IEEE ROBOTICS & AUTOMATION LETTERS
A PUBLICATION OF THE IEEE ROBOTICS & AUTOMATION SOCIETY

IEEE Xplore® Digital Library
http://ieeexplore.ieee.org/Xplore

Real-time Semantic Segmentation with Fast Attention

Ping Hu, Federico Perazzi, Fabian Caba Heilbron, Oliver Wang, Zhe Lin, Kate Saenko, and Stan Sclaroff

***Abstract*—In deep CNN based models for semantic segmentation, high accuracy relies on rich spatial context (large receptive fields) and fine spatial details (high resolution), both of which incur high computational costs. In this paper, we propose a novel architecture that addresses both challenges and achieves state-of-the-art performance for semantic segmentation of high-resolution images and videos in real-time. The proposed architecture relies on our fast spatial attention, which is a simple yet efficient modification of the popular self-attention mechanism and captures the same rich spatial context at a small fraction of the computational cost, by changing the order of operations. Moreover, to efficiently process high-resolution input, we apply an additional spatial reduction to intermediate feature stages of the network with minimal loss in accuracy thanks to the use of the fast attention module to fuse features. We validate our method with a series of experiments, and show that results on multiple datasets demonstrate superior performance with better accuracy and speed compared to existing approaches for real-time semantic segmentation. On Cityscapes, our network achieves 74.4% mIoU at 72 FPS and 75.5% mIoU at 58 FPS on a single Titan X GPU, which is ∼50% faster than the state-of-the-art while retaining the same accuracy.**

***Index Terms*—Semantic Segmentation, Real-time Speed, Fast Attention.**

I. Introduction

SEMANTIC segmentation is a fundamental task in robotic sensing and computer vision, aiming to predict dense semantic labels for given images [1]–[8]. With the ability to extract scene contexts such as category, location, and shape of objects and stuff (everything else), semantic segmentation can be widely applied to many important applications like robots [9]–[11] and autonomous driving [12]–[14]. For many of these applications, *efficiency* is critical, especially in real-time (≥30FPS) scenarios. To achieve high accuracy semantic segmentation, previous methods rely on features enhanced with rich contextual cues [15]–[19] and high-resolution spatial details [20], [21]. However, rich contextual cues are typically captured via very deep networks with sizable receptive fields [15]–[18] that require high computational costs; and detailed spatial information demand for inputs of high resolution [20], [21], which incur high FLOPs during inference.

Recent efforts have been made to accelerate models for real-time applications [5], [20]–[25]. These efforts can be roughly grouped into two types. The first strategy is to adopt compact and shallow model architectures [20], [21], [23], [26]. However, this approach may decrease the model capacity and limit the size of the receptive field for features, therefore decreasing the model's discriminative ability. Another technique is to restrict the input to be low-resolution [23], [24], [26]. Though greatly decreasing the computational complexity, low-resolution images may lose important details like object boundaries or small objects. As a result, both types of methods sacrifice effectiveness for speed, limiting their practical applicability.

In this work, we address these challenges by proposing the Fast Attention Network (FANet) for real-time semantic segmentation. To capture rich spatial contextual information, we introduce an efficient fast attention module. The original self-attention mechanism has been shown to be beneficial for various vision tasks [27], [28] due to its ability to capture non-local context from the input feature maps. However, given c channels, the original self-attention [27], [28] has a computational complexity of $\mathcal{O}(n^2c)$, which is quadratic with respect to the feature's spatial size $n = height \times width$. In the task of semantic segmentation, where high-resolution feature maps are required, this is costly and limits the model's efficiency and applications to real-time scenarios. Instead, in our fast attention module, we replace the Softmax normalization used in self-attention with cosine similarity, thus converting the computation process to a series of matrix multiplication upon which the matrix-multiplication associativity can be applied to reduce the computational complexity to the linear $\mathcal{O}(nc^2)$, without loss of spatial information. The proposed fast attention is $\frac{n}{c}$ times more efficient than the standard self-attention, given $n \gg c$ in semantic segmentation (e.g. n=128×256 and c=512)
.

FANet works by first extracting different stages of feature maps, which are then enhanced by fast attention modules and finally merged from deep to shallow stages in a cascaded way for class label prediction. Moreover, to process high-resolution inputs at real-time speed, we apply additional spatial reduction into FANet. Rather than directly down-scaling the input images, which loses spatial details, we opt for down-sampling

Manuscript received August 25, 2020; Revised October 25, 2020; Accepted November 13, 2020. This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by DARPA and NSF Award No.1928477, and a gift funding from Adobe Research.

Ping Hu, Kate Saenko, and Stan Sclaroff are with Department of Computer Science, Boston University, Boston, MA 02215 USA. Kate Saenko is also with MIT-IBM Watson AI Lab, Cambridge, MA 02142 USA (e-mail: pinghu@bu.edu, saenko@bu.edu, sclaroff@bu.edu).

Federico Perazzi is with Facebook, Menlo Park, CA 94025 USA (e-mail: fperazzi@fb.com).

Fabian Caba Heilbron, Oliver Wang, and Zhe Lin are with Adobe, San Jose, CA 95110 USA (e-mail: caba@adobe.com, owang@adobe.com, zlin@adobe.com).

Digital Object Identifier (DOI): see top of this page.

Digital Object Identifier 10.1109/LRA.2020.3039744

intermediate feature maps. This strategy not only reduces computations but also enables lower layers to learn to extract features from high-resolution spatial details, enhancing FANet effectiveness. As a result, with very low computational cost, FANet makes use of both rich contextual information and full-resolution spatial details. We conduct extensive experiments to validate our proposed approach, and the results on multiple datasets demonstrate that FANet can achieve the fastest speed with state-of-the-art accuracy when compared to previous approaches for real-time semantic segmentation. Furthermore, in pursuit of better performance for video streams, we generalize the fast attention module to spatial-temporal contexts, and show (in Sec. 4) that this has the same computational cost as the single-frame model and does not increase with the length of the temporal range. This allows us to add rich spatial-temporal context to video semantic segmentation while avoiding an increase in computation.

In summary, we contribute the following: (1)We introduce the fast attention module for non-local context aggregation for efficient semantic segmentation, and further generalize it to a spatial-temporal version for video semantic segmentation. (2) We empirically show that applying extra spatial reduction to intermediate feature stages of the network effectively decreases computational costs while enhancing the model rich spatial details. (3) We present a Fast Attention Network for real-time semantic segmentation of images and videos with state-of-the-art accuracy and much higher efficiency over previous approaches.

II. Related Work

Extracting rich context information is key for high-quality semantic segmentation. To achieve this goal, different types of architecture design have been proposed [23], [29]–[33]. The first type of methods focus on enhancing the representation ability of the deep features [18], [31], [34]. Dilated convolutions [35], [36] are proposed to enlarge receptive field without shrinking spatial resolution. DeepLab [16] and PSP-Net [15] capture multi-scale spatial context from deep layers. BiseNet [23], [29] adopts multiple branches to process input of different scales, in order for the balance between effectiveness and computational cost. Another effective way for extracting spatial context is adopting encoder-decoder architecture [37]–[39]. Early works like SegNet [40] and U-net [41] adopt the symmetric structures for encoder and decoder. Deeplab-v3+ [42] integrates dilated convolution and spatial pyramid pooling into the architecture. To further enrich output features with high-level context, both ShelfNet [43] and DFANet [44] apply multiple encoder-decoder branch pairs and extend with more interconnected encoding streams.

Recently, self-attention [27], [28] mechanism has emerged as a very effective module for capturing long-range context in semantic segmentation. Fu *et al.* [18] apply self-attention mechanism for both spatial domain and channel domain. He *et al.* [45] combine self-attention model with ASPP [16] to boost performance with multi-scale context. However, self-attention models may suffer from low efficiency due to its intensive computational cost. To relieve this, Zhu *et al.* [46] propose to sample sparse anchor pixel locations for saving computation. Huang *et al.* [47] only consider the pixels on the same column and row. Although these methods reduce computation, they all take an approximation of the self-attention model and only partially collect the spatial information. In contrast, our fast attention does not only greatly save computation, but also capture full information from the feature map without loss of spatial information.

We also notice that there are several works on bilinear feature pooling [48], [49] that are related to our fast attention. Yet, our work differentiates from them in three aspects. (1) Methods [48], [49] approximate the affinity between pixels, while our fast attention is derived in a strictly equivalent form to built accurate affinity. (2) Unlike methods [48], [49] that focus on recognition tasks, our fast attention effectively tackles the dense semantic segmentation task. (3) As we will show later, in contrast to methods [48], [49], our fast attention allows for very efficient feature reuse in the video scenario, which can benefit video semantic segmentation with extra temporal context without increasing computation.

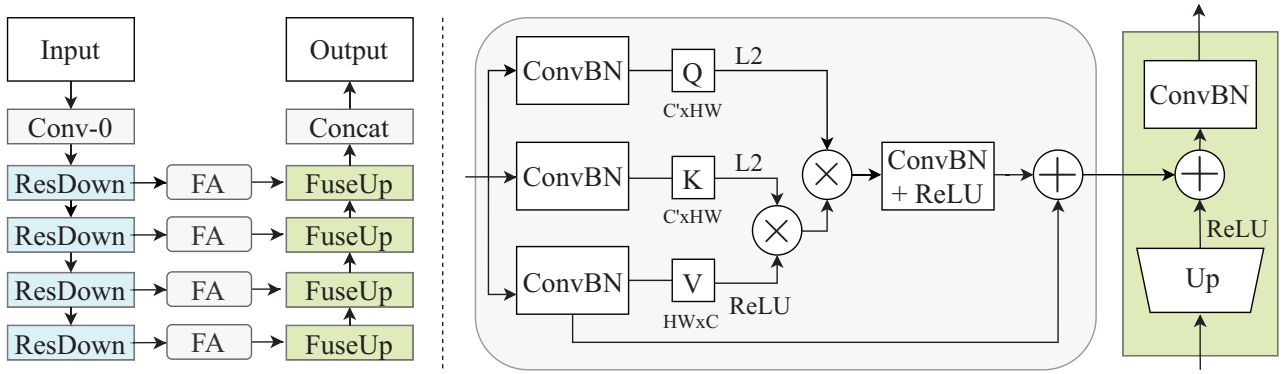
Existing methods for tackling video semantic segmentation can be grouped into two types. The first one [50]–[57] takes advantage of the redundant information in video frames, and reduce computation by reusing the high-level feature computed at keyframes. These methods run very efficiently, while always struggling with the spatial misalignment between frames, which leads to a decreased accuracy. Differently, the other type of methods ignore the redundancy and focus on capturing the temporal context information from neighboring frames for better effectiveness [58]–[60], which, however, incurs extra computation to sharply decrease the efficiency. In contrast to these methods, our FANet can be easily extended to also aggregate temporal context and allow for efficient feature reuse, achieving both high effectiveness and efficiency.

III. Fast Attention Network

In this section, we describe the Fast Attention Network (FANet) for real-time image semantic segmentation. We start by presenting the fast attention module and analyzing its computational advantages over original self-attention. Then we introduce the architecture of FANet. Last, we show that extra spatial reduction at intermediate feature stages of the model enables us to extract rich spatial details from high-resolution inputs while keeping a low computational cost.

A. Fast Attention Module

The self-attention module [27], [28] aims to capture non-local contextual information for each pixel location as a weighted sum of features at all positions in the feature map. Given a flattened input feature map $X \in \mathbb{R}^{n\times c}$ where c is the channel size and $n = height \times width$ is the spatial size, the self-attention model [27], [28] applies 1×1 convolutions to encode the feature maps into a ***Value*** map $V \in \mathbb{R}^{n\times c}$ that contains the semantic information of each pixel, a ***Query*** map $Q \in \mathbb{R}^{n\times c'}$ together with a ***Key*** map $K \in \mathbb{R}^{n\times c'}$ that are used to build correlations between pixel positions. Then the self-attention is calculated as $Y = f(Q, K) \cdot V$ where



a) Network architecture b) Fast Attention c) FuseUp

Fig. 1. (a) Architecture of Fast Attention Network (FANet). (b) Structure of Fast Attention (FA). (c) Structure of "FuseUp" module. Distinct from channel attention(CA) which only aggregates feature along the channel dimension for each pixel independently, our Fast Attention aggregates contextual information over the spatial domain thus achieving better effectiveness

$f(\cdot,\cdot) : \mathbb{R}^{n\times c'} \times \mathbb{R}^{n\times c'} \rightarrow \mathbb{R}^{n\times n}$ is the ***Affinity*** operation modeling the pairwise relations between all spatial locations. The *Softmax* function is typically used to model affinity $f(\cdot,\cdot)$, resulting in the popular self-attention response [46], [47], [61],

$$Y = Softmax(Q \cdot K^{\top}) \cdot V \tag{1}$$

Due to the existence of the normalization term in the *Softmax* function, the computation of Eq. (1) needs first compute the inner matrix multiplication $Q \cdot K^{\top}$, and then the one outside. This results in a computational complexity of $\mathcal{O}(n^2c)$. In semantic segmentation, feature maps have high spatial resolution. As complexity is quadratic with respect to the spatial size, this incurs high computational and memory costs, thus limiting the applications to scenarios especially those requiring real-time speed.

We tackle this challenge by first removing the Softmax affinity. As indicated in [27], there are a number of other affinity functions possible that can be used instead. One example, the dot product affinity can be computed simply as: $f(Q,K) = Q \cdot K^{\top}$. However, directly adopting the dot product may lead to affinity with unbounded values, and can be arbitrarily large. To avoid this, we instead use normalized cosine similarity for affinity computation,

$$Y = \frac{1}{n}(\hat{Q} \cdot \hat{K}^{\top}) \cdot V \tag{2}$$

where $\hat{Q}$ and $\hat{K}$ are the results of Q and K after L2-normalization along the channel dimension. Unlike Eq. 1, we observe that Eq. 2 can be represented as a series of matrix multiplications, which means that we can apply standard matrix-multiplication associativity to change the order of computation to achieve our fast attention as follows,

$$Y = \frac{1}{n}\hat{Q} \cdot (\hat{K}^{\top} \cdot V) \tag{3}$$

where $n = height \times width$ is the spatial size, and $\hat{K}^{\top} \cdot V$ is computed first.

Without loss of generality, this fast attention module can be computed with a computational complexity of $\mathcal{O}(nc^2)$, which is only about $\frac{c}{n}$ of the computational requirement of Eq. 1 (note that n is typically much larger than c in semantic segmentation). An illustration of fast attention module is shown in Fig. 1 (b). We notice that Channel Attention (CA) [18] has similar computation to our FA, yet there is a critical difference between them. CA only aggregates feature along the channel dimension thus computing a c-by-c pairwise affinity matrix for channels at first, and applies it to collect information from other channels instead of pixels. In contrast, our FA captures spatial context by building the n-by-n pairwise affinity matrix for pixels at first, and then applies it to collect features from other pixel locations. Yet since we break the limitation of softmax operation with normalized cosine similarity, we are able to apply the matrix-multiplication associativity to change computation order and achieve effective spatial attention with reduced computational costs. In the experiments, we quantitatively demonstrate that our fast attention outperforms channel attention.

B. Network Architecture

We describe our architecture for image semantic segmentation Fig 1 (a). The network is an encoder-decoder architecture with three components: encoder (left), context aggregation (middle), and decoder (right). We use a light-weight backbone (ResNet18 [62] without last fully connected layers) as the encoder to extract features from the input at different semantic levels. Given an input with resolution $h \times w$, the first res-block ("Res-1") in the encoder produces feature maps of $\frac{h}{4} \times \frac{w}{4}$ resolution. The other blocks sequentially output feature maps with resolution downsampled by a factor of 2. Our network applies the fast attention modules at each stage. As shown in Fig. 1 (b), the fast attention module is composed of three 1×1 convolutional layers for embedding the input features to be *Query*, *Key*, and *Value* maps respectively. When generating the *Query* and *Key*, we remove the ReLU layer, which only output positive values, to allow for a wider range of correlation between pixels. The L2-normalization along the channel dimension makes sure the affinity is between -1 to +1. After the feature pyramid is processed by the fast attention modules, the decoder gradually merges and upsamples the features in a sequential fashion from deep feature maps to

shallow ones. To enhance the decoded features with a high-level context, we further connect the middle features via a skip connection. An output with $\frac{h}{4} \times \frac{w}{4}$ resolution is predicted based on the enhanced feature output by the decoder.

C. Extra Spatial Reduction for Real-time Speed

Being able to generate semantic segmentation for high resolution inputs efficiently is challenging. Typically, high-resolution inputs provide rich spatial details that help achieve better accuracy, but dramatically reduce efficiency [15]–[17], [37], [42]. On the other hand, using smaller input resolution saves computational costs, but generates worse results due to the loss of spatial details [23], [24], [26].

To alleviate this, we adopt a simple yet effective strategy, which is to apply additional down-sample operations to the intermediate feature stages of the network rather than directly down-sampling the input images. We conduct an additional experiment where we use different types of spatial reduction operations, such as pooling and strided convolution at different feature stages, and evaluate how this impacts the resulting quality and speed trade-off. When applying an extra spatial reduction operator to our model, a similar up-sampling operation is added to the same stage of the decoder to keep the output resolution. We select the best choice of these, which we show in Section IV-C, not only reduces computation for upper layers, but also allows lower layers to learn to extract rich spatial details from high-resolution inputs and enhance performance. Thus allowing for both real-time efficiency and effectiveness with full-resolution input.

D. Extending to Video Semantic Segmentation

In many real-world applications of semantic segmentation, such as self-driving and robotics, video streams are the natural input for vision systems to understand the physical world. Nevertheless, most existing approaches for semantic segmentation focus on processing static images, and pay less attention to video data. In addition to spatial context from individual frames, video sequences also contain important temporal context derived from dynamics in the camera and scene. To take advantage of such temporal context for better accuracy, in this section we extend our fast attention module to spatial-temporal contexts, and show that it improves video semantic segmentation without increasing computational costs.

Given $\{Q_T, K_T, V_T\}$ extracted from the target frame T, and $\{Q_{T-i}, K_{T-i}, V_{T-i}\}$ with $i \in \{1, 2, .., t-1\}$ from the previous $t-1$ frames respectively, the spatial-temporal context within such a t-frame window can be aggregated via the traditional self-attention [27] as,

$$Y_T = \sum_{i=0}^{t-1} f(Q_T, K_{T-i}) \cdot V_{T-i}. \tag{4}$$

This has a computational complexity of $\mathcal{O}(tn^2c)$, t times higher than the single-frame spatial attention in Eq. 1.

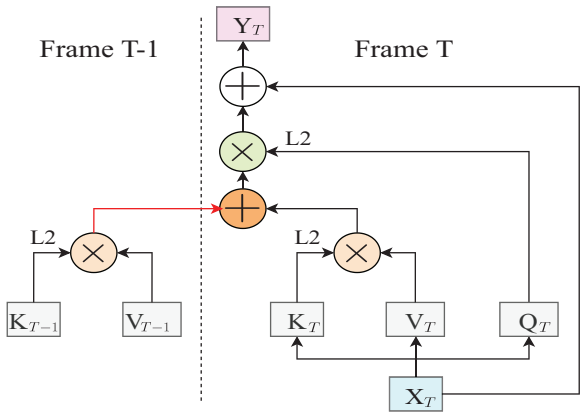


Fig. 2. Visualization of our fast attention for spatial-temporal context aggregation (t=2). The red arrows indicate the feature stored and reused by future frames.

By replacing the original self-attention with our fast attention, the spatial-temporal context for the target frame T can be computed as

$$Y_T = \sum_{i=0}^{t-1} \frac{1}{n} \hat{Q}_T \cdot (\hat{K}_{T-i}^\top \cdot V_{T-i}) \tag{5}$$

$$= \frac{1}{n} \hat{Q}_T \cdot (\hat{K}_T^\top \cdot V_T + \sum_{i=1}^{t-1} \hat{K}_{T-i}^\top \cdot V_{T-i}) \tag{6}$$

where n is the spatial size, $\hat{Q}$ and $\hat{K}$ indicate the L2-normalized Q and K respectively. At time step T, the results for $\hat{K}_{T-i}^\top \cdot V_{T-i}$ with $i \in \{1, 2, ..., t-1\}$ have already been computed and simply can be reused. We can see in Eq. 6, that we only need to compute and store the term $\hat{K}_T^\top \cdot V_T$, add it to those of the previous frames' (this matrix addition's cost is negligible), and multiply it by $\hat{Q}_T$. Therefore, given a t-frame window our spatial-temporal fast attention has a computational complexity of $\mathcal{O}\left(nc^2\right)$, which is as efficient as the single-frame fast attention, and free of t. Therefore, our fast attention is able to aggregate spatial-temporal context without increasing computational cost. An illustration of the spatial-temporal FA is shown in Fig. 2. By replacing the fast attention modules with this spatial-temporal version, FANet is able to sequentially segment video frames with feature enhanced with spatial-temporal context.

IV. Experiments

A. Datasets and Evaluation

Cityscapes [12] is a large benchmark containing 19 semantic classes for urban scene understanding with 2975/500/1525 scenes for train/validation/test respectively. *CamVid* [63] is another street-view dataset with 11 classes. The annotated frames are divided into 367/101/233 for training/validation/testing. *COCO-Stuff* [2] contains both diverse indoor and outdoor scenes for semantic segmentation. This dataset has 9,000 densely annotated images for training and 1,000 for testing. Following previous work [20], we adopt the resolution 640×640 and evaluate on 182 classes including 91 for things and 91 for stuff. We evaluate our method on image semantic

TABLE I
GFLOPS FOR NON-LOCAL MODULE [27] AND OUR FAST ATTENTION MODULE WITH C×128×256 FEATURES AS INPUT.

C=	32	64	128	256	512	1024
Self-Att. [28]	68	103	173	313	602	1203
Ours	0.2	0.6	1.7	5	19	73

segmentation for all four datasets, and additionally evaluate on Cityscapes for video semantic segmentation. The mIoU (mean Intersection over Union) is reported for evaluation.

B. Implementation Details

We use ResNet-18/34 [62] pretrained on Imagenet as the encoder in FANet, and randomly initialize parameters in fast attention modules as well as the decoder network. We train using mini-batch stochastic gradient descent (SGD) with batch-size 16, weight decay 5e^{-4}, and momentum 0.9. The learning rate is initialized as 1e^{-2}, and multiplied with $(1-\frac{iter}{max\_iter})^{0.9}$ after each iteration. We apply data augmentation including random horizontal flipping, random scaling (from 0.75 to 2), random cropping and color jittering in the training process. During testing, we input images at full resolution, and resize the output to the original size for calculating the accuracy. In not specified, all the evaluation experiments are conducted with batchsize 1 on a single Titan X GPU.

C. Method Analysis

Fast Attention. We first show the advantage in efficiency due to our fast attention. In Table I, we compare GFLOPs between a single original self-attention module and our fast attention module. Note that our fast attention runs significantly more efficiently for different size input features with more than 94% less computation.

Then, we compare our fast attention to the original self-attention module [28] in our FANet. As shown in Table II, compared to the model without attention (denoted as "w/o Att."), applying the original self-attention module to the network increases mIoU by 2.4% while decreasing the speed from 83 fps to 8 fps. In contrast to the original self-attention module, our fast attention (denoted as "FA with L2-norm") can achieve only slightly worse quality performance while greatly saving the computation cost. To further analyze our cosine-similarity based fast attention, we also train without the L2-normalization for both *Query* and *Key* features (denoted as "FA w/o L2-norm") and achieve 74.1% mIoU on the Cityscapes val, which is lower than 75.0% mIoU of our full model. This validates the necessity of cosine similarity to ensure bounded values for affinity computation. In Table IV, we also show per-class performance for different settings. As shown in the last two rows, by applying the L2-norm, we got performance improvement for most of the categories. In particular for object classes like "Traffic light", "Person", "Trunk", "Bus", and "Train", L2-Norm helps to achieve more than one percent improvement. This may be because that in Fast Attention we apply the spatial affinity matrix to collect and accumulate features from pixels in a linear way, and without L2-Norm the affinity values can be unconstrained. Thus classes of small areas will be heavily affected by other classes. Applying L2-Norm can relive this by output pixel-pair affinity with constrained magnitude, thus achieving more balanced relationships among classes. Yet Fast Attention maybe still not optimal compared to the original self-attention model, which adopts softmax-based normalization to attenuate the accumulation of information from unrelated classes and achieves more balanced pair-wise relationships among classes. As revealed in the first row of Table IV, replacing L2-Normalization with Softmax operation (denoted as "self-att.") further improves the performance for most of the small-area classes (e.g. "Pole", "Traffic light", "Traffic sign", "Trunk" etc.).

TABLE II
PERFORMANCE ON CITYSCAPES FOR DIFFERENT ATTENTION MECHANISMS FOR FANET-18. "FA" DENOTES OUR FAST ATTENTION.

	mIoU(%)	Speed(fps)	GFLOPs
w/o Att.	72.7	83	48
Self-Att. [28]	75.1	8	121
Channel-Att. [18]	74.6	70	51
FA w/o L2-norm	74.1	72	49
FA with L2-norm	75.0	72	49

TABLE III
PERFORMANCE ON CITYSCAPES *val* FOR DIFFERENT CHANNEL NUMBERS (c') IN FAST ATTENTION IN FANET-18.

#Channel (c')	8	16	32	64	128
mIoU (%)	73.5	74.6	75.0	75.0	75.0
Speed (fps)	74	74	72	69	65

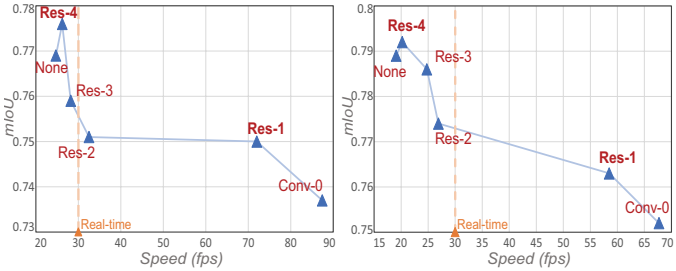
In Table III, we analyze the influence of channel numbers for *Key* and *Query* maps in our fast attention module. As we can see, too few channels such as c'=8 or c'=16 saves computation, but limits the representing capacity of the feature and leading to lower accuracy. On the other hand, when increasing the channel number from 32 to 128, the accuracy becomes stable, yet the speed drops. As a result, we adopt c'=32 in our experiments.

Spatial Reduction Next, we analyze the effect of applying the extra spatial reduction at different feature stages of FANet. The effects of additionally down-sampling different blocks are presented in Fig. 3. As we can see, down-sampling before "Conv-0" (down-scaling the input image), reduces the computation of all the subsequent layers, but loses critical spatial details which reduces the result quality. "Res-1" indicates that we reduce the spatial size at the stage of the first Res-block in FANet. Extra spatial reduction at higher stages like "Res-2", "Res-3", and "Res-4" do not increase speed significantly. Interestingly enough, we observe that applying down-sampling to "Res-4" actually performs *better* than "None", no additional downsampling, as shown in Fig 3. We hypothesize that this may be because that the block "Res-4" process high-level features, and adding extra down-sampling helps to enlarge the receptive field thus benefiting with rich contextual information. Based on these observations and with an aim of real-time semantic segmentation, we choose to apply extra down-sampling to "Res-1" and denote the model as FANet-18/34 based on the ResNet encoder used.

In additional to doubling the stride of convolutional layers to

TABLE IV
PER-CLASS PERFORMANCE ON CITYSCAPES VAL SET.

	mIoU(%)	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic light	Traffic sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle
Self-Att. [28]	**75.1**	97.7	82.1	**91.6**	**56.4**	**55.7**	**60.9**	**66.8**	**75.7**	91.6	61.4	**94.7**	78.1	56.7	**94.3**	**77.8**	82.9	**76.4**	**51.8**	73.5
FA w/o L2-norm	74.1	97.2	82.8	91.5	55.3	54.4	59.1	64.2	74.3	91.5	60.2	94.5	77.3	58.0	94.1	75.4	78.8	73.5	51.1	73.3
FA with L2-norm	75.0	**97.9**	**83.3**	**91.6**	55.5	55.1	60.3	66.2	74.9	**91.7**	**61.8**	**94.7**	**78.5**	**58.1**	94.1	76.8	**85.1**	74.5	50.7	**73.9**



(a) ResNet-18 as encoder (b) ResNet-34 as encoder

Fig. 3. Accuracy and speed analysis on Cityscapes *val* for adding an additional down-sampling operation (rate=2) to different stages of the encoder in FANet. "Conv-0" means to directly down-sample the input image. "Res-n" indicates double the stride of the first Conv layer in the n-th Res-block. "None" means no additional down-sampling operation is applied.

achieve 75.0% mIoU, we also experiment with other forms of down-sampling including average pooling (72.9% mIoU) and max pooling (74.2% mIoU). Enlarging stride for Conv layers performs the best. This may be because that stride convolution helps to capture more spatial details while keeping sizable receptive fields.

Runtime on different devices. We test speed of the proposed FANet on several popular Nvidia GPU devices including Tesla V100, RTX 2080 Ti, as well as the embedded system Jeston Nano (4GB). As shown in Table V, FANet achieves 121 fps on V100 and 87 fps on 2080Ti with a high resolution 1024×2048. And on Jetson Nano, FANet is still able to achieve real-time speed at a resolution 512×1024. By adopting smaller input resolutions, FANet can achieve even faster speeds. This shows that FANet provides a wide spectrum of speed-resolution trade-off for users, and can be adapted to scenarios of different computation budgets.

TABLE V
INFERENCING SPEED OF FANET-18 ON DIFFERENT NVIDIA DEVICES WITH DIFFERENT INPUT RESOLUTIONS. TENSORRT IS ADOPTED TO DEPLOY FANET ON JETSON NANO.

Speed (fps)	256×512	512×512	512×1024	1024×1024	1024×2048
V100	236	234	230	196	121
2080Ti	164	163	160	153	87
Jetson Nano	85	52	31	18	7

D. Image Semantic Segmentation

We compare our final method to the recent state-of-the-art efficient approaches for real-time semantic segmentation. For fair comparisons, we evaluate the speed for different methods with PyTorch on the same Titan X GPU. Please check our supplementary material for details. On benchmarks including Cityscapes [12], CamVid [63], and COCO-Stuff [2], our FANet achieves accuracy comparable to the state-of-the-art with the highest efficiency.

Cityscapes. In Table VI, we present the speed-accuracy comparison. FANet-34 achieves mIoU 76.3% for validation and 75.5% for testing at a speed of 58 fps with full-resolution (1024×2048) inputs. To our best knowledge, FANet-34 outperforms existing approaches for real-time semantic segmentation with better speed and state-of-the-art accuracy. By adopting a lighter-weight encoder ResNet-18, our FANet-18 further accelerates the speed to 72 fps, which is nearly two times faster than the recent methods like ShelfNet [43] and SwiftNet [21]. Although the accuracy drops to mIoU 75.0% for validation and 74.4% for testing, it is still much better than many previous methods like SegNet [40] and ICNet [20], and comparable to the most recent methods like BiseNet [23] and ShelfNet [43]. The performance achieved by our models demonstrates the superior ability to better balance the accuracy and speed for real-time semantic segmentation. Some visual results of our method are shown in Fig. 4.

CamVid. Results for this dataset are reported in Table VII. As we can see, our FANet outperforms previous methods with better accuracy and much faster speed. Comparing to BiseNet [23], our FANet-18 runs 2× efficient, and our FANet-34 outperforms with 1.4% mIoU and a faster speed.

COCO-Stuff. To be consistent with previous methods [20], we evaluate at resolution 640×640 for segmenting the 182 categories. As shown in Table VII, for the general scene understanding task with this dataset, our FANet is also able to achieve satisfying accuracy with much faster speed than previous methods. Compared to the state-of-the-art real-time model ICNet [20], our FANet-34 achieves both better accuracy and speed, and FANet-18 can further accelerate the speed with a comparable mIoU.

E. Video Semantic Segmentation

In this part, we evaluate our method for video semantic segmentation on the challenging dataset Cityscapes [12]. Without significantly increasing the computational cost, our method can effectively capture both spatial and temporal contextual information to achieve better accuracy, and outperforms previous methods with much lower latency. In Table VIII, we compare our method with recent state-of-the-art approaches for video semantic segmentation. Compared

TABLE VI
IMAGE SEMANTIC SEGMENTATION PERFORMANCE COMPARISON WITH RECENT STATE-OF-THE-ART REAL-TIME METHODS ON CITYSCAPES DATASET. "GFLOPS@1MPX" SHOWS THE GFLOPS FOR INPUT WITH RESOLUTION 1M PIXELS.

Methods	mIoU(%)		Speed(fps)	GFLOPs	GFLOPs@1Mpx	Input Resolution
	val	test				
SegNet [40]	–	56.1	36	143	650	360×640
ICNet [20]	67.7	69.5	38	**30**	**15**	1024×2048
ERFNet [64]	71.5	69.7	48	103	206	512×1024
BiseNet [23]	74.8	74.7	47	67	59.5	768×1536
ShelfNet [43]	–	74.8	39	95	47.5	1024×2048
SwiftNet [21]	75.4	**75.5**	40	106	53	1024×2048
FANet-34	**76.3**	**75.5**	58	65	32.5	1024×2048
FANet-18	75.0	74.4	**72**	49	24.5	1024×2048

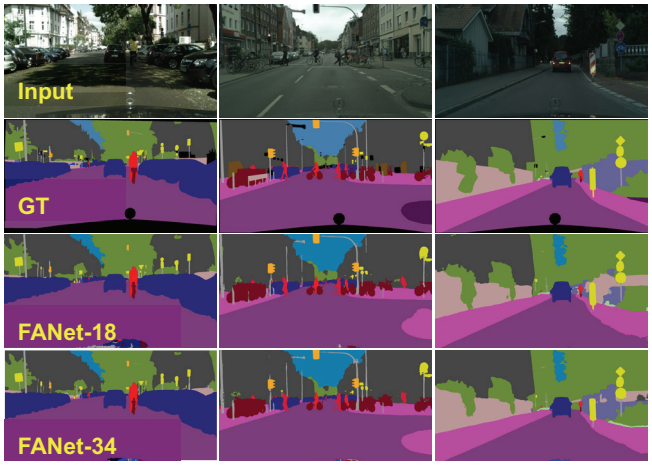


Fig. 4. Image semantic segmentation results on Cityscapes.

TABLE VII
IMAGE SEMANTIC SEGMENTATION PERFORMANCE ON CAMVID (LEFT) AND COCO-STUFF (RIGHT).

Method	mIoU (%)	Speed (fps)
SegNet [40]	55.6	12
ENet [26]	51.3	46
ICNet [20]	67.1	82
BiseNet [23]	68.7	75
FANet-34	**70.1**	121
FANet-18	69.0	**154**

Method	mIoU (%)	Speed (fps)
FCN [30]	22.7	9
DeepLab [16]	26.9	14
ICNet [20]	29.1	110
BiseNet [23]	25.6	113
FANet-34	**29.5**	142
FANet-18	27.8	**191**

to the image segmentation baseline models FANet18 and FANet34, both our spatial-temporal version FANet18+Temp and FANet34+Temp help to improve the accuracy at the same computational costs. We also see that most of the existing methods fail to achieve real-time speed ($\geq$ 30fps), apart from DVSNet which has much lower accuracy than ours. Methods like Clockwork [53] and DFF [54] save the overall computation while suffering from high latency due to the heavy computation at keyframes. PEARL [59] and Networp [58] achieves state-of-the-art accuracy at the cost of very low speed and high latency. In contrast, FANet18+Temp and FANet34+Temp achieve state-of-the-art accuracy with a faster speed. FANet18+Temp achieves more than 200× better efficiency than Netwarp [58]. FANet34+Temp outperforms PEARL [59] with 40× faster speed.

TABLE VIII
VIDEO SEMANTIC SEGMENTATION ON CITYSCAPES. "+TEMP" INDICATES FANET WITH SPATIAL-TEMPORAL ATTENTION (T=2). AVG RT IS THE AVERAGE PER-FRAME RUNNING TIME, AND MAXLATENCY IS THE MAXIMUM PER-FRAME RUNNING TIME.

Method	mIoU (%)↑	Speed (fps)↑	Avg RT (ms)↓	MaxLatency (ms)↓
DVSNet-fast [56]	63.2	30.4	33	-
Clockwork [53]	64.4	5.6	177	221
DFF [54]	69.2	5.7	175	644
Accel [52]	72.1	2.9	340	575
Low-Latency [51]	75.9	7.5	133	133
Netwarp [58]	**80.6**	0.33	3004	3004
FANet34	76.3	58	17	17
FANet34+Temp	76.7	58	17	17
FANet18	75.0	**72**	**14**	**14**
FANet18+Temp	75.5	**72**	**14**	**14**

V. CONCLUSION

We have proposed a novel Fast Attention Network for real-time semantic segmentation. In the network, we introduce fast attention to efficiently capture contextual information from feature maps. We further extend the fast attention to spatial-temporal context, and apply our models to achieve low-latency video semantic segmentation. To ensure high-resolution input with high efficiency, we also propose to apply spatial reduction to the intermediate feature stages. As a result, our model is enhanced with both rich contextual information and high-resolution details, while keeping a real-time speed. Extensive experiments on multiple datasets demonstrate the efficiency and effectiveness of our method.

REFERENCES

[1] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
[2] H. Caesar, J. Uijlings, and V. Ferrari, "COCO-Stuff: Thing and stuff classes in context," in *CVPR*, 2018.
[3] A. Milan, T. Pham, K. Vijay, D. Morrison, A. W. Tow, L. Liu, J. Erskine, R. Grinover, A. Gurman, T. Hunn *et al.*, "Semantic segmentation from limited training data," in *ICRA*, 2018.
[4] M. Tang, A. Djelouah, F. Perazzi, Y. Boykov, and C. Schroers, "Normalized cut loss for weakly-supervised CNN segmentation," in *CVPR*, 2018.
[5] V. Nekrasov, T. Dharmasiri, A. Spek, T. Drummond, C. Shen, and I. Reid, "Real-time joint semantic segmentation and depth estimation using asymmetric annotations," in *ICRA*, 2019.

[6] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *arXiv preprint arXiv:2001.05566*, 2020.

[7] P. Hu, S. Sclaroff, and K. Saenko, "Uncertainty-aware learning for zero-shot semantic segmentation," *NeurIPS*, 2020.

[8] X. Sun, R. Panda, R. Feris, and K. Saenko, "AdaShare: Learning what to share for efficient deep multi-task learning," *NeurIPS*, 2020.

[9] I. Kostavelis and A. Gasteratos, "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems*, 2015.

[10] E. Stenborg, C. Toft, and L. Hammarstrand, "Long-term visual localization using semantically segmented images," in *ICRA*, 2018.

[11] K. Wada, K. Okada, and M. Inaba, "Joint learning of instance and semantic segmentation for robotic pick-and-place with heavy occlusions in clutter," in *ICRA*, 2019.

[12] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, 2016.

[13] W. Zhou, S. Worrall, A. Zyner, and E. Nebot, "Automated process for incorporating drivable path into real-time semantic segmentation," in *ICRA*, 2018.

[14] A. Meyer, N. O. Salscheider, P. F. Orzechowski, and C. Stiller, "Deep semantic lane segmentation for mapless driving," in *IROS*, 2018.

[15] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017.

[16] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE TPAMI*, 2017.

[17] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, "Context encoding for semantic segmentation," in *CVPR*, 2018.

[18] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *CVPR*, 2019.

[19] W. Jiang, Y. Wu, L. Guan, and J. Zhao, "DFNet: Semantic segmentation on panoramic images with dynamic loss weights and residual fusion block," in *ICRA*, 2019.

[20] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *ECCV*, 2018.

[21] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, "In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images," in *CVPR*, 2019.

[22] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "ESPNet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *ECCV*, 2018.

[23] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *ECCV*, 2018.

[24] D. Marin, Z. He, P. Vajda, P. Chatterjee, S. Tsai, F. Yang, and Y. Boykov, "Efficient segmentation: Learning downsampling near semantic boundaries," in *ICCV*, 2019.

[25] J. Kuen, X. Kong, Z. Lin, G. Wang, J. Yin, S. See, and Y.-P. Tan, "Stochastic downsampling for cost-adjustable inference and improved regularization in convolutional networks," in *CVPR*, 2018.

[26] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[27] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *CVPR*, 2018.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[29] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "BiSeNet V2: Bilateral network with guided aggregation for real-time semantic segmentation," *arXiv preprint arXiv:2004.02147*, 2020.

[30] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[31] H. Ding, X. Jiang, B. Shuai, A. Qun Liu, and G. Wang, "Context contrasted feature and gated multi-scale aggregation for scene segmentation," in *CVPR*, 2018.

[32] P. Purkait, C. Zach, and I. Reid, "Seeing behind things: Extending semantic segmentation to occluded regions," in *IROS*, 2019.

[33] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang, "FasterSeg: Searching for faster real-time semantic segmentation," in *ICLR*, 2020.

[34] X. Li, Z. Liu, P. Luo, C. Change Loy, and X. Tang, "Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade," in *CVPR*, 2017.

[35] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *CVPR*, 2016.

[36] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *ICLR*, 2016.

[37] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters – improve semantic segmentation by global convolutional network," in *CVPR*, 2017.

[38] Z. Zhang, X. Zhang, C. Peng, X. Xue, and J. Sun, "ExFuse: Enhancing feature fusion for semantic segmentation," in *ECCV*, 2018.

[39] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *CVPR*, 2017.

[40] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. on PAMI*, 2017.

[41] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[42] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018.

[43] J. Zhuang, J. Yang, L. Gu, and N. Dvornek, "ShelfNet for fast semantic segmentation," in *ICCV Workshops*, 2019.

[44] H. Li, P. Xiong, H. Fan, and J. Sun, "DFANet: Deep feature aggregation for real-time semantic segmentation," in *CVPR*, 2019.

[45] J. He, Z. Deng, L. Zhou, Y. Wang, and Y. Qiao, "Adaptive pyramid context network for semantic segmentation," in *CVPR*, 2019.

[46] Z. Zhu, M. Xu, S. Bai, T. Huang, and X. Bai, "Asymmetric non-local neural networks for semantic segmentation," in *ICCV*, 2019.

[47] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "CCNet: Criss-cross attention for semantic segmentation," in *ICCV*, 2019.

[48] K. Yue, M. Sun, Y. Yuan, F. Zhou, E. Ding, and F. Xu, "Compact generalized non-local network," in *NIPS*, 2018.

[49] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, "Aˆ 2-Nets: Double attention networks," in *NIPS*, 2018.

[50] P. Hu, F. Caba, O. Wang, Z. Lin, S. Sclaroff, and F. Perazzi, "Temporally distributed networks for fast video semantic segmentation," in *CVPR*, 2020.

[51] Y. Li, J. Shi, and D. Lin, "Low-latency video semantic segmentation," in *CVPR*, 2018.

[52] S. Jain, X. Wang, and J. E. Gonzalez, "Accel: A corrective fusion network for efficient semantic segmentation on video," in *CVPR*, 2019.

[53] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell, "Clockwork convnets for video semantic segmentation," in *ECCV*, 2016.

[54] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," 2017.

[55] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *CVPR*, 2017.

[56] Y.-S. Xu, T.-J. Fu, H.-K. Yang, and C.-Y. Lee, "Dynamic video segmentation network," in *CVPR*, 2018.

[57] I. Krešo, J. Krapac, and S. Šegvić, "Efficient ladder-style densenets for semantic segmentation of large images," *IEEE Trans. on ITS*, 2020.

[58] R. Gadde, V. Jampani, and P. V. Gehler, "Semantic video cnns through representation warping," in *CVPR*, 2017.

[59] X. Jin, X. Li, H. Xiao, X. Shen, Z. Lin, J. Yang, Y. Chen, J. Dong, L. Liu, Z. Jie *et al.*, "Video scene parsing with predictive feature learning," in *ICCV*, 2017.

[60] D. Nilsson and C. Sminchisescu, "Semantic video segmentation by gated recurrent flow propagation," in *CVPR*, 2018.

[61] X. Li, Z. Zhong, J. Wu, Y. Yang, Z. Lin, and H. Liu, "Expectation-maximization attention networks for semantic segmentation," in *ICCV*, 2019.

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[63] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *ECCV*, 2008.

[64] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Efficient convnet for real-time semantic segmentation," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1789–1794.