Applied Soft Computing Journal 96 (2020) 106682

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

AGLNet: Towards real-time semantic segmentation of self-driving images via attention-guided lightweight network

Quan Zhou [a,*], Yu Wang [a], Yawen Fan [a], Xiaofu Wu [a], Suofei Zhang [b], Bin Kang [b], Longin Jan Latecki [c]

[a] National Engineering Research Center of Communications and Networking, Nanjing University of Posts and Telecommunications, Nanjing, 21003, China
[b] Department of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, 21003, China
[c] Department of Computer and Information Science, Temple University, Philadelphia, USA

ARTICLE INFO

Article history:
Received 9 January 2020
Received in revised form 17 July 2020
Accepted 22 August 2020
Available online 2 September 2020

Keywords:
Robot vision
Self-driving
Real-time semantic segmentation
Convolutional neural networks
Encoder–decoder networks

ABSTRACT

The extensive computational burden limits the usage of convolutional neural networks (CNNs) in edge devices for image semantic segmentation, which plays a significant role in many real-world applications, such as augmented reality, robotics, and self-driving. To address this problem, this paper presents an attention-guided lightweight network, namely *AGLNet*, which employs an encoder–decoder architecture for real-time semantic segmentation. Specifically, the encoder adopts a novel residual module to abstract feature representations, where two new operations, channel split and shuffle, are utilized to greatly reduce computation cost while maintaining higher segmentation accuracy. On the other hand, instead of using complicated dilated convolution and artificially designed architecture, two types of attention mechanism are subsequently employed in the decoder to upsample features to match input resolution. Specifically, a factorized attention pyramid module (FAPM) is used to explore hierarchical spatial attention from high-level output, still remaining fewer model parameters. To delineate object shapes and boundaries, a global attention upsample module (GAUM) is adopted as global guidance for high-level features. The comprehensive experiments demonstrate that our approach achieves state-of-the-art results in terms of speed and accuracy on three self-driving datasets: CityScapes, CamVid, and Mapillary Vistas. *AGLNet* achieves 71.3%, 69.4%, and 30.7% mean IoU on these datasets with only 1.12M model parameters. Our method also achieves 52 FPS, 90 FPS, and 53 FPS inference speed, respectively, using a single GTX 1080Ti GPU. Our code is open-source and available at https://github.com/xiaoyufenfei/Efficient-Segmentation-Networks.

1. Introduction

Recently, building deeper and larger convolutional neural networks (CNNs) is a primary trend for solving robot vision tasks, such as image classification [1–3], object detection [4–6], and semantic segmentation [7–9]. To improve the representation ability of visual data, most accurate CNNs usually have hundreds even thousands of convolutional layers and feature channels, e.g., ResNet family [1,10,11]. In spite of achieving higher performance, these advances are at the sacrifice of running time and inference speed. Especially in the context of many real-world scenarios, such as augmented reality, robotics, and self-driving, the computationally cheaper networks with smaller model size are often required to carry out online estimation and decision in a timely fashion. Therefore, those state-of-the-art networks requiring enormous resources are not suitable for computationally limited mobile platforms (e.g., drones, robots, and smartphones), which have limited energy overhead, restrictive memory constraints, and reduced computational capabilities. This kind of limitation is particularly prominent on the computationally heavy task of semantic segmentation [8,9,12–14], which plays a significant role in robot vision [15,16]. The goal here aims at helping robotics to understand surroundings by partitioning an input image into a serious of disjoint regions, where each one is associated with a pre-defined semantic label including stuff (e.g., sky, road, buildings) and discrete objects (e.g., person, car, traffic light).

In order to adapt to real-world applications, many lightweight style convolutional neural networks have been designed to leverage the segmentation accuracy and implementing efficiency, which can be roughly divided into two categories: network compression [17–20] and convolution factorization [21–23]. The first category prefers to reduce model size by compressing pre-trained

* Corresponding author.
E-mail address: quan.zhou@njupt.edu.cn (Q. Zhou).

https://doi.org/10.1016/j.asoc.2020.106682
1568-4946/

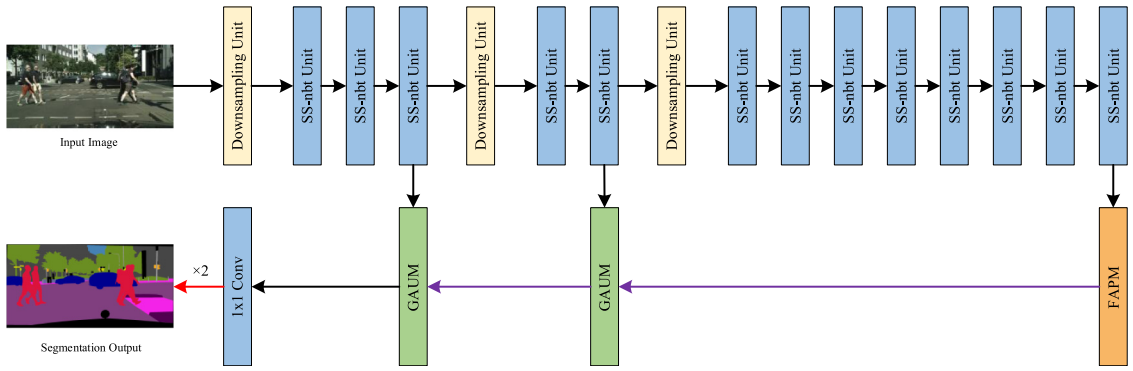

Fig. 1. The detailed architecture of *AGLNet*. We use SS-nbt to extract dense Features in encoder. Then FAPM and GAUM are employed to estimate precise segmentation results with localization details. The purple and red lines represent the deconvolution and Bilinear upsampling, respectively. (Best viewed in color).

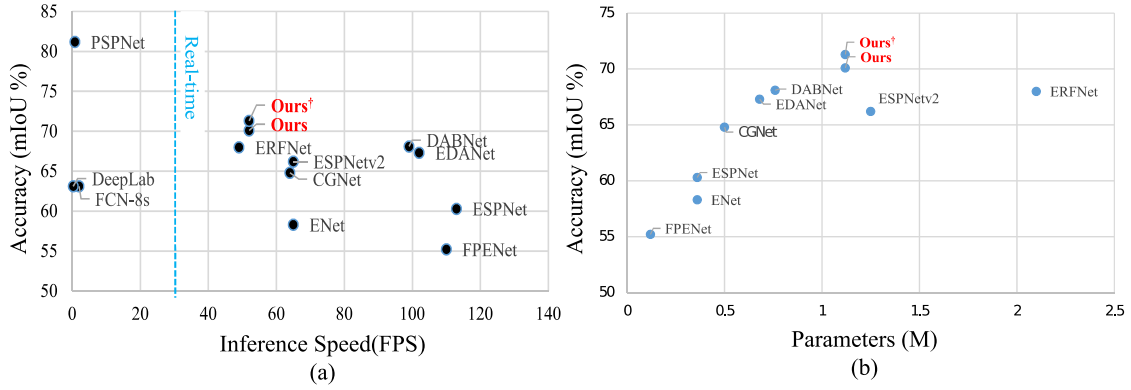

Fig. 2. Comparison with state-of-the-art networks in terms of available trade-off between accuracy and efficiency. From left to right are (a) segmentation accuracy vs. frames per second, and (b) segmentation accuracy vs. model size. '*Ours*' and '*Ours*†' denote training *AGLNet* with or without using coarse annotation data in CityScapes. (Best viewed in color).

networks, including hashing [17], pruning [18], and quantization [19,20]. Quantization networks [19,20] utilize less bits to encode model parameters instead of changing network architecture by pruning network weights [18]. To further remove the redundancy, an alternative approach to lighten CNNs depends on sparse coding theory [24,25], where the model weights are always sparse, thus only small number of parameters are involved in inference calculation. On the contrary, motivated from the convolution factorization principle (CFP) [10,22,23,26] that decomposes a standard convolution, the second category focuses on directly training a lightweight network from the perspective of reducing convolutional operation. For example, group convolution and depthwise separable convolution [2,10] are widely used in MobileNet [23,27] and ShuffleNet families [26,28]. ENet [21] employs ResNet [1] as backbone to perform efficient inference. Zhao et al. [29] propose a cascade network that incorporates high-level label guidance to improve performance. In [13,22,30], an encoder–decoder architecture is adopted to restore object details, which greatly reduce the number of parameters while maintaining the accuracy. Although these advances have conducted preliminary research on designing lightweight architecture networks for semantic segmentation, pursuing the best accuracy in very limited computational budgets still remains an open research problem for real-time semantic segmentation in robot vision.

In this paper, we aim at solving this trade-off as a whole, considering accuracy and run-time efficiency issues equally relevant. We introduce a novel lightweight network, called *AGLNet*, using an attention-guild encoder–decoder architecture for real-time semantic segmentation. As shown in Fig. 1, our *AGLNet* is composed of two parts: encoder and decoder network. Motivated from CFP [10,22,23,26], the core unit of encoder is a computationally efficient residual module, adopting Split and Shuffle in a non-bottleneck structure (SS-nbt), that leverages identity mapping and 1-D factorized convolutions with channel split and shuffle. While the identity mapping allows the convolutions to learn residual functions that facilitate training, the split and shuffle operations enhance the information exchange within the feature channels while remaining similar computational costs compared to 1D factorized convolutions [22]. In the decoder, instead of using complicated dilated convolution [30,31] and artificially designed architecture, two types of attention mechanism are employed to upsample features to match the resolution of input image. As illustrated in Fig. 1, a factorized attention pyramid module (FAPM) is used to extract dense features through exploring hierarchical spatial attention from low-resolution high-level output, where the factorized convolution is utilized to further lighten the entire network. Additionally, to delineate object shapes and boundaries, a global attention upsample module (GAUM) from low-level features is adopted as a global guidance for high-level semantics, in spite of adding a bit of computational burden, but achieving significant performance improvement. More specifically, the spatial attention is calculated from low-level features to assign weights for each pixel location of high-level features. Thereafter, channel attention are encoded to reweight channel features, where abundant category information can be used to select most important feature channels. Fig. 2 also shows the leaderboard of most recent light-weight networks for semantic segmentation on CityScapes dataset in terms of accuracy and efficiency. It can be seen that, although our method performs slower than ESPNet [30] and FPENet [32], we

improve 10% segmentation accuracy. Compared with DABNet [33] and EDANet [34], our approach reaches the best balance between accuracy and efficiency. In summary, the major contributions of our *AGLNet* are four-folds:

- In contrast to previous lightweight networks [13,22,29,30] that usually employ symmetrical architecture, our *AGLNet* adopts an asymmetrical encoder–decoder network structure. The encoder produces the downsampling features using SS-nbt unit, leading to the smaller network size while maintaining powerful representation ability. Subsequently, the decoder upsamples features with the guidance of attention mechanism, helping to improve segmentation accuracy while remaining high inference efficiency.
- The SS-nbt unit of *AGLNet* adopts a split-transform-merge scheme to design residual layer, where 1D factorized convolution with channel split and shuffle operations leverages network parameters and feature representation, approaching the representational power of large and dense layers, but at a considerably lower computational complexity. In addition, channel shuffle is also differentiable, indicating that our *AGLNet* can be trained in an end-to-end manner.
- Two types of lightweight attention mechanisms, called FAPM and GAUM, are employed in decoder to improve segmentation performance. The hierarchical architecture of FAPM enlarges receptive fields of high-level features, allowing us to gather multiple scale context. On the other hand, the spatial and channel attentions are encoded in GAUM to aggregate the low-level spatial details and recalibrate the significance of feature channels efficiently.
- We test *AGLNet* on three self-driving datasets: CityScapes, CamVid, and Mapillary Vistas. The comprehensive experiments demonstrate that our approach achieves available state-of-the-art results in terms of speed and accuracy. Specifically, *AGLNet* achieves 71.3% and 69.4% mean IoU on the CityScapes and CamVid test set, and 30.7% on the Mapillary Vistas validation set, respectively, with only 1.12M model parameters and 52 FPS, 90 FPS, and 53 FPS inference speed on three datsets using a single GTX 1080Ti GPU.

The remainder of this paper is organized as follows. After a brief discussion of related work in Section 2, the detailed architecture of *AGLNet* is introduced in Section 3. The proposed network has been evaluated on three self-driving datasets: CityScapes [35], CamVid [36], and Mapillary Vistas [37], and the experiments can be found in Section 5. Finally, the concluding remarks and future work are given in Section 6.

2. Related work

In this section, we review the related advances for real-time semantic segmentation in robot vision using efficient CNN architecture.

2.1. Real-time semantic segmentation in robot vision

To aid robot decision, real-time segmental segmentation requires very fast running speed to produce high-quality estimations. As a pioneer work, ENet [21] designs a bottleneck module in residual layer to shrink the model. SegNet [13] employs a small network structure and the skipped connections to achieve high speed. ICNet [29] and ContextNet [38] utilize image pyramid as inputs to build cascaded networks that incorporates high-level label guidance to improve performance. On the contrary, FPENet [39] encodes multi-scale context using efficient feature pyramid to save computational costs. ESPNet [30] adopts dilated convolutions with spatial pyramid to enhance efficiency. BiSeNet [40] extracts high-level semantics and low-level spatial features using context and spatial paths, accelerating inference speed by transferring the computation of depth to two subnetworks. In [22,41], a symmetrical encoder–decoder architecture is employed, which greatly reduce the number of parameters while maintaining the accuracy. Different from these methods, *AGLNet* designs a lightweight network in an asymmetrical encoder–decoder architecture, where the encoder utilizes split-transform-merge strategy to construct residual layer, and decoder employs two types of attention module, FAPM and GAUM, to achieve available trade-off between segmentation accuracy and implementing efficiency.

2.2. Convolutional factorization

Most state-of-the-art efficient networks [23,26–28,30,31] use convolutional factorization that decomposes a standard convolution into several steps to reduce computational complexity. They usually factorize a 2D convolution into two 1D convolution (e.g., decomposing $n \times n$ to $1 \times n$ and $n \times 1$), such as group convolution [3,26,42], depth-wise separable convolution [23], and its dilated version [30,31]. More specifically, group convolution [3,26,42] splits the input feature channels into groups and each group is convolved independently. As a special case of group convolution, depth-wise separable convolution [23] is widely used in many efficient networks [27,28], where a standard convolution is decomposed into two steps: depth-wise and 1×1 pointwise convolution. The first step performs light-weight filtering where each input channel is considered as a group, and the second step learns a linear combinations among all input channels. To learn representations from a large effective receptive field, some lightweight networks [30,31] extend depth-wise convolutions using depth-wise 'dilated' separable convolutions. Other representable networks [22] reduce model size through factoring one 2D convolution (e.g., 3×3) to two 1D convolution (e.g., 1×3 and 3×1). Unlike these efficient network, our *AGLNet* employs SS-nbt unit to avoid pointwise convolution, saving a large number of computational costs. In contrast to ShuffleNets [26,28] that only perform convolution on half number of input feature channels, our *AGLNet* makes full use of all input channels with multiple branches of convolution to improve network representation ability. Additionally, our SS-nbt unit enhances the information exchange within feature channels while maintaining similar computational costs compared to 1D factorized convolutions.

2.3. Visual attention

Motivated from the application of speech recognition [43], visual attention is widely-used in computer vision community in recent years [44–49]. Attention mechanism can be used as global context to guide the feed-forward network for improving performance [46,50]. For example, in [45], the attention of CNN is encoded relying on the scale of the input image. In [46,47], feature channel attention is applied to achieve state-of-the-art results for image recognition task. Some attention networks [40,47,51] embeds global average pooling branch to enlarge the receptive field, and enhances the consistency of dense pixel-wise classification. EncNet [52] also introduces a global pooling scheme [47] to capture high-level semantics and predicts scaling factors that are conditional on these encoded semantics. FPENet [39] adds the attention module to the decoder branches. Unlike these models, our *AGLNet* employs two types of attention module, FAPM and GAUM, which encode the spatial- and channel-based attention, as global context to improve segmentation performance.

An early version of this work was first published in [53]. This journal version extends previous one in three aspects: (1) The previous version still use standard convolution in decoder, leading to the heavy network and slow down the implementing efficiency. Conversely, we apply factorized 1D convolution instead of 2D standard convolution, further reducing the model complexity. (2) In contrast to [53] that only utilizes FAPM, our *AGLNet* employs GAUM as a global guidance to recovery precise resolution details. (3) We have performed more exhaustive evaluations and ablation experiments, and reported more comparisons and improved results.

3. AGLNet

In this section, we first introduce the core unit, SS-nbt, with split and shuffle operations in encoder. Thereafter, two proposed attention modules, FAPM and GAUM, are employed for semantic segmentation task. Finally, we describe the whole encoder–decoder network architecture of *AGLNet*.

3.1. SS-nbt

We focus on solving the efficiency limitation that is essentially present in the residual block, which is used in recent accurate CNNs for image classification [1,11,26] and semantic segmentation [12,21,22]. To reduce computation, the group convolutions [11,26] and depthwise separable convolutions [2,9,23] are adopted as standard steps in residual block. As shown in Fig. 3, the recent years has witnessed multiple successful instances of lightweight residual layer [21,23]. For instance, the bottleneck module (Fig. 3(a)) comes from the standard residual layer of ResNet [1], which requires less computational resources by reducing input channels. Although it is commonly adopted in state-of-the-art networks [21,23], the performance descend drastically when network depth increases. Another two outstanding residual module are non-bottleneck-1D (Fig. 3(b)) and ShuffleNet (Fig. 3(c)), where the first one is a 1D version of standard convolution while the second one utilizes pointwise convolutions (i.e., 1×1 convolution) in bottleneck structure. However, the contrary opinion of [26] claims that pointwise convolution accounts for most of the computational complexity, which is especially disadvantageous for lightweight models.

To balance performance and efficiency given limited computational budgets, we introduce two simple operators, called channel split and shuffle, in residual layer. We refer to this proposed module as split-shuffle-non-bottleneck (SS-nbt), as depicted in Fig. 3(d). Motivated from [10,20], a *split-transform-merge* strategy is employed in the design of our SS-nbt, approaching the representational power of large and dense layers, but at a considerably lower computational complexity. At the beginning of each SS-nbt, the input is split into two lower-dimensional branches, where each one has half channels of the input. To avoid pointwise convolution, the transformation is performed using a set of factorized 1D filter kernels (e.g., 1×3, 3×1), and the convolutional outputs of two branches are merged using concatenation so that the number of channels keeps the same. Note we use some factorized dilated 1D convolution to increase the receptive fields. Actually, the input feature can be split into arbitrary number of branches instead of two branches. The extremely case is the branch numbers are equal to the channel number of input feature, where each branch only contains one channel of input feature. Along with the increase of split branches, however, yields repeated access of feature memory, probably slowing down the computational efficiency [27,28]. To facilitate training, the stacked output is added with input through the branch of identity mapping. The same channel shuffle operation [26] is finally used to enable information communication between two split branches. After the shuffle, the next SS-nbt unit begins. It is clear that our residual module is not only efficient, but also accurate. Firstly, factorized convolution involves less model parameters, lead to high computational efficiency in each SS-nbt unit. In contrast to ShuffleNets [26,28] that only perform convolution on half number of input channels, high efficiency allows us to use more feature channels, yielding more powerful representation of visual data. Secondly, in each SS-nbt unit, the merged feature channels are randomly shuffled, and then join into next unit. The channel shuffle operation can be regarded as a kind of feature reuse along with visual data flows forward to the deepest layer of the network, which to some extent enlarges network capacity without significantly increasing complexity.

3.2. FAPM

In this section, we consider how to provide pixel-wise attention for high-level features in a very efficient fashion. In the scenario of real-time semantic segmentation, the pyramid structure [29–31] has been used at several grid scales to abstract features, where dilated convolution is often performed using filter kernels with different size. In spite of increasing receptive field effectively in pixel-level, dilated convolution often results in gridding artifacts that may be harmful for the local consistency of filter responses. Additionally, such structure neglects encoding channel- and pixel-wise attention for high-level features, and, most importantly, lacks considering lightweight architecture of convolution.

Based on above observations, we propose the factorized attention pyramid module (FAPM), as shown in Fig. 4. Our FAPM is consist of two attention parts: pyramid feature attention (PFA) and global pooling attention (GPA), both of which construct pixel-wise attention to improve performance. To increase receptive field, the PFA adopts a hierarchical U-shape structure [54], which integrates contextual features from three different pyramid scales. As green arrows shown in Fig. 4, to reduce computational burden, we first utilize factorized convolution (e.g., 1×7 and 7×1 for scale 1, 1×5 and 5×1 for scale 2, and 1×3 and 3×1 for scale 3) with stride 2 for each pyramid scale to downsample feature resolution. These downsampled feature maps are transferred still using factored filter kernels with stride 1 to better extract context cues from each individual pyramid scale. Since high-level feature maps have small resolution, using larger filter kernel size (e.g., 1×7 and 7×1) does not increase too much computational budgets. Thereafter, the transferred features are sequentially enlarged through bilinear upsampling, as red arrows shown in Fig. 4, and then added to the counterpart of different scales step-by-step. Finally, the pixel-level attention produced from PFA is multiplied by the transferred feature maps, generated from original features of CNNs using 1×1 pointwise convolution. On the other side, to further enhance performance, a GPA branch is introduced to integrate global context prior attention. More specifically, GPA branch first utilizes global average pooling to encode channel attention, then an 1×1 convolution is applied to learn the linear combination of input channels. At the end of this branch, bilinear upsampling is once again employed to match the resolution of input feature maps.

Benefiting from the hierarchical architecture of PFA, FAPM can capture multi-scale context cues, and produce pixel-level attention for convolutional features. Unlike [9] and [55] that stack multi-scale feature maps, our context information is abstracted using factorized convolution, and pixel-wisely multiplied with original convolutional features, without introducing too much computational budgets.

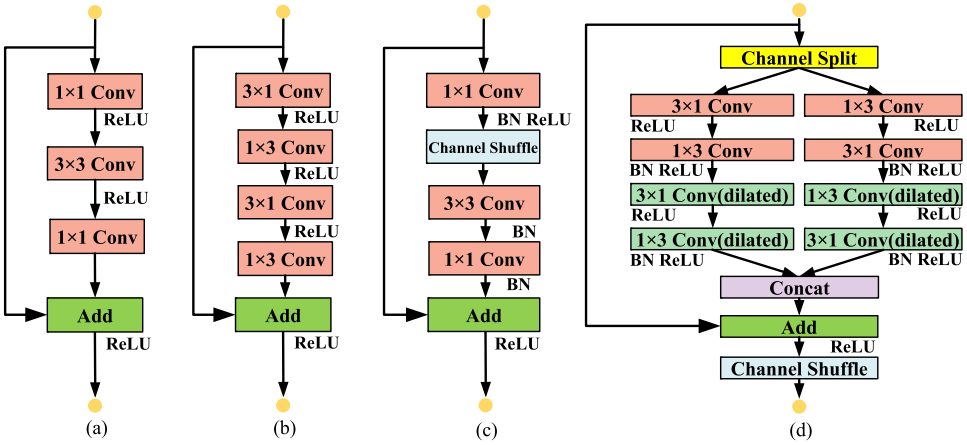

Fig. 3. Comparison of different residual layer modules. From left to right are (a) bottleneck [21,23], (b) non-bottleneck-1D [22], (c) ShuffleNet [26], and (d) our SS-nbt module. (Best viewed in color).

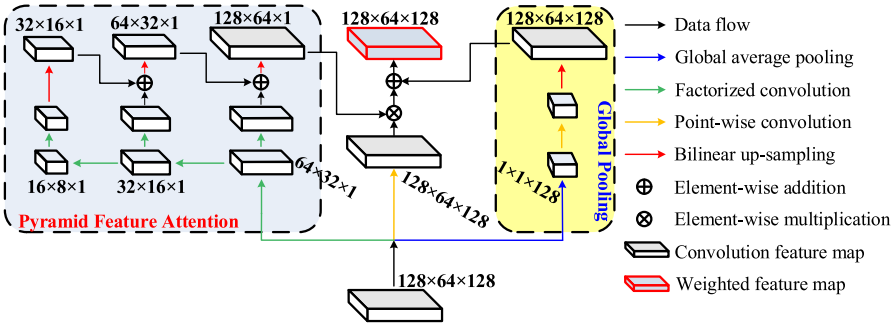

Fig. 4. Architecture of FAPM. Our FAPM is consist of two attention parts: pyramid feature attention and global pooling attention. (Best viewed in color).

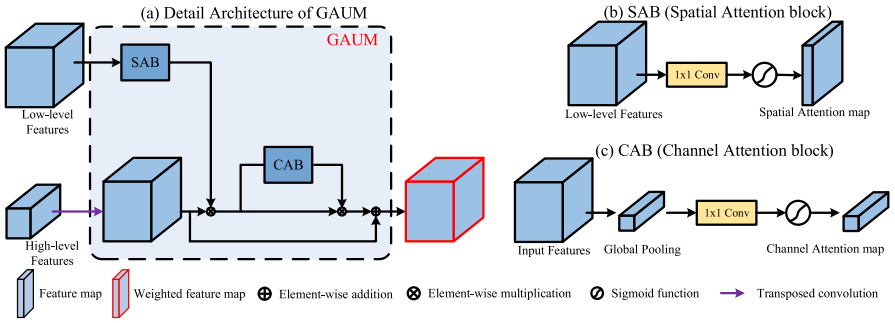

Fig. 5. (a) Detail architecture of GAUM. (b) and (c) are spatial- and channel-wise attention block used in (a), respectively. (Best viewed in color).

3.3. GAUM

The commonly-used architecture in encoder–decoder network for semantic segmentation is U-shape model [12,54–56], which is designed to aggregate features extracted from intermediate convolution layers to recover original input resolution. Some approaches adopts naive decoder structure using simple bilinear upsampling [8,55] or transposed convolution [54], where the low-level spatial information is often ignored leading to coarse segmentation output. The alternative methods [12,56] make an effort to refine object shapes and boundaries by aggregating low-level and upsampled features, leading to complicated decoder modules that are at the sacrifice of running time and speed. Other networks [32,57] resort to employ global attention that squeezes high-level context and embeds it into low-level features as a guidance. Although low-level features are rich in spatial details while high-level features contain abundant context semantics, it is very difficult to aggregate different scale features due to their different feature resolutions and channels. Motivated from [58], this section introduces a novel information fusion module, GAUM, which encodes spatial- and channel-wise attention to improve representation ability of visual data, still maintaining low computational costs.

As shown in Fig. 5(a), the GAUM is mainly composed by two parts: spatial attention block (SAB) and channel attention block (CAB). As purple arrow shown in Fig. 5(a), a transposed convolution is first applied to enlarge resolution of high-level features. These upsampled feature maps are sequentially multiplied by the outputs of SAB and CAB, where the first one reweights the filter responses for each pixel localization, and second one assigns different weight for each feature channel. Finally, the weighted features are fused with upsampled features by element-wise addition. Immediately below, we elaborated on the details of SAB and CAB, respectively.

3.3.1. SAB

Considering that the category-based distribution is uneven on the different image pixels, we propose SAB to make the network pay more attention to informative features. Let $\boldsymbol{X}$ be the input feature maps, $\boldsymbol{f}$ stands for a 1×1 convolution, and $*$ denotes convolutional operation. Then the spatial attention map $\boldsymbol{S}$ is defined as:

$$\boldsymbol{S} = \sigma(\boldsymbol{X} * \boldsymbol{f}), \tag{1}$$

where $\sigma(\cdot)$ represents sigmoid function. After transformation, the shape of $\boldsymbol{X}$ changes from $C \times H \times W$ to $1 \times H \times W$. Finally, we element-wise multiply the input $\boldsymbol{X}$ and the spatial weight map $\boldsymbol{S}$ to get our weighted feature maps $\boldsymbol{X}_s$:

$$\boldsymbol{X}_s = \boldsymbol{X} \otimes \boldsymbol{S}, \tag{2}$$

where $\otimes$ denotes the element-wise manipulation.

3.3.2. CAB

Our channel attention mainly concerns that different channel features have totally different weights. Let $\boldsymbol{X}_s(i, j)$ stands for the values of $\boldsymbol{X}_s$ at pixel position (i, j). Firstly, a global average pooling is performed on $\boldsymbol{X}_s$, which encodes the channel-wise global spatial information into a channel descriptor:

$$\boldsymbol{G} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \boldsymbol{X}_s(i, j), \tag{3}$$

As a result, the shape of $\boldsymbol{X}_s$ changes from $C \times H \times W$ to $1 \times 1 \times C$. Similar to SAB, $\boldsymbol{G}$ is first directly fed into an 1×1 convolutional layer, and then passed through a sigmoid function, resulting in the channel attention map $\boldsymbol{C}$:

$$\boldsymbol{C} = \sigma(\boldsymbol{G} * \boldsymbol{f}), \tag{4}$$

The final weighted feature maps are obtained by multiplying feature map $\boldsymbol{X}_s$ and the attention map $\boldsymbol{C}$:

$$\boldsymbol{X}_{s,c} = \boldsymbol{X}_s \otimes \boldsymbol{C} = \boldsymbol{X} \otimes \boldsymbol{S} \otimes \boldsymbol{C}, \tag{5}$$

The abstracted spatial attention map produced from low-level features encodes the importance of each pixel, which focuses on localizing the objects and refining the corresponding shapes and boundaries with spatial details. Conversely, the squeezed channel attention map generated from upsampled high-level features reflects the importance of each channel, which focuses on the global context to provide content information. The GAUM extracts these two kinds of attention, through which semantic concepts and spatial details are efficiently embedded into each upsampling stage of our *AGLNet*.

3.4. Network architecture of AGLNet

The entire network architecture is shown in Fig. 1. Our *AGLNet* follows a lightweight encoder–decoder architecture with FAPM and GAUM. Unlike [13,54], our approach employs an asymmetric sequential architecture, where a encoder produces downsampled feature maps, and a subsequent decoder upsamples the feature maps to match input resolution. In order to preserve spatial information and reduce number of parameters, the total downsampling rate is 8. The detailed structure of the proposed model is shown in Table 1.

Besides SS-nbt unit and FAPM, the encoder also includes downsampling unit, which is performed by stacking two parallel outputs of a single 3×3 convolution with stride 2 and a Max-pooling. Downsampling enables more deeper network to gather context, while at the same time helps to reduce computation. In the similar spirit of [10], we postpone downsampling in encoder, maintaining more spatial information that is benefit to improve performance. Moreover, the usage of dilated convolutions [22,59] allows our architecture to have large receptive field, further leading to an improvement in accuracy. Compared to the use of larger kernel sizes, this technique has been proven more effective in terms of computational cost and parameters. For the decoder, two GAUMs are used to aggregate features and recover the resolution step-by-step. Then a 1×1 convolutional layer is applied to map the feature channels to the numbers of object categories, and a $2\times$ bilinear upsampling is used to produce the pixel-level classifier.

Table 1
Detailed architecture of proposed *AGLNet*. Input image size is $512 \times 1024 \times 3$. "Output size" denotes the dimension of output feature maps, C is the number of classes.

Stage	Layer	Operator	Mode	Channel	Output size
Encoder	1	Downsampling Unit	–	32	256×512
	2–4	$3\times$ SS-nbt Unit	dilated 1	32	256×512
	5	Downsampling Unit	–	64	128×256
	6–7	$2\times$ SS-nbt Unit	dilated 1	64	128×256
	8	Downsampling Unit	–	128	64×128
	9	SS-nbt Unit	dilated 1	128	64×128
	10	SS-nbt Unit	dilated 2	128	64×128
	11	SS-nbt Unit	dilated 5	128	64×128
	12	SS-nbt Unit	dilated 9	128	64×128
	13	SS-nbt Unit	dilated 2	128	64×128
	14	SS-nbt Unit	dilated 5	128	64×128
	15	SS-nbt Unit	dilated 9	128	64×128
	16	SS-nbt Unit	dilated 17	128	64×128
	17	FAPM	–	128	64×128
Decoder	18	GAUM	–	64	128×256
	19	GAUM	–	32	256×512
	20	1×1 Conv	stride 1	C	256×512
	21	Bilinear interpolation	$\times 2$	C	512×1024

4. End-to-end training of *AGLNet*

In training our *AGLNet*, one major problem is category unbalancing, where there is large variation in the number of training samples in each class. A typical example is the category of "traffic sign" and "road" in CityScapes dataset, in which the object instants of first class occupy a very small number of image regions, while those of the second one occupy a large number of pixels. Therefore, we train *AGLNet* in end-to-end manner using weighted cross entropy loss function. Let $z_k(\boldsymbol{x}, \boldsymbol{\theta})$ denotes the unnormalized log probabilities for pixel $\boldsymbol{x}$ with kth category given network parameters $\boldsymbol{\theta}$, then the soft-max function $p_k(\boldsymbol{x}, \boldsymbol{\theta})$ is defined as:

$$p_k(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{\exp\{z_k(\boldsymbol{x}, \boldsymbol{\theta})\}}{\sum_{k'}^{\mathcal{K}} \exp\{z_{k'}(\boldsymbol{x}, \boldsymbol{\theta})\}}, \tag{6}$$

where $\mathcal{K}$ is the total number of pre-defined object categories. In the inference process, the kth semantic category is assigned to pixel $\boldsymbol{x}$ if it achieves the highest predicted probability $k^* = \arg\max_k p_k(\boldsymbol{x}, \boldsymbol{\theta})$.

For the task of semantic segmentation, the loss is always summed up over all the pixels in a mini-batch. For notation simplify, let N be the total number of pixels in a training batch, y_i denote the ground-truth semantic label for pixel $\boldsymbol{x}_i$, and p_{ik} indicate the estimated probability $p_k(\boldsymbol{x}_i, \boldsymbol{\theta})$ when pixel $\boldsymbol{x}_i$ belongs to category k, our training target is to find an optimal model parameters $\boldsymbol{\theta}^*$ that minimizes the weighted cross-entropy loss $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta})$:

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}), \tag{7}$$

For CityScapes dataset [35], unbalanced training samples often result in the preference on common categories that appears frequently and less improvement on the hard objective at training

stage. In order to solve this problem, we utilize the Online Hard Example Mining (OHEM) scheme [60] to define our weighted loss function:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = -\frac{1}{\sum_{i=1}^{N}\sum_{k=1}^{\mathcal{K}} \delta(y_i = k, p_{ik} < \eta)} \times \sum_{i=1}^{N}\sum_{k=1}^{\mathcal{K}} \delta(y_i = k, p_{ik} < \eta) \log p_{ik}, \tag{8}$$

where $\eta \in (0, 1]$ is a pre-defined threshold, and $\delta(\cdot)$ denotes a indicator function, which equals one when the inside condition holds, and otherwise equals zero.

For CamVid dataset [36], on the other hand, the weighted loss function is defined as:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = -\sum_{i=1}^{N}\sum_{k=1}^{\mathcal{K}} w_{ik} q_{ik} \log p_{ik}, \tag{9}$$

where $q_{ik} = q(y_i = k|\boldsymbol{x}_i)$ is the ground-truth distribution when semantic label of pixel $\boldsymbol{x}_i$ is k, and w_{ik} represents a weight coefficient, which is always defined as inverse ratio with respect to the counted number of training samples for kth category in training data [10,21].

5. Experiments

This section reports the experimental results of our method on two challenging self-driving datasets: CityScapes [35] and CamVid [36]. Some ablation studies are also conducted to better understand the underlying behavior of our network for semantic segmentation in robot vision.

5.1. Datasets

We test *AGLNet* on Mapillary Vistas [37], CityScapes [35] and CamVid [36] datasets, which are commonly-used benchmarks for real-time semantic segmentation. All datasets focus on city street scenes for self-driving, where a car is treated as an autonomous robot to perceive surroundings, including recognizing, localizing and segmenting object instances in input images. The Mapillary Vistas dataset is a large dataset for panoptic segmentation. It includes 65 object categories (28 for stuff and 37 for objects), and the images have wide range of resolutions. This dataset is densely annotated, in which 18K/2K/5K images are used for training, validation, and testing, respectively. The Cityscapes dataset [35] contains 30 classes and only 19 classes (e.g., road, car, pedestrian, bicycle, sky, etc.) of them are used for semantic segmentation evaluation. This dataset contains 5000 high-resolution (2048 × 1024) pixel-level finely annotated images, which are divided into 2975/500/1525 images for training, validation and testing respectively. It also contains another set of nearly 20,000 images with coarse annotation. On the other hand, CamVid [36] is a smaller dataset, only involving 11 object categories with 701 images. All images are collected from 5 videos with resolution 960 × 720, where the split principles are 367 for training, 101 for validation, and 233 for testing. For fair comparison, we downsample the original image size to 1024 × 512 and 480 × 360, respectively, as input resolution for two datasets.

5.2. Baselines

To show the advantages of our approach, we selected 9 state-of-the-art models as baselines, including ENet [21], ERFNet [22], ESPNet [30], ESPNet V2 [31], CGNet [41], EDANet [34], FSCNNet [61], FPENet [32], and DABNet [33]. Experimental results of some baseline models are produced using default parameter settings given by the authors, while others are directly reported from the published literature. All the baselines are evaluated and measured by the mean intersection-over-union (mIoU) class score [22,61], which is calculated as the mean portion of the intersection between the ground truth and the predictions, averaged across all semantic classes for all datasets.

Table 2
Individual category results of different methods on the Cityscapes *test* sets.

Method	Roa	Sid	Bui	Wal	Fen	Pol	TLi	TSi	Veg	Ter
ENet [21]	96.3	74.2	85.0	32.1	33.2	43.4	34.1	44.0	88.6	61.4
ERFNet [22]	97.7	81.0	89.8	42.5	48.0	56.2	59.8	65.3	91.4	68.2
CGNet [41]	95.9	73.9	89.9	43.9	46.0	52.9	55.9	63.8	91.7	68.3
EDANet [34]	97.8	80.6	89.5	42.0	46.0	52.3	59.8	65.0	91.4	68.7
ESPNet [30]	95.7	73.3	86.6	32.8	36.4	47.0	46.9	55.4	89.8	66.0
ESPNet V2 [31]	97.3	78.6	88.8	43.5	42.1	49.3	52.6	60.0	90.5	66.8
FSCNN [61]	97.4	77.8	87.4	39.7	41.8	35.0	39.4	50.5	88.5	63.3
DABNet [33]	97.8	80.7	90.2	47.9	48.1	56.4	61.8	67.0	92.0	69.5
FPENet [32]	96.4	71.7	84.6	27.1	28.8	43.2	39.2	34.4	89.3	61.3
Ours	97.8	81.0	91.0	51.3	50.6	58.3	63.0	68.5	92.3	71.3
Ours[a]	**99.2**	**82.5**	**92.4**	**52.0**	**52.0**	**59.3**	**64.5**	**69.4**	**93.0**	**73.0**
Method	Sky	Ped	Rid	Car	Tru	Bus	Tra	Mot	Bic	mIoU
ENet [21]	90.6	65.5	38.4	90.6	36.9	50.5	48.1	38.8	55.4	58.3
ERFNet [22]	94.2	76.8	57.1	92.8	50.8	60.1	51.8	47.3	61.7	68.0
CGNet [41]	94.1	76.7	54.2	91.3	41.3	55.9	32.8	41.1	60.9	64.8
EDANet [34]	93.6	75.7	54.3	92.4	40.9	58.7	56.0	50.4	64.0	67.3
ESPNet [30]	92.5	68.5	45.9	89.9	40.0	47.7	40.7	36.4	54.9	60.3
ESPNet V2 [31]	93.3	72.9	53.1	91.8	53.0	65.9	53.2	44.2	59.9	66.2
FSCNN [61]	92.7	65.7	46.4	91.0	**57.0**	**70.3**	**56.5**	40.9	52.6	62.8
DABNet [33]	94.3	80.3	59.2	93.7	46.0	57.1	35.0	50.4	66.8	68.1
FPENet [32]	92.3	68.1	42.7	89.8	29.1	38.9	27.5	29.1	54.5	55.2
Our	94.2	80.1	59.6	93.8	48.4	68.1	42.1	52.4	67.8	70.1
Ours[a]	**95.2**	**81.4**	**60.3**	**95.3**	49.3	69.6	43.5	**53.4**	**69.3**	**71.3**

[a]Methods trained using both fine and coarse data.

Table 3
Speed and accuracy comparison of *AGLNet* on Cityscapes *test* dataset. Except to the lightweight baselines, we also compared with some heavy models, including FCN [8], PSPNet [55] and DeepLab [9].

Method	Input size	Extra data	#Params	FLOPs	FPS	mIoU (%)
SegNet [13]	640 × 360	ImN	29.5M	286G	16.7	57
FCN-8S [8]	512 × 1024	no	–	136.2G	2	63.1
DeepLab [9]	512 × 1024	ImN	262.1M	457.8G	0.25	63.1
RefineNet [12]	512 × 1024	ImN	118.1M	526G	9.1	73.6
OCNet [62]	512 × 1024	ImN	62.6M	549G	8.7	80.1
PSPNet [55]	713 × 713	ImN + Coa.	250.8M	412.2G	0.78	81.2
ENet [21]	512 × 1024	no	0.36M	4.4G	65	58.3
ERFNet [22]	512 × 1024	no	2.1M	26.86G	49	68.0
CGNet [41]	512 × 1024	no	0.5M	7.01G	64	64.8
EDANet [34]	512 × 1024	no	0.68M	8.95G	102	67.3
ESPNet [30]	512 × 1024	no	0.36M	4.7G	113	60.3
ESPNetv2 [31]	512 × 1024	ImN	1.25M	5.85G	65	66.2
FSCNN [61]	512 × 1024	Coa.	1.14M	1.76G	**230**	62.8
DABNet [33]	512 × 1024	no	0.76M	10.46	99	68.1
FPENet [32]	512 × 1024	no	**0.12*M***	**1.58*G***	110	55.2
Ours	512 × 1024	no	1.12M	13.88G	52	70.1
Ours[†]	512 × 1024	Coa.	1.12M	13.88G	52	**71.3**

"ImN" and **"Coa."** mean pre-training model using ImageNet dataset [63] or the coarse annotation set of Cityscapes dataset. **"–"** indicates that the corresponding result is not provided by the methods. We reproduce some models, and evaluate speed under the same setting with our *AGLNet* for fair comparison.

5.3. Implementation details

AGLNet is implemented on the hardware platform of DELL workstation with a single GTX 1080Ti GPU, and trained in an end-to-end manner using Adam optimizer [61] for CityScapes dataset, and the Rectified Adam (RAdam) [64] combined with LookAhead [65] for CamVid dataset. For two datasets, we favor

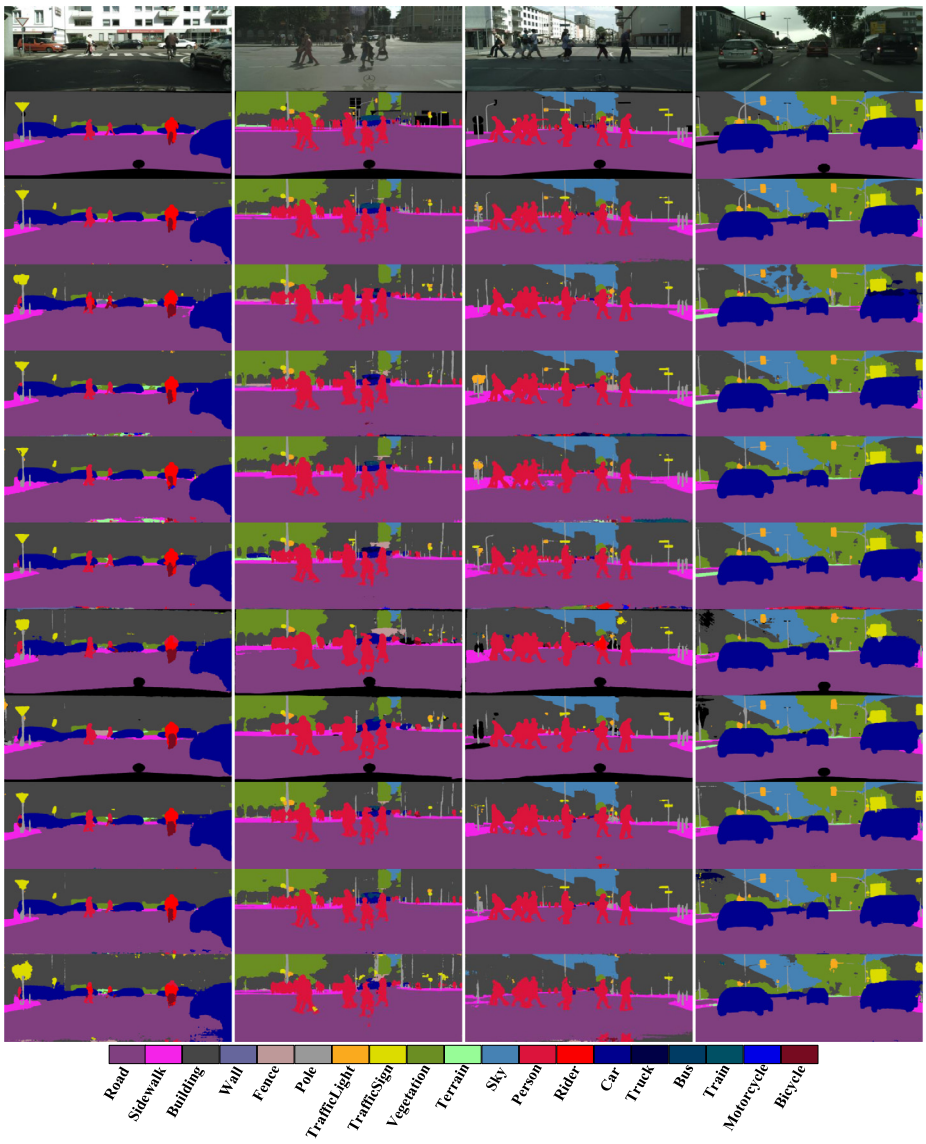

Fig. 6. The visual comparison on Cityscapes val set. From top to bottom are input images, ground truth, segmentation outputs from *AGLNet*, ENet [21], ERFNet [22], CGNet [41], EDANet [34], ESPNet [30]. ESPNetv2 [31], FSCNNet [61], DABNet [33] and FPENet [32]. (Best viewed in color).

a large minibatch size (set as 8) to make full use of the GPU memory. The initial learning rate is set as $5 \times e^{-4}$ and $3 \times e^{-2}$ for two dataset. The 'poly' learning rate policy [14] is adopted in our training process, where the learning rate is updated by $(1 - \frac{iter\_num}{max\_iter})^{power}$ with power 0.9, together with momentum and weight decay, are set to 0.9, 10^{-4}, and the maximum number of training epochs, is set to 500 and 1000, for two datasets, respectively. For data augmentation, we adopted random horizontal flipping, left–right flipping, and random scaling between 0.5 and 2 on the input images during training. Finally, we randomly crop the image into fixed size for training. All images of two datasets were normalized to zero mean and unit variance.

5.4. Evaluation results on CityScapes

For fair comparison, all the baselines are performed using the same hardware platform with a single NVIDIA GTX 1080Ti GPU. Tables 2 and 3 compare our *AGLNet* with selected state-of-the-art networks. The results show that *AGLNet* outperforms these baselines in terms of segmentation accuracy wile still achieves real-time implementing efficiency. Among all the approaches, our method only has 1.12M model size, while achieves 52 FPS inference speed and 70.1% mIoU without using extra training data. Only using extra coarse annotated training data, the segmentation accuracy improves 1.2% to 71.3% mIoU. From Table 3, it is observed that 16 out of the 19 object categories obtains best mIoU scores, especially for some categories, achieving remarkable improvement (e.g., 8.1% for 'Wal' and 2.1% for 'Mot') than the second ranked method. Regarding to the efficiency, *AGLNet* has nearly half size but a bit more faster than ERFNet [29]. Other lightweight baselines are faster than our *AGLNet*, yet at the sacrifice of segmentation accuracy. For example, FSCNN achieves the highest implementing speed, but delivers poor segmentation accuracy of 8.5% drop than our *AGLNet*. We also compare with some heavy models and report the results on Table 3. It shows that our approach achieves higher performance than [8,9] that are not able to perform real-time estimation. Fig. 6 shows some qualitative results on the CityScapes val set. It is demonstrated that, compared with baselines, our *AGLNet* not only correctly classifies object with different scales, but also produces consistent qualitative results for all classes.

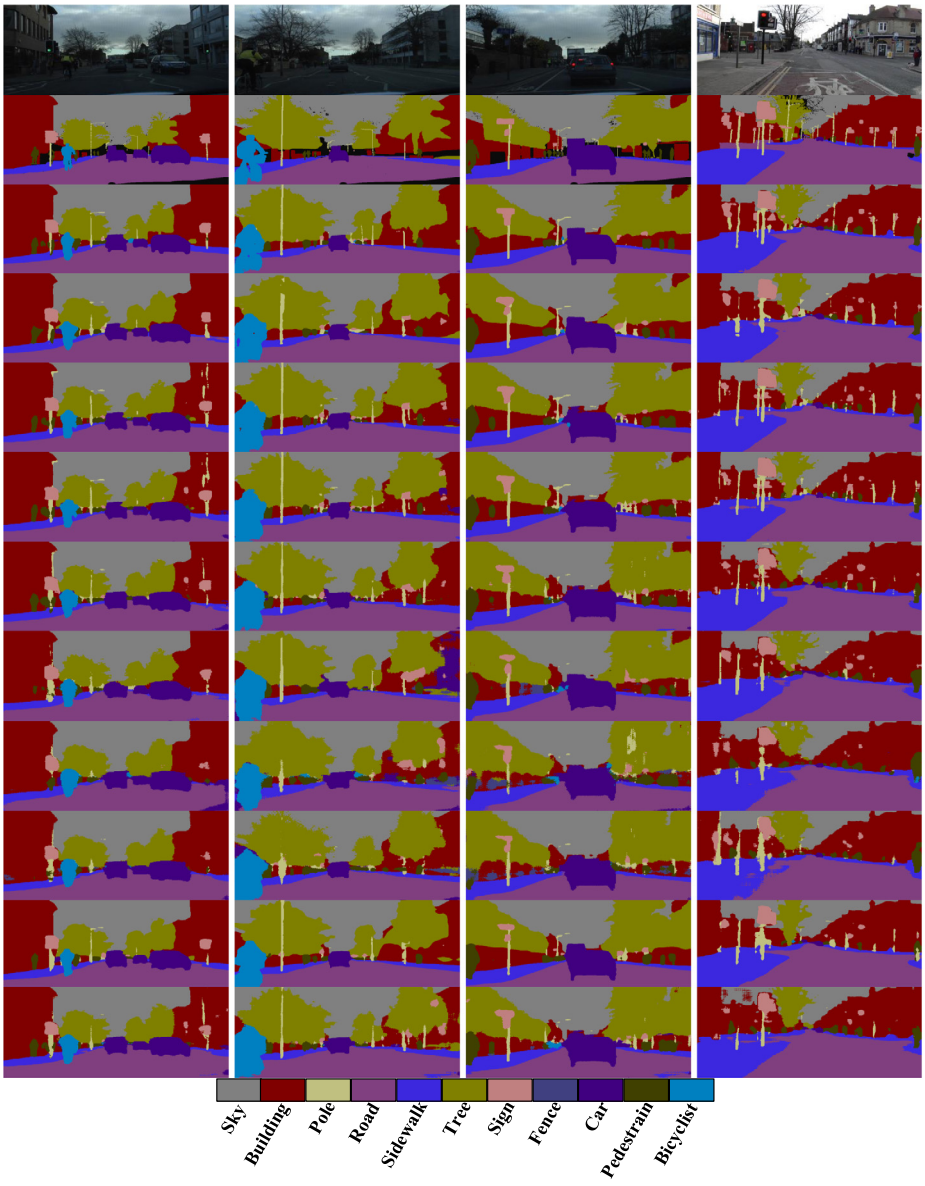

Fig. 7. The visual comparison on CamVid test set. From top to bottom are input images, ground truth, segmentation outputs from *AGLNet*, ENet [21], ERFNet [22], CGNet [41], EDANet [34], ESPNet [30]. ESPNetv2 [31], FSCNNet [61], DABNet [33] and FPENet [32]. (Best viewed in color).

Table 4
Accuracy result of each individual category on CamVid *test* dataset.

Method	Sky	Building	Pole	Road	Sidewalk	Tree	Sign	Fence	Car	Pedestrain	Bicyclist	mIoU (%)
ENet [21]	91.2	74.9	23.4	92.1	73.7	68.1	30.1	20.9	77.3	41.1	45.8	58.1
ERFNet [22]	92.0	81.3	37.8	95.0	81.1	75.0	45.0	36.2	84.3	58.3	58.2	67.7
CGNet [41]	90.8	79.8	28.1	95.3	81.9	73.2	41.6	32.9	81.3	52.9	53.9	64.7
EDANet [34]	89.8	79.4	24.3	94.0	81.0	71.1	37.3	31.4	76.9	51.1	53.5	62.7
ESPNet [30]	**92.0**	75.0	25.0	91.5	73.8	68.4	29.5	23.7	74.5	42.4	45.2	58.2
ESPNetv2 [31]	91.0	71.0	18.1	90.1	67.2	61.3	20.0	21.1	69.7	28.8	33.4	52.0
FSCNN [61]	90.2	74.3	15.0	91.7	72.6	67.9	28.9	17.4	70.1	31.9	35.6	54.2
DABNet [33]	91.1	81.0	29.4	93.8	78.7	74.1	43.0	37.2	81.7	56.2	56.5	65.7
FPENet [32]	91.0	76.3	31.0	93.8	78.3	68.8	32.1	25.1	77.7	45.6	45.6	60.5
Ours	91.8	**82.6**	**39.0**	**95.4**	**83.1**	**76.1**	**45.3**	**39.5**	**87.0**	**61.5**	**62.7**	**69.4**

5.5. Evaluation results on CamVid

We also evaluate *AGLNet* on CamVid [36] dataset, and report the results in Tables 4 and 5. Compared with the selected state-of-the-art baselines, *AGLNet* shows the superior performance in terms of running speed and segmenting accuracy. From Table 4, except one class "sky", *AGLNet* achieves the best performance in rest categories. Note *AGLNet* performs faster on CamVid dataset (52 vs. 90 FPS) than Cityscapes dataset, due to its smaller input image resolutions (1024 $\times$ 512 of Cityscapes and 480 $\times$ 360 of CamVid). Several visual examples of segmentation outputs are shown in Fig. 7.

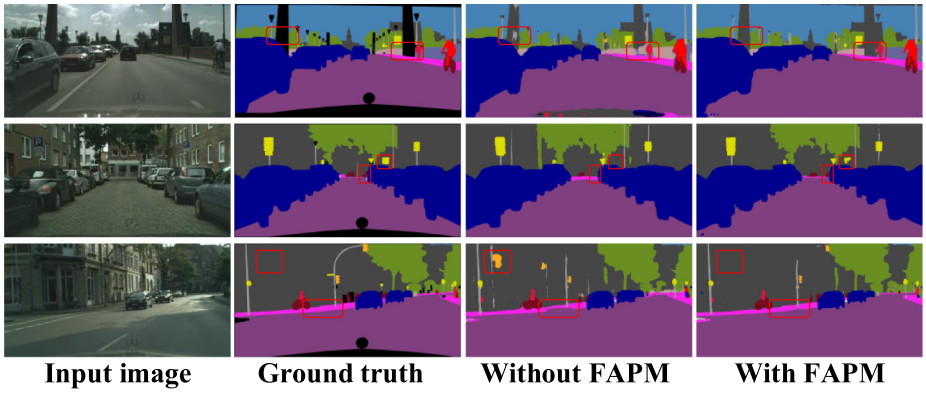

Fig. 8. Visualization results of FAPM on Cityscapes val set. (Best viewed in color).

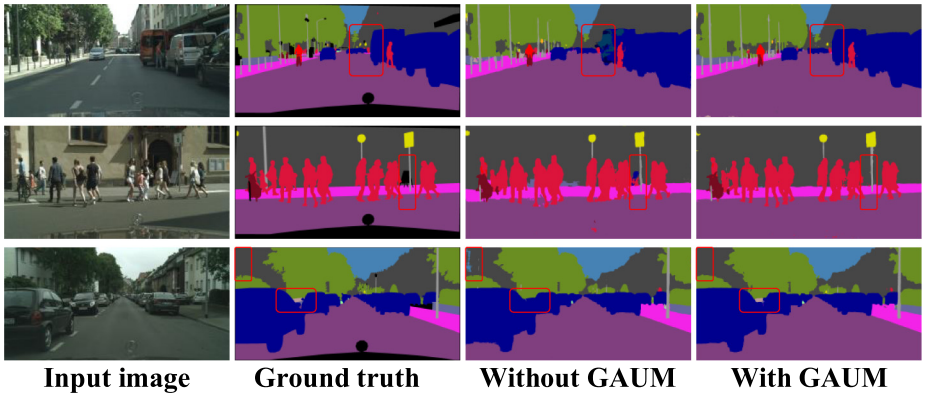

Fig. 9. Visualization results of GAUM on Cityscapes val set. (Best viewed in color).

Table 5
Comparison with state-of-the-art networks on the CamVid *test* dataset.

Method	#Params (M)	FPS	mIoU (%)
ENet [21]	0.36	98.8	58.1
ERFNet [22]	2.10	139.1	67.7
CGNet [41]	0.50	99.1	64.7
EDANet [34]	0.68	175.1	62.7
ESPNet [30]	0.36	190.3	58.2
ESPNetv2 [31]	1.25	118.5	52.0
FSCNN [61]	1.14	**245.1**	54.2
DABNet [33]	0.76	164.5	65.7
FPENet [32]	**0.12**	116.5	60.5
Ours	1.12	90.1	**69.4**

Table 6
Comparison with state-of-the-art methods on the Mapillary Vistas *validation* set in terms of mIOU scores.

Method	Input size	Extra data	# Params	FLOPs	FPS	mIoU (%)
FPENet [32]	1024 × 2048	no	0.76M	20.9	**103**	28.33
DABNet [33]	1024 × 2048	no	0.12M	**3.10**	75	29.60
Ours	1024 × 2048	no	1.12M	24.12	53	**30.70**

5.6. Evaluation results on mapillary Vistas

We then demonstrate that our method scales nicely when augmenting the number and classes on Mapillary Vistas dataset in Table 6. To verify our method on this dataset, we select DABNet [33] and FPENet [32] as baselines. Compared with Cityscapes and CamVid datasets, our method only obtains 30.7% mIoU score due to large number of classes in Mapillary Vistas dataset. Even so, Table 6 still demonstrates that our method performs better than FPENet [32] in terms of mIoU (30.7% vs 28.33%). Although DABNet [33] runs nearly twice faster than our network, our approach still achieves 1.1% improvement in segmentation accuracy.

Table 7
Ablation study results of *AGLNet* on Cityscapes *val* dataset.

Model	FAPM	GAUM	train	val	mIoU (%)	Params (M)
AGLNet			✓		66.12	0.91
AGLNet	✓		✓		67.62	0.95
AGLNet		✓	✓		69.19	1.08
AGLNet	✓	✓	✓		69.39	1.12
AGLNet[a]	✓	✓	✓	✓	74.50	1.12

Params: Model size.
[a]Methods trained using both train and val data.

5.7. Ablative studies

To investigate the effectiveness of two attention module of our proposed *AGLNet*, we conduct ablative studies on Cityscapes val dataset, where FAPM and GAUM are separately added, and combined together to our system. Table 7 reports the contributions of each component and their combinations in terms of mIoU. It is observed that introducing more attention module leads to the improvement of performance. Compared with the baseline that no attention module is adopted, only using FAPM leads to the result of 67.62% mIoU, which brings 1.5% improvement. On the other hand, employing GAUM individually outperforms the baseline by 3.07%, yielding 69.19% segmentation accuracy. This is due to the fact that, compared with FAPM, GAUM takes advantage

of semantics of high-level feature and spatial details of low-level features as interact guidance to improve the performance. Another interesting observation is that, using GAUM has a bit larger model size than using FAPM (e.g., 0.95 vs. 1.08). This is probably because we use two GAUM units in our *AGLNet*. When both attention module is explored, our *AGLNet* achieves highest segmentation accuracy, which improves to 69.39%. We also employ val set to train *AGLNet* using both FAPM and GAUM. It achieves 74.5% mIoU, which indicates that more training data is beneficial to further improve the performance.

Some segmentation outputs of visual examples are illustrated in Figs. 8 and 9, which has the consistent results with Table 7. Specifically, the effects of FAPM can be visualized in Fig. 8. Some details and object boundaries are more clearer (e.g., 'building' and 'sidewalk' in first and third example) and missing tiny objects are correctly classified (e.g., 'bicycle' and 'traffic sign' in second example). FAPM enhances the discrimination ability by capturing multi-scale context information. Meanwhile, Fig. 9 demonstrate that, with our GAUM, some misclassified category are now correctly classified, such as the 'car' in first example and 'tree' in third example. The semantic consistency have been improved obviously.

6. Conclusion remarks and future work

This paper has described a *AGLNet* model, which designs a lightweight encoder–decoder network for real-time semantic segmentation of self-driving images. The encoder adopts channel split and shuffle operations in residual layer, enhancing information communication in the manner of feature reuse. On the other hand, the decoder employs two attention blocks, FAPM and GAUM, where the first one adopts spatial pyramid architecture to enlarge receptive fields without introducing significant computational budgets, and the second one employs the interaction of high-level and low-level features as guidance to improve performance. The entire network is trained end-to-end. To evaluate our method, the experiments are conducted on two popular self-driving datasets: Cityscapes and CamVid. The experimental results show our *AGLNet* achieves best trade-off on CityScapes dataset in terms of segmentation accuracy and implementing efficiency. In the future, we would like to make an effort to quantize model parameters, leading to faster running speed for real-time semantic segmentation.

CRediT authorship contribution statement

Quan Zhou: Conceptualization, Methodology, Writing - original draft. **Yu Wang:** Software, Validation, Investigation, Data curation. **Yawen Fan:** Writing - review & editing, Visualization. **Xiaofu Wu:** Writing - review & editing, Supervision. **Suofei Zhang:** Writing - review & editing, Project administration. **Bin Kang:** Writing - review & editing. **Longin Jan Latecki:** Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the associated editor and all the anonymous reviewers for their valuable comments and insightful suggestions. This work was jointly supported in part by the National Natural Science Foundation of China under Grants 61876093, 61801242, 61671253, the National Natural Science Foundation of Jiangsu Province under Grant BK20181393, the National Science Foundation under Grant IIS-1302164, and in the part by the China Scholarship Council under Grant 201908320072.

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[3] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Annual Conference on Neural Information Processing Systems, 2012, pp. 1097–1105.
[4] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in: Annual Conference on Neural Information Processing Systems, 2015, pp. 91–99.
[5] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You nly look once: unified, real-time object detection, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.
[6] R. Girshick, Fast R-CNN, in: IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.
[7] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.
[8] L. Jonathan, S. Evan, D. Trevor, Fully convolutional networks for semantic segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (4) (2017) 640–651.
[9] L.C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2018) 834–848.
[10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.
[11] S. Xie, R. Girshick, P. Dollar, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2017, pp. 5987–5995.
[12] L. Guosheng, M. Anton, S. Chunhua, I. Reid, RefineNet: multi-Path Refinement Networks for High-Resolution Semantic Segmentation, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2017, pp. 5168–5177.
[13] B. Vijay, A. Kendall, R. Cipolla, Segnet: A deep convolutional encoder-decoder architecture for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (12) (2017) 2481–2495.
[14] H. Zhao, J. Shi, X. Qi, X. Wang, J.Y. Jia, Pyramid scene parsing network, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 6230–6239.
[15] J.D. Chen, Y.K. Cho, Z. Kira, Multi-view incremental segmentation of 3-D point clouds for mobile robots, IEEE Robot. Autom. Lett. 4 (2) (2019) 1240–1246.
[16] K.Q. Li, W.B. Tao, L.M. Liu, Online semantic object segmentation for vision robot collected video, IEEE Access 7 (2) (2019) 107602–107615.
[17] W. Chen, J. Wilson, S. Tyree, K. Weinberger, Y. Chen, Compressing neural networks with the hashing trick, in: International Conference on Machine Learning, 2015, pp. 2285–2294.
[18] S. Han, H. Mao, W.J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, in: International Conference on Learning Representations, 2016, pp. 1–14.
[19] J. Wu, C. Leng, Y. Wang, Q. Hu, J. Cheng, Quantized convolutional neural networks for mobile devices, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 5168–5177.
[20] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, in: European Conference on Computer Vision, 2016, pp. 525–542.
[21] A. Paszke, A. Chaurasia, S. Kim, E. Culurciello, Enet: A deep neural network architecture for real-time semantic segmentation, 2016, arXiv preprint arXiv:1606.02147.
[22] E. Romera, J.M. Alvarez, L.M. Bergasa, R. Arroyo, ERFNet: Efficient residual factorized convnet for real-time semantic segmentation, IEEE Trans. Intell. Transp. Syst. 19 (1) (2018) 263–272.
[23] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: efficient convolutional neural networks for mobile vision applications, 2017, arXiv preprint arXiv:1704.04861.
[24] W. Wen, C. Wu, Y. Wang, Y. Chen, H. Li, Learning structured sparsity in deep neural networks, in: Annual Conference on Neural Information Processing Systems, 2016, pp. 2074–2082.
[25] B. Liu, M. Wang, H. Foroosh, M. Tappen, M. Pensky, Sparse convolutional neural networks, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2015, pp. 806–814.

[26] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2018, pp. 6848–6856.
[27] M. Sandler, A. Howard, M.L. Zhu, A. Zhmoginov, L.C. Chen, Mobilenet V2: Inverted residuals and linear bottlenecks, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2019, pp. 4510–4520.
[28] N. Ma, X.Y. Zhang, H.T. Zheng, J. Sun, Shufflenet v2: Practical guidelines for efficient cnn architecture design, in: European Conference on Computer Vision, 2018, pp. 116–131.
[29] H.S. Zhao, X.J. Qi, X.Y. Shen, J.P. Shi, J.Y. Jia, Icnet for real-time semantic segmentation on high-resolution images, 2018, arXiv preprint arXiv:1704.08545v2.
[30] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, H. Hajishirzi, ESPNet: Efficient spatial pyramid of dilated convolutions for semantic segmentation, 2018, arXiv preprint arXiv:1803.06815v3.
[31] S. Mehta, M. Rastegari, L. Shapiro, H. Hajishirzi, ESPNet V2: A light-weight, power efficient, and general purpose convolutional neural network, 2019, arXiv:1811.11431v3.
[32] M.Y. Liu, H.J. Yin, Feature pyramid encoding network for real-time semantic segmentation, 2019, arXiv preprint arXiv:1909.08599v1.
[33] G. Li, I.Y. Yun, J.H. Kim, J.K. Kim, DABNet: depth-wise asymmetric bottleneck for real-time semantic segmentation, 2019, arXiv:1907.11357v1.
[34] S.Y. Lo, H.M. Hang, S.W. Chan, J.H. Lin, Efficient dense modules of asymmetric convolution for real-time semantic segmentation, 2018, arXiv preprint arXiv:1809.06323v2.
[35] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The scapes dataset for semantic urban scene understanding, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 3213–3223.
[36] G.J. Brostow, J. Shotton, J. Fauqueur, R. Cipolla, Segmentation and recognition using structure from motion point clouds, in: European Conference on Computer Vision, 2008, pp. 44–57.
[37] S.R.B. G. Neuhold, P. Kontschieder, The mapillary vistas dataset for semantic understanding of street scenes, in: IEEE International Conference on Computer Vision, 2017, pp. 4990–4999.
[38] R.P. Poudel, U.B.S. Liwicki, C. Zach, Contextnet: Exploring context and detail for semantic segmentation in real-time, 2018, arXiv preprint arXiv:1805.04554.
[39] M.Y. Liu, H.J. Yin, Feature pyramid encoding network for real-time semantic segmentation, 2019, arXiv preprint arXiv:1909.08599v1.
[40] C.Q. Yu, J.B. Wang, C. Peng, C.X. Gao, G. Yu, N. Sang, Bisenet: Bilateral segmentation network for real-time semantic segmentation, in: European Conference on Computer Vision, 2018, pp. 325–341.
[41] T.Y. Wu, S. Tang, R. Zhang, Y.D. Zhang, CGNet: A light-weight context guided network for semantic segmentation, 2018, arXiv preprint arXiv:1811.08201v1.
[42] G. Huang, S.C. Liu, L.V. der Maaten, K.Q. Weinberger, Condensenet: An efficient densenet using learned group convolutions, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2018, pp. 2752–2761.
[43] J.K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio, Attention-based models for speech recognition, in: Annual Conference on Neural Information Processing Systems, 2015, pp. 577–585.
[44] K. Xu, B. Jimmy, K. Ryan, Show attend and tell: Neural image caption generation with visual attention, in: International Conference on Machine Learning, 2015, pp. 2048–2057.
[45] L.-C. Chen, Y. Yang, J. Wang, W. Xu, A.L. Yuille, Attention to scale: Scale-aware semantic image segmentation, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 3640–3649.
[46] F. Wang, M.Q. Jiang, C. Qian, S. Yang, C. Li, H.Q. Zhang, X.G. Wang, X.O. Tang, Residual attention network for image classification, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2017, pp. 6450–6458.
[47] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, 2017, arXiv preprint arXiv:1709.01507.
[48] H. Lu, D. Wang, Y. Li, J. Li, X. Li, H. Kim, S. Serikawa, I. Humar, CONet: A cognitive ocean network, IEEE Wirel. Commun. 26 (3) (2019) 90–96.
[49] X. Xu, H. Lu, J. Song, Y. Yang, H.T. Shen, X. Li, Ternary adversarial networks with self-supervision for zero-shot cross-modal retrieval, IEEE Trans. Cybern. 50 (6) (2020) 2400–2413.
[50] V. Mnih, N. Heess, A. Graves, Recurrent models of visual attention, in: Annual Conference on Neural Information Processing Systems, 2014, pp. 1–9.
[51] C.Q. Yu, J.B. Wang, C. Peng, C.X. Gao, G. Yu, N. Sang, Learning a discriminative feature network for semantic segmentation, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2018, pp. 325–341.
[52] H. Zhang, K. Dana, J.P. Shi, Z.Y. Zhang, X.G. Wan, A. Tyagi, A. Agrawal, Context encoding for semantic segmentation, 2018, arXiv preprint arXiv:1803.08904.
[53] Y. Wang, Q. Zhou, J. Liu, J. Xiong, G.W. Gao, X.F. Wu, L.J. Latecki, LEDNet: A lightweight encoder-decoder network for real-time semantic segmentation, in: IEEE International Conference on Image Processing, 2019, pp. 177–186.
[54] O. Ronneberger, F. Philipp, B. Thomas, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer Assisted Intervention, 2015, pp. 225–233.
[55] H. Zhao, J. Shi, X. Qi, X. Wang, J.Y. Jia, Pyramid scene parsing network, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 6230–6239.
[56] C. Peng, Z. Xiangyu, Y. Gang, L. Guiming, S. Jian, Large kernel matters: Improve semantic segmentation by global convolutional network, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2017, pp. 1743–1751.
[57] H.C. Li, P.F. Xiong, J. An, L.X. Wang, Pyramid attention network for semantic segmentation, 2018, arXiv:1805.10180.
[58] Z.L. Zhang, X.Y. Zhang, C. Peng, X.Y. Xue, J. Sun, Exfuse: Enhancing feature fusion for semantic segmentation, in: European Conference on Computer Vision, 2018, pp. 269–284.
[59] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, in: International Conference on Learning Representations, 2016, pp. 1–10.
[60] A. Shrivastava, A. Gupta, R. Girshick, Training region-based object detectors with online hard example mining, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 761–769.
[61] R.P. Poudel, S. Liwicki, Fast-scnn: fast semantic segmentation network, 2019, arXiv:1902.04502v1.
[62] Y.H. Yuan, J.D. Wang, OCNet: Object context network for scene parsing, 2018, arXiv:1809.00916v1.
[63] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.
[64] L.Y. Liu, H.M. Jiang, P.C. He, W.Z. Chen, X.D. Liu, J.F. Gao, J.W. Han, On the variance of the adaptive learning rate and beyond, 2019, arXiv:1908.03265v1.
[65] M. Zhang, J. Lucas, J. Ba, G.E. Hinton, Lookahead Optimizer: k steps forward, 1 step back, in: Annual Conference on Neural Information Processing Systems, 2019, pp. 9593–9604.