmentation; we will briefly review several popular architectures in this section. DNNs have shown impressive performance in image recognition (Peng et al., 2018), object detection (Hu et al., 2017), natural language processing (Devlin et al., 2018), and semantic segmentation; therefore, they have been widely used in feature generation and as encoders for segmentation tasks (Zhang QS and Zhu, 2018; Pan, 2019). Moreover, light-weight convolution approaches such as depth-wise separable convolutions, batch normalization, and inverted residual and linear bottleneck designs have paved way toward light-weight network architectures (Ioffe and Szegedy, 2015; Howard et al., 2017; Ren et al., 2017; Ma et al., 2018; Sandler et al., 2018; Mehta et al., 2019).

Since FCN-based models (Long et al., 2014) produce significant improvements in this field, researchers extended this concept using dilated convolutions and probabilistic graphical models (Liu et al., 2015; Zheng et al., 2015; Lin et al., 2016; Yu F et al., 2017). This concept has been adopted in many encoder-decoder architectures (Noh et al., 2015; Lin et al., 2019; Zhang and Peng, 2019b). In addition, recurrent neural networks (RNNs) have been used for object detection and segmentation tasks (Salvador et al., 2017; Zhang and Peng, 2019a). Visin et al. (2015) proposed ReSeg, an RNN-based architecture for semantic segmentation. The architecture comprises four RNNs which sweep the image both horizontally and vertically to retrieve the

contextual information. However, these approaches use deep feature generators to extract dense features and use sophisticated reconstruction methods to resize the images; consequently, the network becomes computationally expensive due to the large number of network parameters and floating-point operations (FLOPs) (Long et al., 2014; Noh et al., 2015; Chen et al., 2017, 2018; Hu et al., 2017; Yang et al., 2017; Zhao et al., 2017; Lin et al., 2019), which require more memory and power for heavy computation. Additionally, the low inference speed makes RNNs impractical for real-time applications (Siam et al., 2018).

Real-time applications aim to make models fit mobile/embedded devices and achieve high-quality segmentation. Badrinarayanan et al. (2017) proposed an end-to-end trainable semantic segmentation encoder-decoder architecture, SegNet, and used a small architecture and pooling indices to reduce the number of parameters. Paszke et al. (2016) introduced ENet, using dilated convolutions and performing several down-sampling operations to increase the inference speed. Zhao et al. (2018) proposed ICNet, using the cascade architecture, taking multi-scale images as input, and using label guidance and a fusion strategy to obtain segmentation results. Mehta et al. (2018) proposed ESPNet, decomposing pointwise convolutions and spatial pyramid of depth-wise dilated separable convolution modules to increase the receptive field, making the computation efficient. Li HC et al. (2019) proposed DFANet, employing multiple interconnected encoding streams to incorporate high-level context into the encoded features. Poudel et al. (2018) proposed ContextNet, a two-branch network architecture for feature extraction, with a deep branch to extract contextual features and a shallow high-resolution branch for global features, and fused them later in a decoder. Poudel et al. (2019) introduced fast-SCNN using the same feature extractor as ContextNet, and proposed the learning to downsample architecture, which shares the computation and weights among the first few layers of the deep and shallow branches for fast and efficient segmentation. However, we observed that in pursuit of keeping the model light-weight and fast, networks often lack the tendency to efficiently reuse the features from early to later layers. Unlike other approaches to increasing the efficacy of the network, we introduce stage-pooling to reuse the model's features extracted from early to later layers instead of making the network more complex and deeper.

3 Motivation

Real-time semantic segmentation models proposed recently are mostly multi-branch architectures, with each branch working at a different resolution. The deep branch consists of a series of convolution and max-pooling layers, which significantly reduces the spatial resolution. To refine and recover the spatial details of segmentation results, the shallow branch is employed. Note that early layers of DNNs extract low-level features, such as corners, edges/colors, and textures. Moreover, a receptive field of limited size is used in earlier layers, and feature pooling strategies such as spatial pyramid pooling (SPP) and atrous spatial pyramid pooling (ASPP) are often employed at the end of a network to make it more mobile-friendly. Consequently, maximum use of early and intermediate features at a high resolution is not considered. Therefore, we introduce a stage-pooling network that can efficiently extract the features from early layers at a high spatial resolution, resulting in sharp edges and corners, which is of utmost importance in semantic segmentation.

4 Network architecture

In this section, we describe the whole architecture of our model (Fig. 1). In general, SPSSN is an asymmetric encoder-decoder architecture. Our network uses the bottom architecture proposed by Poudel et al. (2019), which consists of one learning to downsample module, one deep branch, one shallow branch, four stage-pooling modules, one feature fusion module, and one classifier module. Stage-pooling modules serve as a bridge between deep and shallow layers at multiple stages, fusing the features of the deep layer at different feature and spatial dimensions into the shallow branch. The complete architecture of SPSSN is shown in Table 1.

4.1 Learning to downsample module

Learning to downsample is an initial module in our SPSSN, first introduced by Poudel et al. (2019); it takes the input image at full resolution and consists of three convolution layers. The first layer is a standard convolution layer (Conv2D), and the other

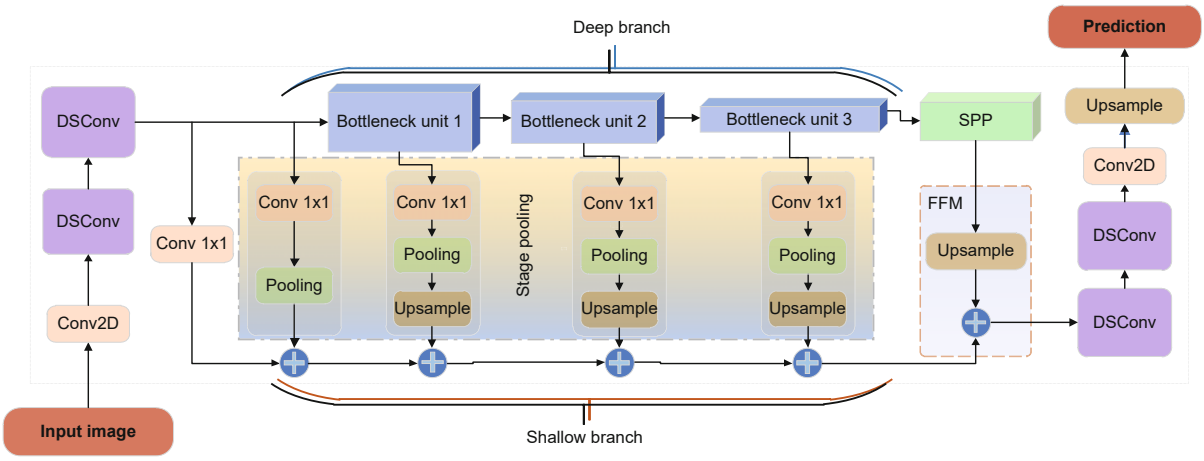

Fig. 1 Architecture of a stage-pooling semantic segmentation network (SPSSN): the overall network architecture contains four depth-wise separable convolutions (DSConv), three bottleneck units, four stage-pooling modules, one spatial pyramid pooling (SPP), pointwise convolutions, and one feature fusion module (FFM)

Table 1 Network architecture of SPSSN

Name	Input size	Output size	n
Conv2D	2048×1024×3	1024×512×32	1
DSConv	1024×512×32	512×256×48	1
DSConv	512×256×48	$256\times128\times64_{\mathrm{db,sb}}$	1
Stage-pooling 1	$256\times128\times64_{\mathrm{db}}$	$256\times128\times128_{\mathrm{sb}}$	1
Bottleneck	$256\times128\times64_{\mathrm{db}}$	$128\times64\times64_{\mathrm{db}}$	3
Stage-pooling 2	$128\times64\times64_{\mathrm{db}}$	$256\times128\times128_{\mathrm{sb}}$	1
Bottleneck	$128\times64\times64_{\mathrm{db}}$	$64\times32\times96_{\mathrm{db}}$	3
Stage-pooling 3	$64\times32\times96_{\mathrm{db}}$	$256\times128\times128_{\mathrm{sb}}$	1
Bottleneck	$64\times32\times96_{\mathrm{db}}$	$64\times32\times128_{\mathrm{db}}$	3
Stage-pooling 4	$64\times32\times128_{\mathrm{db}}$	$256\times128\times128_{\mathrm{sb}}$	1
SPP	$64\times32\times128_{\mathrm{db}}$	$64\times32\times128_{\mathrm{db}}$	1
FFM	$(64\times32\times128)^{4\times}_{\mathrm{db}}$ $+256\times128\times128_{\mathrm{sb}}$	256×128×128	1
DSConv	256×128×128	256×128×128	2
Conv2D	256×128×128	$256\times128\times c$	1

SPSSN consists of standard convolutions (Conv2D), depth-wise separable convolutions (DSConv), residual bottleneck convolutions (Bottleneck), stage-pooling modules, a spatial pyramid pooling (SPP) module, and a feature fusion module (FFM). n represents the number of blocks. c is the size of classifier. "db" and "sb" represent the deep branch and shallow branch, respectively. $^{4\times}$ means up-sampling four times

two layers are depth-wise separable convolution layers (DSConv). The depth-wise separable convolution is designed to use less computational resource than the standard convolution (Conv2D).

4.2 Deep branch

The deep branch consists of three bottleneck units (Fig. 1), and takes the direct input from the learning to downsample module at 1/8 resolution of the original input. Each bottleneck unit is composed of three bottleneck residual blocks proposed by MobileNetV2 (Sandler et al., 2018). We downsample only the spatial dimension in the first two bottleneck units, while the last bottleneck unit outputs the same resolution (Table 1). SPP is employed after the last bottleneck unit, with the original setting the same as used in Zhao et al. (2017) and Poudel et al. (2019).

4.3 Stage-pooling module

Lin et al. (2019) proposed RefineNet, in which a chained residual pooling (CRP) module was first introduced, which pools the features at multiple scales. Later, Nekrasov et al. (2018) proposed light-weight RefineNet (RefineNet-LW); they removed and replaced several convolution layers from CRP to make the network light-weight, but adopted the same multi-path architecture and complex network design. Light-weight chain residual pooling (CRP-LW) consists of two 5×5 maxpool operations and two pointwise 1×1 convolutions, and the summation of the residual connection and output of each convolution layer is shown in Fig. 2b. CRP-LW is referred to as the pooling module in our discussion. A stage-pooling module consists of one pointwise convolution layer, one pooling module, and one up-sampling layer (Fig. 2a). Each stage-pooling module takes the input from a bottleneck unit, except the first one which takes input directly from the learning to downsample module. Each bottleneck unit outputs different channel dimensions (i.e., 64, 96, 128) and

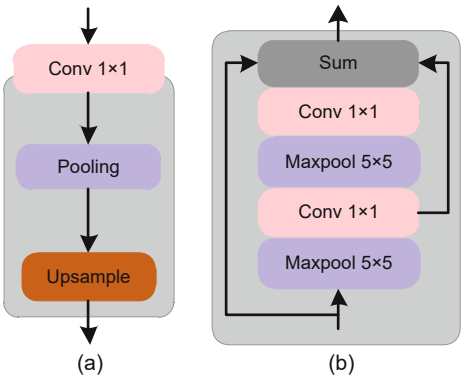

Fig. 2 Stage-pooling module with a pointwise convolution, a pooling module, and an up-sampling layer is shown in (a), and CRP as a pooling module used in (a) is shown in (b)

spatial dimensions (i.e., 1/16, 1/32, and 1/32) of the original resolution. To extract more feature details at each stage, we adopt the same 128-channel dimension in each stage-pooling module using pointwise convolution. Then, the pooling module pools the features at the same channel dimension but at different spatial dimensions. We then upsample the spatial resolution at 1/8 of every stage-pooling module using bilinear interpolation. We employ four stage-pooling modules as shown in Fig. 1. The purpose of the first two stage-pooling modules is to extract low-level features at a high spatial resolution, such as color texture/edge information, and we add them to the shallow branch, which operates at fixed feature and spatial dimensions. The third and fourth stage-pooling modules operate at different feature dimensions (i.e., 64 and 128) but at the same spatial dimension. This allows the stage-pooling modules to obtain important local, global, and contextual information, which is essential in understanding the whole scene.

4.4 Shallow branch

The shallow branch takes the input from the learning to downsample module after an adoptive pointwise convolution, which upsamples the feature map to the size of 128. The shallow branch can refine spatial and feature details simultaneously because it works at a high spatial resolution and fuses the output of each stage-pooling module. Although it is possible to use more advanced fusion modules, such modules will increase the computation cost.

4.5 Feature fusion and criterion

In the feature fusion module, we simply perform the element-wise addition of features from the deep and shallow branches similar to the methods in Poudel et al. (2018) and Zhao et al. (2018). The low- and high-level features are reciprocal to each other. That is, low-level features have fine spatial details but sparse semantic information, and high-level features have rich semantic information but coarse spatial details. The shallow branch has low-level high-resolution features, and the deep branch has high-level low-resolution features. Therefore, the fusion of shallow and deep branches outputs a feature map rich in both spatial and semantic information, which benefits the model for better segmentation results. The cross-entropy loss at each pixel over the categories is used as our criterion, because stochastic gradient descent (SGD) is used as an optimizer. It is evident that after feature fusion, even adding few layers will increase the accuracy (Poudel et al., 2019), so we keep two depth-wise separable convolution layers before the classifier.

5 Experiments

In this section, we will clarify the experimental details of our proposed model for training and testing. We evaluated and compared our results with those of baseline networks on the standard benchmarks of the Cityscapes (Cordts et al., 2016) and CamVid (Brostow et al., 2009) test sets.

5.1 Analysis of the SPSSN architecture

We adopted Cityscapes (Cordts et al., 2016) and CamVid (Brostow et al., 2009), two pixel-level semantic segmentation datasets, to evaluate our network.

1. Cityscapes dataset

Cityscapes is a large-scale urban scene dataset for semantic segmentation, consisting of images of street scenes from 50 different cities in Germany. It contains 5000 finely annotated images and 20 000 coarsely annotated images. The image resolution of the Cityscapes dataset is 2048×1024 pixels under a diversely changing environment, which makes real-time semantic segmentation a difficult task. Following the standard settings of Cityscapes, the finely annotated images were split into 2979, 500, and 1525

images for training, validation, and testing, respectively. We trained our model only on finely annotated images for 800 epochs. The model performance was evaluated on the Cityscapes validation and test sets. To be fair, we compared our results with those of several state-of-the-art models.

2. CamVid dataset

The Cambridge-driving labeled video database (CamVid) is a street scene dataset for semantic segmentation, with images captured from a camera mounted within a car. It consists of 701 manually annotated images, split into 367, 101, and 233 images for training, validation, and testing, respectively. The images have a maximum resolution of 960×720 pixels, recorded with a frame rate of 30 frames/s. Originally, the labels for 32 classes were provided. However, due to the rare occurrence of some classes, most literature focused on only 11. We used the same standard subset of 11 classes as used by Sturgess et al. (2009) for analysis of model efficiency. We used training and validation datasets to train our model, similar to Yu CQ et al. (2018) and Zhao et al. (2018).

5.2 Implementation details

We evaluated the performance of SPSSN on pixel-level semantic segmentation Cityscapes and CamVid datasets. We trained the model from scratch for both datasets using the mini-batch SGD optimization process with multi-class cross-entropy loss as the criterion, because it better generalizes the model on unseen data (Wilson et al., 2017). The momentum and weight-decay were set to 0.9 and 1×10^{-5}, respectively.

The poly-learning rate strategy was adopted as a common configuration, where the initial learning rate was multiplied by $\left(1-\dfrac{\text{epoc}}{\text{maxEpochs}}\right)^{0.9}$, where epoc corresponds to the current epoch and maxEpochs to the total number of epochs. The batch size and initial learning rate were selected according to the dataset. For Cityscapes, a batch size of four, crop size of 2048×1024, and initial learning rate of 2×10^{-2} were used. For CamVid, a batch size of eight, crop size of 720×720, and initial learning rate of 0.005 for the first 100 epochs and 0.0005 for the rest of training were used. To keep the model light-weight and fast with a high accuracy, we did not perform down-sampling in the first bottleneck unit because the crop size of 720×720 was used for CamVid.

Standard data augmentation techniques were employed, like random scaling (0.5, 2.0), horizontal flip, and random cropping from (0.5, 2) into a fixed size for training. All the experiments were carried out on a computer with Intel® Xeon® CPU E5-1620 v2@3.70 GHz, 78 GB memory, and a single NVIDIA TITAN Xp (Pascal) GPU, with CUDA 10.1 and CuDNN v7.3.

5.3 Evaluation metrics

We adopted primarily three evaluation metrics for performance measurement. The mean intersection-over-union (mIoU) of classes and categories as segmentation accuracy metric (Eq. (1)), the number of parameters as computation complexity metric, and the number of frames per second (FPS) as a speed metric. FPS is the inverse of time needed for the model to complete a single forward pass. FPS was measured on a single TITAN Xp GPU, unless stated differently. The number of iterations was set to 200 to avoid any accidental errors during speed evaluation. Similar to ENet (Paszke et al., 2016), we omitted all the batch normalization layers during inference and merged their parameters to the closest convolution layers.

$$\text{IoU}=\frac{\text{TP}}{\text{TP}+\text{FP}+\text{FN}}, \tag{1}$$

where the numbers of true positive, false positive, and false negative pixel-wise predictions are represented by TP, FP, and FN, respectively. The calculation of IoU is class-specific. The mean IoU (mIoU) can be obtained by calculating the mean value of IoU of all categories. mIoU is often used to measure the overall performance of models. An additional instance intersection-over-union metric (iIoU) was used for Cityscapes (Eq. (2)), assessing how well labeling represents the individual instances in the scene:

$$\text{iIoU}=\frac{\text{iTP}}{\text{iTP}+\text{FP}+\text{iFN}}, \tag{2}$$

where similar to standard IoU, iTP, FP, and iFN represent the numbers of true positive, false positive, and false negative predictions, respectively. However, unlike the standard IoU calculation, iTP and iFN are determined by measuring each pixel's contribution by the ratio of the average instance size of the

class to the size of the corresponding ground-truth case.

5.4 Evaluation details

In Tables 2 and 3, we compared our results on Cityscapes with those of the state-of-the-art baseline networks. The quantitative results showed that our model achieved a better trade-off between accuracy and efficiency, i.e., 69.4% mIoU with only 1.42×10^6 parameters without any pre-training on other datasets. By comparison with the state-of-the-art network ICNet (Zhao et al., 2018), SPSSN achieved only 0.1% less class (mIoU) accuracy, but ICNet exploited 7.8×10^6 parameters while our network used only 1.42×10^6, only about 1/5. The inference speed of ICNet was only 30 frames/s. However, our model's inference speed at a full resolution image of 2048×1024 pixels was 59 frames/s. Regarding efficiency, our network used 1/21 that of SegNet's parameters (Badrinarayanan et al., 2017) and yielded 12.4% performance gain. The efficient residual factorized network (ERFNet) proposed by Romera et al. (2018) employed 1D factorized convolution with dilation. It had about 1.5 times the number of parameters of SPSSN and achieved only 68% class (mIoU) accuracy with lower inference speed. ESPNet (Mehta et al., 2018) and ENet (Paszke et al., 2016) used fewer parameters but yielded poor segmentation results with 9.1% and 11.1% reduction in class (mIoU) accuracy and 3.8% and 5.6% drop of accuracy in category (mIoU), respectively. Nevertheless, ContextNet (Poudel et al., 2018) and Fast-SCNN (Poudel et al., 2019) employed a two-branch architecture and achieved only 82.7% and 84.7% accuracy in overall category segmentation, respectively, whereas SPSSN achieved 86% category (mIoU) accuracy. The individual class results showed the efficacy of our model over the above-mentioned state-of-the-art methods. ContextNet, Fast-SCNN, and our SPSSN had the same deep branch architecture, but SPSSN outperformed these two networks with a large margin in class and category miIoU. The results on the Cityscapes test set demonstrated that SPSSN achieved better results in 10 out of 19 classes compared with other models listed in Table 4, i.e., in road, sidewalk, building, traffic sign, vegetation, terrain, sky, person, rider, and car. Although ICNet had more categories (mIoU) than our model, SPSSN produced better results in 12 out of 19 category (mIoU) accuracies than ICNet.

Table 3 Comparison of single feed forward time and runtime between SPSSN and several baseline networks

Network	Time (ms)	FPS
SegNet (Badrinarayanan et al., 2017)	67	15
ThunderNet (Xiang et al., 2019)	10.1	96
ENet (Paszke et al., 2016)	13	77
ESPNet (Mehta et al., 2018)	18	54
ERFNet (Romera et al., 2018)	21	41.7
ICNet (Zhao et al., 2018)	33	30
Fast-SCNN (Poudel et al., 2019)	9.4	106.2
ContextNet (Poudel et al., 2018)	24.4	41.9
SPSSN (Ours)	16	59

FPS: number of frames per second

Table 2 Results of class and category in terms of mean intersection-over-union and mean instance intersection-over-union, and the number of parameters on the Cityscapes test set

Network	mIoU (%)		miIoU (%)		Number of parameters ($\times10^6$)
	Class	Category	Class	Category	
BiSeNet2 (Yu CQ et al., 2018)	74.7	–	–	–	49
SegNet (Badrinarayanan et al., 2017)	57.0	79.1	32.0	61.9	29.5
ENet (Paszke et al., 2016)	58.3	80.4	34.4	–	0.36
ESPNet (Mehta et al., 2018)	60.3	82.2	31.8	63.1	0.4
ThunderNet (Xiang et al., 2019)	64.0	84.1	40.4	69.3	4.7
ContextNet (Poudel et al., 2018)	66.1	82.7	36.8	64.3	0.85
ERFNet (Romera et al., 2018)	68.0	86.5	40.4	70.4	2.1
Fast-SCNN (Poudel et al., 2019)	68.0	84.7	37.9	63.5	1.11
BiSeNet1 (Yu CQ et al., 2018)	68.4	–	–	–	5.8
ICNet (Zhao et al., 2018)	69.5	86.4	–	–	7.8
SPSSN (Ours)	69.4	86.0	41.8	70.2	1.42

mIoU: mean intersection-over-union; miIoU: mean instance intersection-over-union

The evaluation results on the CamVid test set are shown in Table 5. Our model achieved 64.3% mIoU and 91.1% global average accuracy. ICNet used no compressed PSPNet50 as a base model for the CamVid dataset, and obtained high accuracy of 67.1% mIoU but with only 27.8 frames/s, while SPSSN achieved a high inference speed of 105 frames/s. Dilation8 achieved 1% better accuracy, but used 1.408×10^8 parameters with only 4.4 frames/s, making it impractical for real-time applications.

5.5 Visualization

The visual results on the Cityscapes validation set and CamVid test set are shown in Figs. 3 and 4, respectively. Stage-pooling modules operated at multiple levels helped the model classify close and distant objects. The classes that usually overlap each other are very difficult to segment, but SPSSN classified these classes with a high precision, such as a biker who sits on the motorbike, road, and sidewalk. Additionally, it made accurate predictions for

Table 4 Comparison of individual class results on the Cityscapes test set with baseline networks

Network	IoU (%)									
	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic sign	Vegetation	Terrain
SegNet	96.4	73.2	84.0	28.4	29.0	35.7	39.8	45.1	87.0	63.8
ENet	96.3	74.2	75.0	32.2	33.2	43.4	34.1	44.0	88.6	61.4
ESPNet	97.0	77.5	76.2	35.0	36.1	45.0	35.6	46.3	90.8	63.2
ERFNet	97.2	80.0	89.5	41.6	45.3	56.4	**60.5**	64.6	91.4	**68.7**
ThunderNet	97.2	77.3	88.3	41.1	38.3	48.5	55.6	60.8	90.66	67.7
ICNet	97.1	79.2	89.7	43.2	48.9	**61.5**	60.4	63.4	**91.5**	68.3
ContextNet	97.4	79.6	89.5	**44.1**	**49.8**	45.5	50.6	64.6	90.2	59.4
SPSSN	**97.7**	**80.8**	**89.8**	43.9	46.5	53.1	58.8	**64.7**	**91.5**	**68.7**

Network	IoU (%)								
	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle
SegNet	91.8	62.8	42.8	89.3	38.1	43.1	44.1	35.8	51.9
ENet	90.6	65.5	38.4	90.6	36.9	50.5	48.1	8.8	55.4
ESPNet	92.6	67.0	40.9	92.3	38.1	52.5	50.1	41.8	57.2
ERFNet	**94.2**	76.1	56.4	92.4	45.7	60.6	27.0	48.7	61.8
ThunderNet	92.9	71.3	46.6	91.6	39.31	49.9	49.8	45.5	62.3
ICNet	93.5	74.6	56.1	92.6	51.3	**72.7**	51.3	**53.6**	**70.5**
ContextNet	93.4	70.9	43.1	91.8	**65.2**	71.9	**64.5**	41.9	66.1
SPSSN	**94.2**	**76.2**	**59.0**	**92.7**	53.5	71.0	59.2	52.9	63.8

IoU: intersection-over-union. The best results are in bold

Table 5 Average class accuracy and global accuracy, mean intersection-over-union, the number of parameters, and the number of frames per second (FPS) on the CamVid test set

Network	mIoU (%)	Class avg. (%)	Global avg. (%)	Number of parameters ($\times10^6$)	FPS
ENet (Paszke et al., 2016)	51.3	68.3	–	0.36	61.2
SegNet (Badrinarayanan et al., 2017)	55.6	65.2	88.5	29.5	4.6
FCN-8s (Long et al., 2014)	57.0	–	88.0	134.5	–
FC-DenseNet56 (Jégou et al., 2017)	58.9	–	88.9	1.5	27
DeepLab-LFOV (Chen et al., 2018)	61.6	–	–	37.3	4.9
Dilation8 (Yu F and Koltun, 2016)	65.3	–	79.0	140.8	4.4
ESPNet (Mehta et al., 2018)	55.6	68.3	–	–	205
ERFNet (Romera et al., 2018)	53.1	65.8	86.3	–	–
ERFNet* (Romera et al., 2018)	62.7	72.2	89.4	–	–
ICNet (Zhao et al., 2018)	67.1	–	–	–	27.8
SPSSN (Ours)	64.3	73.55	91.1	1.4	105

mIoU: mean intersection-over-union; Class avg.: average class accuracy; Global avg.: average global accuracy; FPS: number of frames per second. * Pre-trained on ImageNet

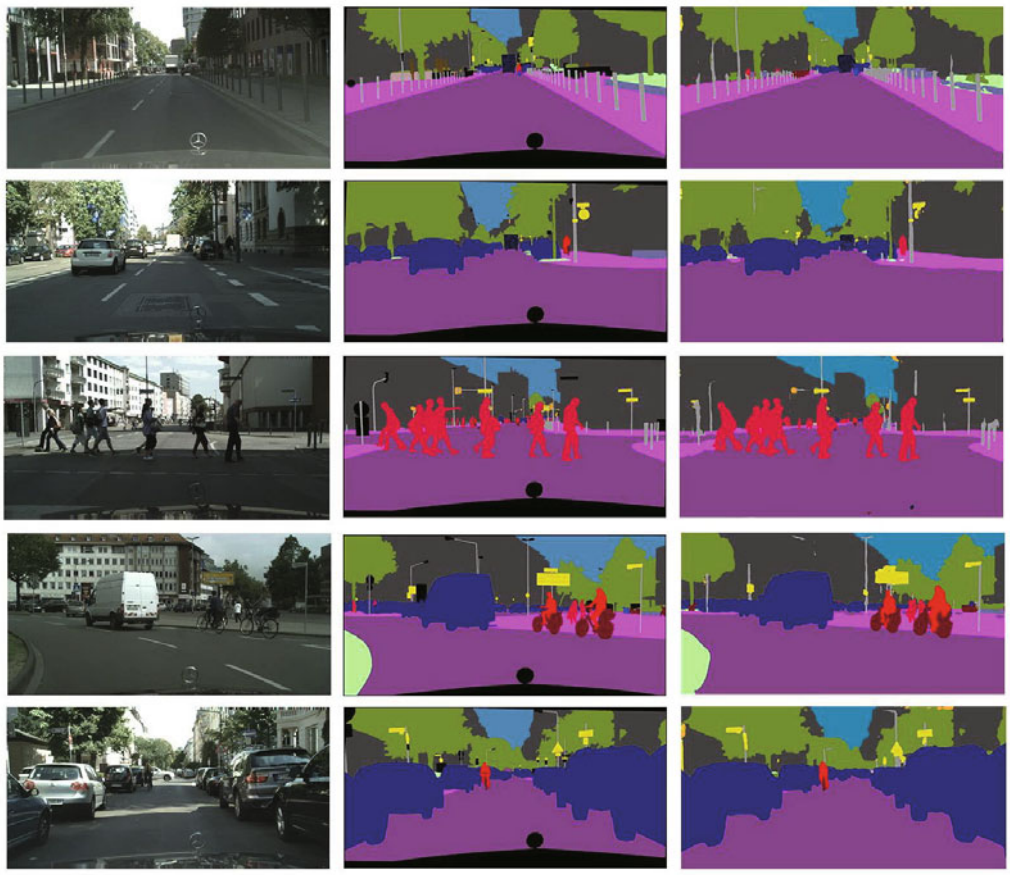

Fig. 3 Visual results on the Cityscapes validation set (left to right: original input image, ground truth, SPSSN results)

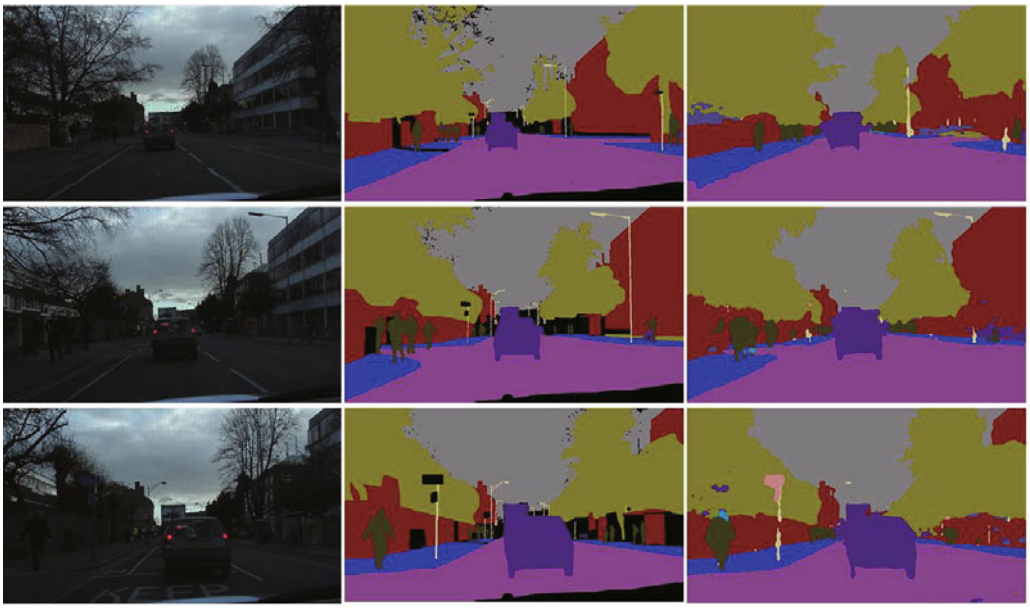

Fig. 4 Visual results on the CamVid test set (left to right: original input image, ground truth, SPSSN results)

most essential classes for environment perception, including road, sidewalk, vehicles, buildings, sky, and vegetation. The reason may be that the network can extract features at both high and low spatial resolutions. Stage-pooling modules can extract important features at multiple levels to help the model efficiently predict objects with varying sizes.

6 Ablation study

As an ablation study, we compared the results of two modules—stage-pooling as shown in Fig. 2 and stream-pooling as shown in Fig. 5—with the same underlying architecture. The results were evaluated on the validation set of Cityscapes (Table 6). Different from stage-pooling, each stream-pooling module took two inputs, one from the previous stream-pooling module, and one from the bottleneck unit. All stream-pooling modules worked at the 128-feature map dimension, and 1/8 of the original spatial resolution. The output of every bottleneck unit was different in channel and spatial dimensions, so pointwise and up-sampling were used to adopt the required feature channel and resolution. The two inputs to the stream-pooling module were then added by element-wise addition, and the pooling module was employed to pool the important features as shown in Fig. 5. In contrast with the stage-pooling module, only the output of the last stream-pooling module was introduced with the shallow branch.

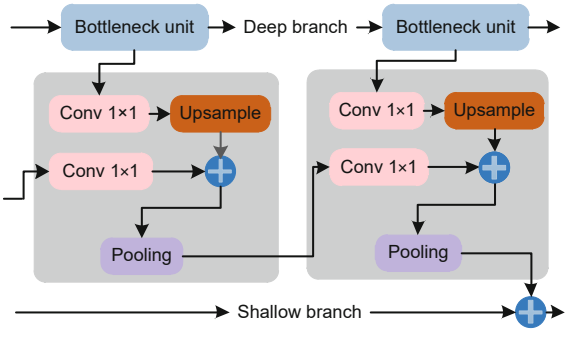

Fig. 5 Illustration of the last two stream-pooling modules

The stream-pooling module achieved 69.7% (mIoU) with 1.47×10^6 parameters, whereas the stage-pooling module achieved 70.4% (mIoU) with 1.42×10^6 parameters on the Cityscapes validation set. The explanation for these findings could be that the stream-pooling module pooled the features at the same resolution of 256×128×128 in all stream-pooling modules as the pooling module was used after up-sampling operation. However, the stage-pooling module pooled the features at different spatial resolutions because up-sampling was done after the pooling module. Additionally, the stage-pooling module used fewer convolution layers than the stream-pooling module, making the network more light-weight and robust. After training on the finely annotated dataset of Cityscapes, fast-SCNN yielded 68.62%, whereas SPSSN yielded 70.4% on the validation set. We also evaluated the inference speed of our model at different input resolutions (Table 7).

Table 6 Evaluation results of the stage-pooling and stream-pooling branches on the Cityscapes validation dataset as an ablation study

Module	mIoU (%)		Number of parameters ($\times10^6$)
	Class	Category	
Fast-SCNN	68.62	–	1.11
Stream-pooling	69.70	85.6	1.47
Stage-pooling	70.40	86.4	1.42

Table 7 Inference speed and FPS of SPSSN at different image resolutions on NVIDIA TITAN Xp (Pascal)

Resolution (pixels)	Time (ms)	FPS
2048×1024	16	59
1024×1024	9	111.44
1024×512	7	135.30

FPS: number of frames per second

7 Conclusions

In this paper, we have proposed a novel stage-pooling architecture for real-time image semantic segmentation. The experimental results demonstrated that stage-pooling increases the network's ability to use low- and intermediate-level features at high resolution. Pooling at different spatial dimensions and stages benefits the model to better segment the multiple objects in real time for diversely changing environments. The stage-pooling technique can increase the efficiency of a network and can also be used for other tasks, such as image classification and object detection, which will be our future work.

Contributors

Saqib MAMOON and Jian-feng LU designed the research. Muhammad Arslan MANZOOR, Fa-en ZHANG, and Zakir ALI processed the data. Saqib MAMOON drafted the manuscript. Muhammad Arslan MANZOOR helped organize the manuscript. Jian-feng LU and Zakir ALI revised and finalized the paper.

Compliance with ethics guidelines

Saqib MAMOON, Muhammad Arslan MANZOOR, Fa-en ZHANG, Zakir ALI, and Jian-feng LU declare that they have no conflict of interest.

References

Badrinarayanan V, Kendall A, Cipolla R, 2017. SegNet: a deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans Patt Anal Mach Intell*, 39(12):2481-2495. https://doi.org/10.1109/TPAMI.2016.2644615

Brostow GJ, Fauqueur J, Cipolla R, 2009. Semantic object classes in video: a high-definition ground truth database. *Patt Recogn Lett*, 30(2):88-97. https://doi.org/10.1016/j.patrec.2008.04.005

Chen LC, Papandreou G, Schroff F, et al., 2017. Rethinking atrous convolution for semantic image segmentation. https://arxiv.org/abs/1706.05587

Chen LC, Papandreou G, Kokkinos I, et al., 2018. DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans Patt Anal Mach Intell*, 40(4):834-848. https://doi.org/10.1109/TPAMI.2017.2699184

Cheng J, Wang P, Li G, et al., 2018. Recent advances in efficient computation of deep convolutional neural networks. *Front Inform Technol Electron Eng*, 19(1):64-77. https://doi.org/10.1631/FITEE.1700789

Chollet F, 2016. Xception: deep learning with depthwise separable convolutions. https://arxiv.org/abs/1610.02357

Christ PF, Elshaer MEA, Ettlinger F, et al., 2016. Automatic liver and lesion segmentation in CT using cascaded fully convolutional neural networks and 3D conditional random fields. Proc 19th Int Conf on Medical Image Computing and Computer-Assisted Intervention, p.415-423. https://doi.org/10.1007/978-3-319-46723-8_48

Cordts M, Omran M, Ramos S, et al., 2016. The Cityscapes dataset for semantic urban scene understanding. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.3213-3223. https://doi.org/10.1109/CVPR.2016.350

Dai JF, He KM, Li Y, et al., 2016a. Instance-sensitive fully convolutional networks. Proc 14th European Conf on Computer Vision, p.534-549. https://doi.org/10.1007/978-3-319-46466-4_32

Dai JF, Li Y, He KM, et al., 2016b. R-FCN: object detection via region-based fully convolutional networks. Proc 30th Int Conf on Neural Information Processing Systems, p.379-387.

Devlin J, Chang MW, Lee K, et al., 2018. BERT: pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/abs/1810.04805

Han S, Mao HZ, Dally WJ, 2016. Deep compression: compressing deep neural network with pruning, trained quantization and Huffman coding. Proc 4th Int Conf on Learning Representations, p.1-14.

He KM, Zhang XY, Ren SQ, et al., 2016. Deep residual learning for image recognition. IEEE Conf on Computer Vision and Pattern Recognition, p.770-778. https://doi.org/10.1109/CVPR.2016.90

Howard AG, Zhu ML, Chen B, et al., 2017. MobileNets: efficient convolutional neural networks for mobile vision applications. https://arxiv.org/abs/1704.04861

Hu H, Gu JY, Zhang Z, et al., 2017. Relation networks for object detection. http://arxiv.org/abs/1711.11575

Huang G, Liu SC, van der Maaten L, et al., 2017. Condensenet: an efficient densenet using learned group convolutions. https://arxiv.org/abs/1711.09224

Hubara I, Courbariaux M, Soudry D, et al., 2016. Binarized neural networks. Proc 30th Int Conf on Neural Information Processing Systems, p.4114-4122.

Hubara I, Courbariaux M, Soudry D, et al., 2018. Quantized neural networks: training neural networks with low precision weights and activations. *J Mach Learn Res*, 18(187):1-30.

Ioffe S, Szegedy C, 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. Proc 32nd Int Conf on Machine Learning, p.1448-1456.

Jégou S, Drozdzal M, Vazquez D, et al., 2017. The one hundred layers tiramisu: fully convolutional densenets for semantic segmentation. Proc IEEE Conf on Computer Vision and Pattern Recognition Workshops, p.1175-1183. https://doi.org/10.1109/CVPRW.2017.156

Lee H, Matin T, Gleeson F, et al., 2019. Efficient 3D fully convolutional networks for pulmonary lobe segmentation in CT images. https://arxiv.org/abs/1909.07474

Li C, Shi CJR, 2018. Constrained optimization based low-rank approximation of deep neural networks. Proc 15th European Conf on Computer Vision, p.746-761. https://doi.org/10.1007/978-3-030-01249-6_45

Li H, Kadav A, Durdanovic I, et al., 2016. Pruning filters for efficient ConvNets. https://arxiv.org/abs/1608.08710

Li HC, Xiong PF, Fan HQ, et al., 2019. DFANet: deep feature aggregation for real-time semantic segmentation. https://arxiv.org/abs/1904.02216

Lin GS, Shen CH, van den Hengel A, et al., 2016. Efficient piecewise training of deep structured models for semantic segmentation. IEEE Conf on Computer Vision and Pattern Recognition, p.3194-3203. https://doi.org/10.1109/CVPR.2016.348

Lin GS, Liu FY, Milan A, et al., 2019. RefineNet: multi-path refinement networks for dense prediction. *IEEE Trans Patt Anal Mach Intell*, p.1228-1242. https://doi.org/10.1109/TPAMI.2019.2893630

Liu ZW, Li XX, Luo P, et al., 2015. Semantic image segmentation via deep parsing network. IEEE Int Conf on Computer Vision, p.1377-1385. https://doi.org/10.1109/ICCV.2015.162

Long J, Shelhamer E, Darrell T, 2014. Fully convolutional networks for semantic segmentation. https://arxiv.org/abs/1411.4038

Ma NN, Zhang XY, Zheng HT, et al., 2018. ShuffleNet V2: practical guidelines for efficient CNN architecture design. Proc 15th European Conf on Computer Vision, p.122-138. https://doi.org/10.1007/978-3-030-01264-9_8

Mazzini D, 2018. Guided upsampling network for real-time semantic segmentation. https://arxiv.org/abs/1807.07466

Mehta S, Rastegari M, Caspi A, et al., 2018. ESPNet: efficient spatial pyramid of dilated convolutions for semantic segmentation. Proc 15th European Conf on Computer Vision, p.561-580. https://doi.org/10.1007/978-3-030-01249-6_34

Mehta S, Rastegari M, Shapiro L, et al., 2019. ESPNetv2: a light-weight, power efficient, and general purpose convolutional neural network. IEEE Conf on Computer Vision and Pattern Recognition, p.9190-9200. https://doi.org/10.1109/CVPR.2019.00941

Nekrasov V, Shen CH, Reid I, 2018. Light-weight RefineNet for real-time semantic segmentation. British Machine Vision Conf, p.125.

Noh H, Hong S, Han B, 2015. Learning deconvolution network for semantic segmentation. https://arxiv.org/abs/1505.04366

Pan Y, 2019. On visual knowledge. *Front Inform Technol Electon Eng*, 20(8):1021-1025. https://doi.org/10.1631/FITEE.1910001

Paszke A, Chaurasia A, Kim S, et al., 2016. ENet: a deep neural network architecture for real-time semantic segmentation. https://arxiv.org/abs/1606.02147

Peng YX, He XT, Zhao JJ, 2018. Object-part attention model for fine-grained image classification. *IEEE Trans Image Process*, 27(3):1487-1500. https://doi.org/10.1109/TIP.2017.2774041

Poudel RPK, Bonde U, Liwicki S, et al., 2018. ContextNet: exploring context and detail for semantic segmentation in real-time. https://arxiv.org/abs/1805.04554

Poudel RPK, Liwicki S, Cipolla R, 2019. Fast-SCNN: fast semantic segmentation network. https://arxiv.org/abs/1902.04502

Rastegari M, Ordonez V, Redmon J, et al., 2016. XNOR-Net: ImageNet classification using binary convolutional neural networks. Proc 14th European Conf on Computer Vision, p.525-542. https://doi.org/10.1007/978-3-319-46493-0_32

Ren SQ, He KM, Girshick R, et al., 2017. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans Patt Anal Mach Intell*, 39(6):1137-1149. https://doi.org/10.1109/TPAMI.2016.2577031

Romera E, Álvarez JM, Bergasa LM, et al., 2018. ERFNet: efficient residual factorized ConvNet for real-time semantic segmentation. *IEEE Trans Intell Transp Syst*, 19(1):263-272. https://doi.org/10.1109/TITS.2017.2750080

Salvador A, Bellver M, Campos V, et al., 2017. Recurrent neural networks for semantic instance segmentation. https://arxiv.org/abs/1712.00617

Sandler M, Howard A, Zhu ML, et al., 2018. MobileNetV2: inverted residuals and linear bottlenecks. IEEE Conf on Computer Vision and Pattern Recognition, p.4510-4520. https://doi.org/10.1109/CVPR.2018.00474

Sherrah J, 2016. Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. https://arxiv.org/abs/1606.02585

Siam M, Gamal M, Abdel-Razek M, et al., 2018. A comparative study of real-time semantic segmentation for autonomous driving. IEEE Conf on Computer Vision and Pattern Recognition Workshops, p.587-597. https://doi.org/10.1109/CVPRW.2018.00101

Soudry D, Hubara I, Meir R, 2014. Expectation backpropagation: parameter-free training of multilayer neural networks with continuous or discrete weights. Proc 27th Int Conf on Neural Information Processing Systems, p.963-971.

Sturgess P, Alahari K, Ladicky L, et al., 2009. Combining appearance and structure from motion features for road scene understanding. British Machine Vision Conf, p.1-11. https://doi.org/10.5244/C.23.62

Szegedy C, Vanhoucke V, Ioffe S, et al., 2015. Rethinking the inception architecture for computer vision. https://arxiv.org/abs/1512.00567

Türkmen S, Heikkilä J, 2019. An efficient solution for semantic segmentation: ShuffleNet V2 with atrous separable convolutions. Proc 21st Scandinavian Conf on Image Analysis, p.41-53. https://doi.org/10.1007/978-3-030-20205-7_4

Visin F, Kastner K, Courville AC, et al., 2015. ReSeg: a recurrent neural network for object segmentation. https://arxiv.org/abs/1511.07053

Wen W, Wu CP, Wang YD, et al., 2016. Learning structured sparsity in deep neural networks. Proc 30th Int Conf on Neural Information Processing Systems, p.1-9.

Wilson AC, Roelofs R, Stern M, et al., 2017. The marginal value of adaptive gradient methods in machine learning. Proc 31st Int Conf on Neural Information Processing Systems, p.1-14.

Wu S, Li GQ, Chen F, et al., 2018. Training and inference with integers in deep neural networks. https://arxiv.org/abs/1802.04680

Xiang W, Mao HD, Athitsos V, 2019. ThunderNet: a turbo unified network for real-time semantic segmentation. IEEE Winter Conf on Applications of Computer Vision, p.1789-1796. https://doi.org/10.1109/WACV.2019.00195

Yang J, Liu QS, Zhang KH, 2017. Stacked hourglass network for robust facial landmark localisation. IEEE Conf on Computer Vision and Pattern Recognition Workshops, p.2025-2033. https://doi.org/10.1109/CVPRW.2017.253

Yu CQ, Wang JB, Peng C, et al., 2018. BiSeNet: bilateral segmentation network for real-time semantic segmentation. Proc 15th European Conf on Computer Vision, p.334-349. https://doi.org/10.1007/978-3-030-01261-8_20

Yu F, Koltun V, 2016. Multi-scale context aggregation by dilated convolutions. Proc 4th Int Conf on Learning Representations, p.1-13.

Yu F, Koltun V, Funkhouser T, 2017. Dilated residual networks. IEEE Conf on Computer Vision and Pattern Recognition, p.636-644. https://doi.org/10.1109/CVPR.2017.75

Zhang JC, Peng YX, 2019a. Hierarchical vision-language alignment for video captioning. Proc 25th Int Conf on Multimedia Modeling, p.42-54. https://doi.org/10.1007/978-3-030-05710-7_4

Zhang JC, Peng YX, 2019b. Object-aware aggregation with bidirectional temporal graph for video captioning. https://arxiv.org/abs/1906.04375

Zhang QS, Zhu SC, 2018. Visual interpretatbility for deep learning: a survey. *Front Inform Technol Electron Eng*, 19(1):27-39. https://doi.org/10.1631/FITEE.1700808

Zhao HS, Shi JP, Qi XJ, et al., 2017. Pyramid scene parsing network. IEEE Conf on Computer Vision and Pattern Recognition, p.6230-6239. https://doi.org/10.1109/CVPR.2017.660

Zhao HS, Qi XJ, Shen XY, et al., 2018. ICNet for real-time semantic segmentation on high-resolution images. Proc 15th European Conf on Computer Vision, p.418-434. https://doi.org/10.1007/978-3-030-01219-9_25

Zheng S, Jayasumana S, Romera-Paredes B, et al., 2015. Conditional random fields as recurrent neural networks. IEEE Int Conf on Computer Vision, p.1529-1537. https://doi.org/10.1109/ICCV.2015.179