Noname manuscript No.
(will be inserted by the editor)

SFNet: Faster and Accurate Semantic Segmentation via Semantic Flow

Xiangtai Li[1,2]* · **Jiangning Zhang**[3] · **Yibo Yang**[1] · **Guangliang Cheng**[2(✉)] · **Kuiyuan Yang**[5] · **Yunhai Tong**[1(✉)] · **Dacheng Tao**[4]

Received: date / Accepted: date

Abstract In this paper, we focus on exploring effective methods for faster and accurate semantic segmentation. A common practice to improve the performance is to attain high-resolution feature maps with strong semantic representation. Two strategies are widely used: atrous convolutions and feature pyramid fusion, while both are either computationally intensive or ineffective. Inspired by the Optical Flow for motion alignment between adjacent video frames, we propose a Flow Alignment Module (FAM) to learn *Semantic Flow* between feature maps of adjacent levels and broadcast high-level features to high-resolution features effectively and efficiently. Furthermore, integrating our FAM to a standard feature pyramid structure exhibits superior performance over other real-time methods, even on lightweight backbone networks, such as ResNet-18 and DFNet. Then to further speed up the inference procedure, we also present a novel Gated Dual Flow Alignment Module to directly align high-resolution feature maps and low-resolution feature maps where we term the improved version network as SFNet-Lite. Extensive experiments are conducted on several challenging datasets, where results show the effectiveness of both SFNet and SFNet-Lite. In particular, when using Cityscapes test set, the SFNet-Lite series achieve 80.1 mIoU while running at 60 FPS using ResNet-18 backbone and 78.8 mIoU while running at 120 FPS using STDC backbone on RTX-3090.

Moreover, we unify four challenging driving datasets (*i.e.*, Cityscapes, Mapillary, IDD, and BDD) into one large dataset, which we named Unified Driving Segmentation (UDS) dataset. It contains diverse domain and style information. We benchmark several representative works on UDS. Both SFNet and SFNet-Lite still achieve the best speed and accuracy trade-off on UDS, which serves as a strong baseline in such a challenging setting. The code and models are publicly available at https://github.com/lxtGH/SFSegNets.

Keywords Fast Semantic Segmentation · Real-time Processing · Sence Understanding · Auto-Driving

1 Introduction

Semantic segmentation is a fundamental vision task that aims to classify every pixel in the images correctly. It involves many real-world applications, including *auto-driving, robot navigation, and image editing.* The seminal work of Long *et. al.* [1] built a deep Fully Convolutional Network (FCN), which is mainly composed of convolutional layers to carve strong semantic representation. However, detailed object boundary information, which is also crucial to the performance, is usually missing due to the use of the down-sampling layers.

To alleviate this problem, state-of-the-art methods [2, 3,4,5] apply atrous convolutions [6] at the last several stages of their networks to yield feature maps with strong semantic representation while at the same time maintaining the high-resolution. Meanwhile, several state-of-the-art approaches [7,8,9] adopt multiscale feature representation to enhance final segmentation results.

* Work done at SenseTime Research, Beijing. ✉: Corresponding Authors.

Xiangtai Li (lxtpku@pku.edu.cn)
Yunhai Tong (yhtong@pku.edu.cn)
Guangliang Cheng (guangliangcheng2014@gmail.com)
1 National Key Laboratory of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University, Beijing, China.
2 SenseTime Research ; 3 Zhejiang University
4 School of Computer Science, Faculty of Engineering, The University of Sydney, Darlington, NSW 2008, Australia.
5 Xiaomi Car.

Recently, several methods [10,11,12] adopt vision transformer architectures and model the semantic segmentation as a per-segment prediction problem. In particular, they achieve stronger performance for the long-tailed datasets, including ADE-20k [13] and COCO-stuff [14] due to the stronger pre-trained models [15] and query-based mask representation [16].

Despite those methods achieving state-of-the-art results on various benchmarks, one fundamental problem is the real-time inference speed, particularly for high-resolution image inputs. Given that the FCN using ResNet-18 [17] as the backbone network has a frame rate of 57.2 FPS for a 1024×2048 image, after applying atrous convolutions [6] to the network as done in [2,3], the modified network *only has a frame rate of 8.7 FPS*. Moreover, under a single GTX 1080Ti GPU with no other ongoing programs, the previous state-of-the-art model PSPNet [2] has a frame rate of only 1.6 FPS for 1024×2048 input images. Consequently, this is problematic for many advanced real-world applications, such as self-driving cars and robot navigation, which desperately demand real-time online data processing.

In order to not only maintain detailed resolution information but also get features that exhibit strong semantic representation, another direction is to build FPN-like [18,19,20] models which leverage the lateral path to fuse feature maps in a top-down manner. In this way, the deep features of the last several layers strengthen the shallow features with high resolution, and therefore, the refined features are possible to satisfy the above two factors and are beneficial to the accuracy improvement. Such designs are mainly adopted by real-time semantic segmentation models. However, the accuracy of these methods [20,21,22,23] still needs improvement when compared to those networks that hold large feature maps in the last several stages. Is there a better solution for high accuracy and high-speed semantic segmentation? We suspect that the low accuracy problem arises from the ineffective propagation of semantics from deep layers to shallow layers, where the semantics are not well aligned across different stages.

To mitigate this issue, we propose explicitly learning the **Semantic Flow** between two network layers of different resolutions. Semantic Flow is inspired by optical flow, which is widely used in video processing task [24] to represent the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by relative motion. *In a flash of inspiration, we find the relationship between two feature maps of arbitrary resolutions from the same image can also be represented with the "motion" of every pixel from one feature map to the other one.* In this case, once precise Semantic Flow is obtained, the network is able to propagate semantic features with minimal information loss. It should be noted that Semantic Flow is different from optical flow, since Semantic Flow takes feature maps from different levels as input and assesses the discrepancy within them to find a suitable flow field that will give a dynamic indication about how to align these two feature maps effectively.

Based on the concept of Semantic Flow, we design a novel network module called Flow Alignment Module (FAM) to utilize Semantic Flow in semantic segmentation. Feature maps after FAM are embodied with both rich semantics and abundant spatial information. Because FAM can effectively transmit semantic information from deep to shallow layers through elementary operations, it shows superior efficacy in improving accuracy and keeping superior efficiency. Moreover, FAM is end-to-end trainable and can be plugged into any backbone network to improve the results with a minor computational overhead. For simplicity, we call the

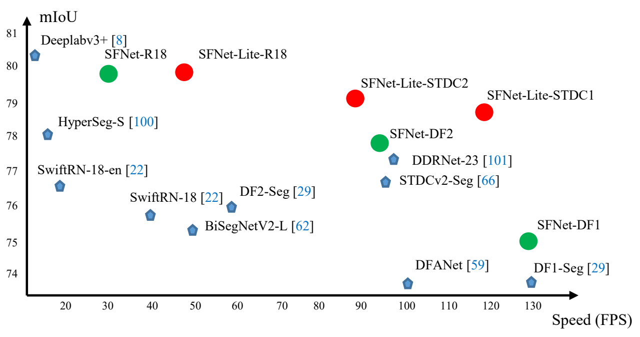

Fig. 1: Inference speed versus mIoU performance on test set of Cityscapes. Previous models are marked as blue points, and our models are shown in red and green points which achieve the best speed/accuracy trade-off. Note that our methods with ResNet-18 as backbone even achieve comparable accuracy with all accurate models at much faster speed. SFNet methods are the green nodes while SFNet-Lite methods are the red nodes.

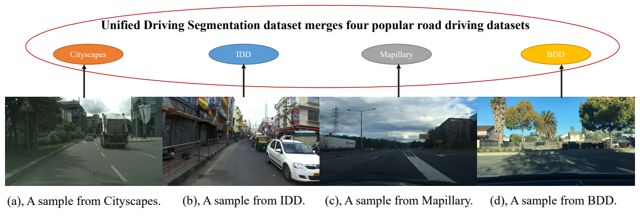
Unified Driving Segmentation dataset merges four popular road driving datasets
Cityscapes
IDD
Mapillary
BDD
(a), A sample from Cityscapes.
(b), A sample from IDD.
(c), A sample from Mapillary.
(d), A sample from BDD.

Fig. 2: Illustration of the merged Unified Driving Segmentation (UDS) benchmark. It contains four datasets including Cityscapes [25] (a), IDD [26](b), Mapillary [27](c) and BDD [28] (d). These datasets have *various styles and texture information*, which make the merged UDS dataset more challenging.

networks that all incorporate FAM but have different backbones as **SFNet**. As depicted in Fig. 1, SFNets with different backbone networks outperform competitors by a large margin at the same speed. In particular, our method adopting ResNet-18 as backbone achieves **79.8%** mIoU on the Cityscapes test server with a frame rate of **33 FPS**. When adopting DF2 [29] as the backbone, our method achieves 77.8% mIoU with 103 FPS and 74.5% mIoU with 134 FPS when equipped with the DF1 backbone [29]. The results are shown in Fig. 1 (green node).

The original SFNet [30] achieves satisfactory results on speed and accuracy trade-off, and several following works [31] generalize the idea of SFNet into other domains. However, the inference speed of SFNet still needs to be faster due to the multi-stage features involved. To speed up the SFNet and maintain accuracy at the same time, we propose a new version of SFNet, named SFNet-Lite. In particular, we design a new flow-aligned module named Gated Dual Flow Aligned Module (GD-FAM). Following FAM, GD-FAM takes two features as inputs and learns *two semantic flows* to refine both high-resolution and low-resolution features simultaneously. Meanwhile, we also generate a shared gate map to control the flow warping processing before the final addition dynamically. The newly proposed GD-FAM can be appended at the end of SFNet backbone *only once*, directly refining the highest and lowest resolution features. Such design avoids multiscale feature fusion and speeds up the SFNet by a large margin. We name our new version of SFNet as **SFNet-Lite**. Moreover, to keep the origin accuracy, we carry out extensive experiments on Cityscapes by introducing more balanced datasets training [5]. As a result, our SFNet-Lite with ResNet-18 backbone achieves **80.1** mIoU on Cityscapes test set but with the speed of **49 FPS** (**16 FPS** improvements with slightly better performance over original SFNet [30]). Moreover, when adopting with STDCv1 backbone, our method can achieve **78.7 mIoU** while running with the speed of **120 FPS.** The results are shown in Fig. 1 (red node).

Since various driving datasets [28, 26, 25] are from different domains, previous real-time semantic segmentation methods train different models on different datasets, which results in that the trained models are sensitive to trained domains and can not generalize well to unseen domain [32]. Recently, M-Seg propose a mixed dataset for multi-dataset semantic to achive one model for multiple dataset training and test. Motivated by above, we verify whether our SFNet series can be more effective in a unified dataset benchmark. Firstly, we benchmark our SFNet and SFNet-Lite on various driving datasets [28, 27, 26] in the experiment part. Secondly, we creat a challenging benchmark by mixing four challenging driving datasets, including Cityscapes, Mapillary, BDD, and IDD. We term our merged dataset Unified Driving Segmentation (UDS). As shown in Fig. 2, our goal is to train **a unified model** to perform semantic segmentation on various scenes. To the best of our knowledge, UDS is the largest public semantic segmentation dataset for the driving scene. In particular, we extract the typical semantic class as defined by Cityscapes and BDD with 19 class labels and merge several classes in Mapillary. We further benchmark representative works on our UDS. Our SFNet also achieves the best accuracy and speed trade-off, which *indicates the generalization ability of semantic flow.* In particular, using DFNet [29] as the backbone, our SFNet and SFNet-Lite achieve *7-9% mIoU* improvements on UDS. This indicates that our proposed FAM and GD-FAM are more practical to multiple-dataset training.

In summary, a preliminary version of this work was published in [30]. In this paper, we make the following significant extensions: (1) We introduce a new flow alignment module (GD-FAM) to increase the speed of SFNet while maintaining the original performance. Experiments show that this new design consistently outperforms our previous module with higher inference efficiency. (2) We conduct more comprehensive ablation studies to verify the proposed method, including quantitative improvements over baselines and visualization analysis. (3) We extend SFNet into Panoptic Segmentation, where we achieve 1.0%-1.5% PQ improvements over three strong baselines. (4) We further benchmark SFNet and several recent representative methods on two more challenging datasets, including Mapillary [27] and IDD [26]. Our SFNet series significantly improve over different baselines and achieve the best speed and accuracy trade-off. In particular, we propose a new setting for training a unified real-time semantic segmentation model by merging existing driving datasets (UDS). Our SFNet series also achieve the best accuracy and speed trade-off, which can be a solid baseline for mixed driving segmentation. We further prove the effectiveness of SFNet and SFNet-Lite on transformer architecture on the ADE20k dataset. Moreover, aided by the RobustNet [32], we further show the effectiveness of SFNet on domain generalization setting.

2 Related Work

Generic Semantic Segmentation. Current state-of-the-art methods on semantic segmentation are based on the FCN framework, which treats semantic segmentation as a dense pixel classification problem. Lots of methods focus on global context modeling with dilated backbone. Global average pooled features are concatenated into existing feature maps in [33]. In PSPNet [2], average pooled features of multiple window sizes, including global average pooling, are upsampled to the same size and concatenated together to enrich global information. The DeepLab variants [34,35,8] propose atrous or dilated convolutions and atrous spatial pyramid pooling (ASPP) to increase the effective receptive field. DenseASPP [36] improves on [37] by densely connecting convolutional layers with different dilation rates to further increase the receptive field of the network. In addition to concatenating global information into feature maps, multiplying global information into feature maps also shows better performance [38,39,40,41]. Moreover, several works adopt the self-attention design to encode the global information for the scene. Using non-local operator [42], impressive results are achieved in [43,44,4]. CCNet [45] models the long-range dependencies by considering its surrounding pixels on the criss-cross path via a recurrent way to save computation and memory cost. Meanwhile, several works [20,7,19, 46,47] adopt encode-decoder architecture to learn the multi-level feature representation. RefineNet [48] and DFN [41] adopted encoder-decoder structures that fuse information in low-level and high-level layers to make dense prediction results. Following such architecture design, GFFNet [9], CCLNet [49], and G-SCNN [50] use gates for feature fusion to avoid noise and feature redundancy. AlignSeg [51] proposes to refine the multiscale features via bottom-up design. IFA [52] proposes an implicit feature alignment function to refine the multiscale feature representation. In contrast, our method transmits semantic information top-down, focusing on real-time application. However, only some of these works can perform inference in real-time, which makes it hard to employ in practical applications.

Vision Transformer based Semantic Segmentation. Recently, transformer-based approaches [53,15,12, 54] replace the CNN backbones with vision transformers and achieve more robust results. Several works [12, 15,55,56] show that the vision transformer backbone leads to better results on long-tailed datasets due to the better feature representation and stronger pre-training on ImageNet classification. SETR [12] replaces the pixel level modeling with token-based modeling, while Segformer [55] proposes a new efficient backbone for segmentation. Moreover, several works [11,10,57] adopt Detection Transformer (DETR) [16] to treat per-pixel prediction as a per-mask prediction. In particular, Maskformer [10] treats the pixel-level dense prediction as a set prediction problem. However, all of these works still can not perform inference in real-time due to the huge computation cost.

Fast Semantic Segmentation. Fast (Real-time) semantic segmentation algorithms attract attention when demanding practical applications that need fast inference and response. Several works are designed for this setting. ICNet [58] uses multiscale images as input and a cascade network to be more efficient. DFANet [59] utilizes a light-weight backbone to speed up its network and proposes a cross-level feature aggregation to boost accuracy, while SwiftNet [22] uses lateral connections as the cost-effective solution to restore the prediction resolution while maintaining the speed. ICNet [58] reduces the high-resolution features into different scales to speed up the inference time. ESPNets [60,61] save computation by decomposing standard convolution into point-wise convolution and spatial pyramid of atrous convolutions. BiSeNets [62,63] introduce spatial path and semantic path to reduce computation. Recently, several methods [64,65,29] use AutoML techniques to search efficient architectures for scene parsing. Moreover, there are several works [66,67] using multi-branch architecture to improve the real-time segmentation results. However, these works result in poor segmentation results compared with those general methods on multiple benchmarks such as Cityscapes [25] and Mapillary [27]. Our previous work SFNet [30] achieves high accuracy via learning semantic flow between multiscale features while running in real-time. However, its inference speed is still limited since more features are involved. Moreover, the capacity of multiscale features needs to be better explored via stronger data augmentation and pre-training. Thus, simultaneous achievement of high speed and high accuracy is still challenging and of great importance for real-time application purposes.

Panoptic Segmentation. Earlier works [68,69,70,71, 72] are proposed to model both stuff segmentation and thing segmentation in one model with different task heads. Detection-based methods [73,68,74,75] usually represent things with the box prediction, while several bottom-up models [76,77] perform grouping instance via pixel-level affinity or center heat maps from semantic segmentation results. The former introduces the complex process, while the latter suffers from performance drops in complex scenarios. Recently, several works [11,57,10] propose directly obtaining segmentation masks without box supervision. However, all of these works ignore the speed issue. In the experiment, we further show that our method can also lead to better panoptic segmentation results.

Lightweight Architecture Design. Another critical research direction is to design more efficient backbones for the downstream tasks via various approaches [78,79, 80,66]. These methods focus on efficient representation learning with various network search approaches. Our work is orthogonal to those works, since we aim to design a lightweight and aligned segmentation head.

Multi-dataset Segmentation. MSeg [81] firstly proposes to merge most existing datasets in one unified taxonomy and train a unified segmentation model for variant scenes. Meanwhile, several following works [82,83] explore multi-dataset segmentation or detection. Compared with MSeg, our UDS dataset mainly focuses on the driving scene and has only 19 classes compared with more than 100 classes in MSeg. The input images are high-resolution and are used for auto-driving applications.

Domain Generalization in Segmentation. The goal domain generalization (DG) [84] methods assume that the model cannot access the target domain during training and aim to improve the generalization ability to perform well in an unseen target domain. DG is slightly different from multi-data segmentation. As for segmentation, several works [85,86,87,32] adopt synthetic data such as GTAV for training and real dataset such as cityscapes for testing. Recently, RobustNet [32] disentangles the domain-specific style and domain-invariant content encoded in higher-order statistics. Our method can also be applied in DG segmentation settings by combing RobustNet [32], where we also find significant improvements over various baselines.

3 Method

In this section, we will first provide some preliminary knowledge about real-time semantic segmentation and introduce the misalignment problem therein. Then, we propose the Flow Alignment Module (FAM) to resolve the misalignment issue by learning Semantic Flow and warping top-layer feature maps accordingly. We also present the design of SFNet. Next, we introduce the proposed SFNet-Lite and the improved GD-FAM to speed up SFNet. Finally, we describe the building process of our UDS dataset and several improvement details for SFNet-Lite training.

3.1 Preliminary

The task of scene parsing is to map a RGB image $\mathbf{X} \in \mathbb{R}^{H \times W \times 3}$ to a semantic map $\mathbf{Y} \in \mathbb{R}^{H \times W \times C}$ with the same spatial resolution $H \times W$, where C is the number of predefined semantic categories. Following the setting of FPN [18], the input image $\mathbf{X}$ is firstly mapped to a set of feature maps $\{\mathbf{F}_l\}_{l=2,\ldots,5}$ from each network stage, where $\mathbf{F}_l \in \mathbb{R}^{H_l \times W_l \times C_l}$ is a C_l-dimensional feature map defined on a spatial grid Ω_l with size of $H_l \times W_l, H_l = \frac{H}{2^l}, W_l = \frac{W}{2^l}$. The coarsest feature map $\mathbf{F}_5$ comes from the deepest layer with the strongest semantics. FCN-32s directly predicts upon $\mathbf{F}_5$ and achieves over-smoothed results without fine details. However, some improvements can be achieved by fusing predictions from lower levels [1]. FPN takes a step further to gradually fuse high-level feature maps with low-level feature maps in a top-down pathway through $2\times$ bilinear upsampling, which is originally proposed for object detection [18] and recently introduced for scene parsing [7, 19]. The whole FPN framework highly relies on upsampling operator to upsample the spatially smaller but semantically stronger feature map to be larger in spatial size. However, the bilinear upsampling recovers the resolution of downsampled feature maps by interpolating a set of uniformly sampled positions (i.e., it can only handle one kind of fixed and predefined misalignment), while the misalignment between feature maps caused by residual connection, repeated downsampling and upsampling operations, is far more complex. Therefore, position correspondence between feature maps needs to be explicitly and dynamically established to resolve their actual misalignment.

3.2 Original Flow Alignment Module and SFNet

Design Motivation. For more flexible and dynamic alignment, we thoroughly investigate the idea of optical flow, which is very effective and flexible for aligning two adjacent video frame features in the video processing task [89,24]. The idea of optical flow motivates us to design a *flow-based alignment module* (**FAM**) to align feature maps of two adjacent levels by predicting a flow field inside the network. We define such flow field as *Semantic Flow*, which is generated between different levels in a feature pyramid.

Module Details. FAM is constructed using the FPN framework, which involves compressing the feature map of each level into the same channel depth using two 1×1 convolution layers before passing it on to the next level. Given two adjacent feature maps $\mathbf{F}_l$ and $\mathbf{F}_{l-1}$ with the same channel number, we up-sample $\mathbf{F}_l$ to the same size as $\mathbf{F}_{l-1}$ via a bi-linear interpolation layer. Then, we concatenate them together and take the concatenated feature map as input for a subnetwork that contains two convolutional layers with the kernel size of 3×3. The output of the subnetwork is the prediction of the semantic flow field $\Delta_{l-1} \in \mathbb{R}^{H_{l-1} \times W_{l-1} \times 2}$. Mathematically, the aforementioned steps can be written as:

$$\Delta_{l-1} = \text{conv}_l(\text{cat}(\mathbf{F}_l, \mathbf{F}_{l-1})), \quad (1)$$

where cat$(\cdot)$ represents the concatenation operation and conv$_l(\cdot)$ is the 3×3 convolutional layer. Since our network adopts the strided convolutions, which could lead

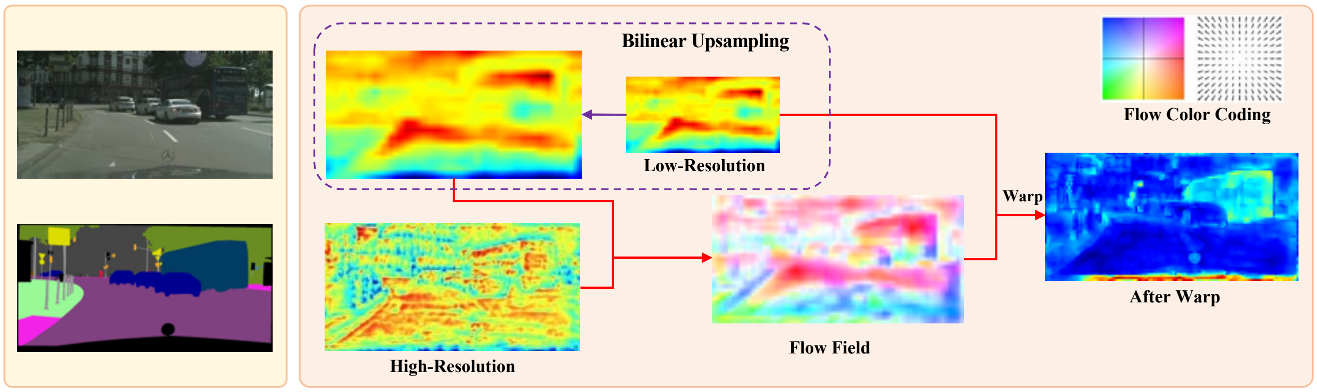

Fig. 3: Visualization of feature maps and semantic flow field in FAM. Feature maps are visualized by averaging along the channel dimension. Larger values are denoted by hot colors and vice versa. We use the color code proposed in [88] to visualize the Semantic Flow field. The orientation and magnitude of flow vectors are represented by hue and saturation, respectively. As shown in this figure, using our proposed semantic flow results in more structural feature representation.

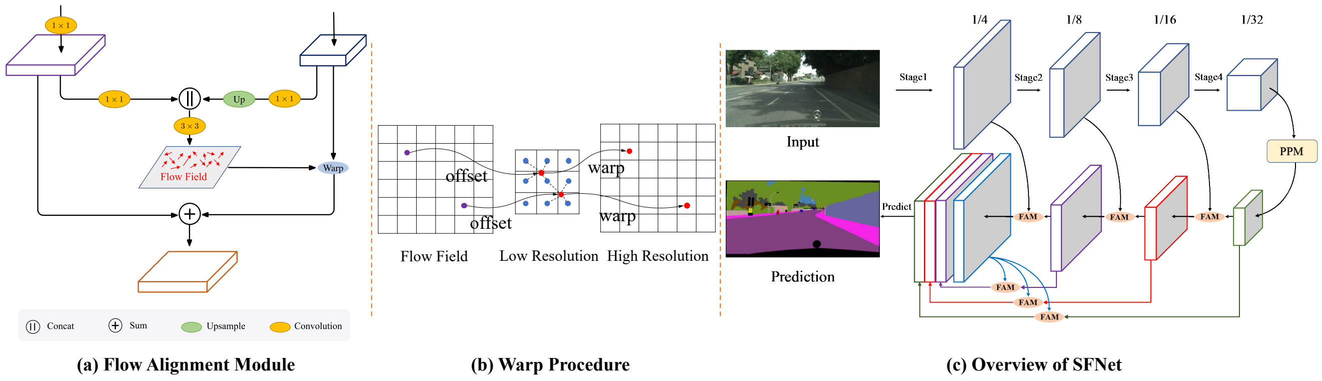

Fig. 4: (a) The details of Flow Alignment Module. We combine the transformed high-resolution feature map and low-resolution feature map to generate the semantic flow field, which is utilized to warp the low-resolution feature map to a high-resolution feature map. (b) Warp procedure of Flow Alignment Module. The value of the high-resolution feature map is the bilinear interpolation of the neighboring pixels in the low-resolution feature map, where the neighborhoods are defined according to the learned semantic flow field. (c) Overview of our proposed SFNet. ResNet-18 backbone with four stages is used for exemplar illustration. FAM: Flow Alignment Module. PPM: Pyramid Pooling Module [2]. Best view it in color and zoom in.

to very low resolution, for most cases, the respective field of the 3×3 convolution conv_l is sufficient to cover most large objects in that feature map. Note that, we discard the correlation layer proposed in FlowNet-C [90], where positional correspondence is calculated explicitly. Because there exists a huge semantic gap between higher-level layer and lower-level layer, explicit correspondence calculation on such features is difficult and tends to fail for offset prediction. Furthermore, including a correlation layer to address this issue would increase the computational cost substantially, which contradicts our objective of developing a fast and accurate network.

After having computed Δ_{l-1}, each position p_{l-1} on the spatial grid Ω_{l-1} is then mapped to a point p_l on the upper level l via a simple addition operation. Since there exists a resolution gap between features and flow field as shown in Figure 4(b), the warped grid and its offset should be halved as Equation 2,

$$p_l = \frac{p_{l-1} + \Delta_{l-1}(p_{l-1})}{2}. \tag{2}$$

We then use the differentiable bi-linear sampling mechanism proposed in the spatial transformer networks [91], which linearly interpolates the values of the 4-neighbors (top-left, top-right, bottom-left, and bottom-right) of p_l to approximate the final output of the FAM, denoted by $\widetilde{\mathbf{F}}_l(p_{l-1})$. Mathematically,

$$\widetilde{\mathbf{F}}_l(p_{l-1}) = \mathbf{F}_l(p_l) = \sum_{p \in \mathcal{N}(p_l)} w_p \mathbf{F}_l(p), \tag{3}$$

where $\mathcal{N}(p_l)$ represents neighbors of the warped points p_l in $\mathbf{F}_l$ and w_p denotes the bi-linear kernel weights estimated by the distance of warped grid. This warping procedure may look similar to the convolution operation of the deformable kernels in deformable convolution network (DCN) [92]. However, our method has a lot of noticeable difference from DCN. First, our predicted offset field incorporates both higher-level and lower-level features to *align the positions* between high-level and low-level feature maps, while the offset field of DCN moves the positions of the kernels according to the predicted location offsets in order to *possess larger and more adaptive respective fields*. Second, our module focuses on aligning features, while DCN works more like an attention mechanism that attends to the salient parts of the objects. More detailed comparison can be found in the experiment part.

On the whole, the proposed FAM module is light-weight and end-to-end trainable because it only contains one 3×3 convolution layer and one parameter-free warping operation in total. Besides these merits, it can be plugged into networks multiple times with only a minor extra computation cost overhead. Figure 4(a) gives the detailed settings of the proposed module, while Figure 4(b) shows the warping process. Figure 3 visualizes the feature maps of two adjacent levels, their learned semantic flow and the finally warped feature map. As shown in Figure 3, the warped feature is more structurally neat than the normal bi-linear upsampled feature and leads to more consistent representation of objects, such as the bus and car.

Figure 4(c) illustrates the whole network architecture, which contains a bottom-up pathway as the encoder and a top-down pathway as the decoder. While the encoder has a backbone network offering feature representations of different levels, the decoder can be seen as a FPN equipped with several FAMs.

Encoder Part. We choose standard networks pre-trained on ImageNet [93] for image classification as our backbone network by removing the last fully connected layer. Specifically, our experiments use and compare the ResNet series [17] and DF series [29]. All backbones consist of 4 stages with residual blocks. To achieve both computational efficiency and larger receptive fields, we include a convolutional layer with a stride of 2 as the first layer in each stage, which downsamples the feature map. We additionally adopt the Pyramid Pooling Module (PPM) [2] for its superior power to capture contextual information. In our setting, the output of PPM shares the same resolution as that of the last residual module. In this situation, we treat PPM and the last residual module together as the last stage for the upcoming FPN. Other modules like ASPP [35] can also be plugged into our network, which is also experimentally ablated in the experiment part.

Aligned FPN Decoder. Our SFNet decoder takes feature maps from the encoder and uses the aligned feature pyramid for final scene parsing. By replacing normal bi-linear up-sampling with FAM in the top-down pathway of FPN [18], $\{\mathbf{F}_l\}_{l=2}^4$ is refined to $\{\widetilde{\mathbf{F}}_l\}_{l=2}^4$, where top-level feature maps are aligned and fused into their bottom levels via element-wise addition and l represents the range of feature pyramid level. For scene parsing, $\{\widetilde{\mathbf{F}}_l\}_{l=2}^4 \cup \{\mathbf{F}_5\}$ are up-sampled to the same resolution (*i.e.*, 1/4 of the input image) and concatenated together for prediction. Considering there still exists misalignment during the previous step, we also replace these up-sampling operations with the proposed FAM. To be noted, we only verify the effectiveness of such design in ablation studies. Our final models for the real-time application do not contain such a replacement for better speed and accuracy trade-off.

3.3 Gated Dual Flow Alignment Module and SFNet-Lite

Motivation. Original SFNet adopts a multi-stage flow-based alignment process, it leads to a slower speed than several representative networks like BiSegNet [62,58]. Since the lightweight backbone design is not our main focus, we explore the **more compact decoder** with **only one** flow alignment module. Decreasing the number of FAM leads to inferior results (shown in experiment part, see Tab. 9(d)). To make up this gap, motivated by the recent success of gating design in segmentation [50,9], we propose a new FAM variant named Gated Dual Flow Alignment Module (GD-FAM) to directly align and fuse both *highest-resolution feature* and *lowest-resolution feature*. Since there is only one aligment, which means less operators are involved, we can speed up the inference time.

Gated Dual Flow Alignment Module. As FAM, GD-FAM takes two features $\mathbf{F}_4$ and $\mathbf{F}_1$ as inputs and directly outputs a refined high resolution feature. We up-sample $\mathbf{F}_4$ to the same size as $\mathbf{F}_1$ via a bi-linear interpolation layer. Then, we concatenate them together and take the concatenated feature map as input for a subnetwork $conv_F$ that contains two convolutional layers with the kernel size of 3×3. Such network directly outputs a new flow map $\Delta_F \in \mathbb{R}^{H_4 \times W_4 \times 4}$.

$$\Delta_F = \mathrm{conv}_F(\mathrm{cat}(\mathbf{F}_4, \mathbf{F}_1)). \tag{4}$$

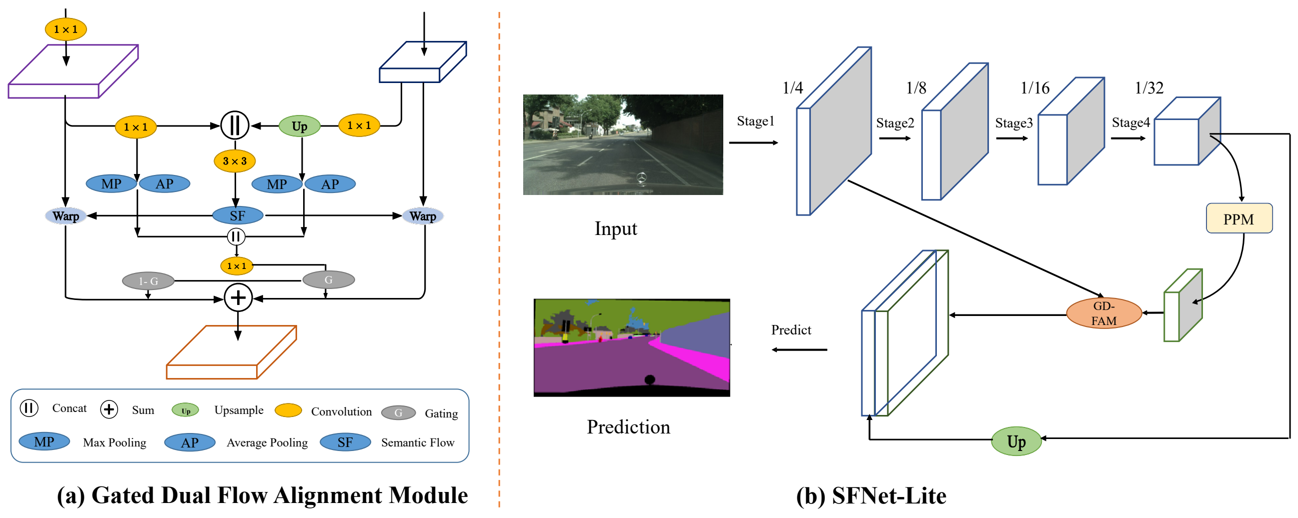

Fig. 5: (a) The details of GD-FAM (Gated Dual Flow Alignment Module). We combine the transformed high-resolution feature map and low-resolution feature map to generate the two semantic flow fields and one shared gate map. The semantic flows are utilized to warp both the low-resolution feature map and the high-resolution feature map. The gate controls the fusion process. (b) Overview of our proposed SFNet-Lite. ResNet-18 backbone with four stages is used for exemplar illustration. GD-FAM: Gated Dual Flow Alignment Module. PPM: Pyramid Pooling Module [2]. Best view it in color and zoom in.

We split such map Δ_F into Δ_{F1} and Δ_{F4} to jointly align both $\mathbf{F}_1$ and $\mathbf{F}_4$. Moreover, we propose to a shared gate map to highlight most important area on both aligned features. Our key insight is to make full use of high level semantic feature and let the low level feature as a supplement of high level feature. In particular, we adopt another subnetwork $conv_g$ to that contains one convolutional layer with the kernel size of 1×1 and one Sigmoid layer to generate such gate map. To highlight the most important regions of both features, we adopt max pooling (Maxpool) and average pooling (Avepool) over both features. Then we concatenate all four maps to generate such learnable gating maps. This process is shown as following:

$$\Delta_G = \mathrm{conv}_g(\mathrm{cat}(\mathrm{Avepool}(\mathbf{F}_4, \mathbf{F}_1)).\mathrm{Maxpool}(\mathbf{F}_4, \mathbf{F}_1))), \tag{5}$$

Then we adopt Δ_G to weight the aligned high semantic features and use inversion of Δ_G to weight the aligned low semantic features as fusion process. The key insights are two folds. Firstly, sharing the same gates can better highlight the most salient region. Secondly, adopting the subtracted gating supplies the missing details in low resolution feature. Such process is shown as following:

$$\mathbf{F}_{fuse} = \Delta_G Wrap(\Delta_{F1}, F1) + (1 - \Delta_G) Wrap(\Delta_{F4}, F4). \tag{6}$$

where the Wrap process is the same as Equation 3. Our key insight is that a better fusion of both features can lead to more fine-grained feature representation: rich semantic and high resolution feature map. The entire process is shown in Figure 5(a).

Table 1: Speed comparison (FPS) on different devices for SFNet and SFNet-Lite. We adopt Resnet-18 as backbone. The FPS is measured by 1024×2048 input images.

Device	1080-TI	TTIAN-RTX	3090-RTX	TITAN-RTX(TensorRT)
SFNet	18.1	20.1	24.2	33.3
SFNet-Lite	26.5	27.2	40.2	48.9

Lite Aligned Decoder. The Lite Aligned Decoder is the simplified version of Aligned Decoder, which contains one GD-FAM and one PPM. As shown in Figure 5(b), the final segmentation head takes the output of $\mathbf{F}_{fuse}$ and upsampled deep features in last stage as inputs and outputs the final segmentation map via one 1×1 convolution over the combined inputs. Lite Aligned Decoder speeds up the Aligned Decoder via involving less multiscale features (only two scales). Avoiding shortcut design can also lead to faster speed when deploying the models on devices for practical usage. More results can be found in the experiment part.

Speed Comparison Analysis. In Table 1, we compare the speed of SFNet and SFNet-Lite on different devices. SFNet-Lite runs faster on various devices. In particular, when deploying both on TensorRT, the SFNet-Lite is *even much faster* than SFNet since it involves less

Table 2: Dataset Information of our merged UDS dataset. We merged Mapillary labels into cityscapes label format.

Dataset Name	Train Images	Validation Images	Number of Class Labels
Cityscapes	2,975	500	19
IDD	6,993	3,000	19
Mapillary	18,000	2,000	65
BDD	7,000	1,000	19
UDS (ours)	34,968	6,500	19

cross scale branches and leads to better optimization for acceleration.

3.4 The Unified Driving Segmentation Dataset

Motivation. Learning a unified driving-target segmentation model is useful since the environment may change a lot during the moving of self-driving cars. MSeg [81] presents a more challenging setting while we only focus on high resolution out-door driving scene. Since the concepts of road scenes are limited, we only have small label space compared with M-Seg, which it has several common scenes (COCO [94], ADE20k [13]).

We verify the effectiveness of our SFNet series on new setting for feature alignment in various domains *without* introducing domain aware learning [32]. The goal of UDS is to provide more fair comparison on driving scene segmentation. To our knowledge, we are the first to benchmark such large-scale driving datasets using one model.

Data Process and Results. We merge four challenging datasets including Mapillary [27], Cityscapes [25], IDD [26] and BDD [28]. Since Mapillary has 65 class labels, we merge several semantic labels into one label. The merging process follows the previous work [32]. We set other labels as ignore region. In this way, we keep the same label definition as Cityscapes and IDD. For IDD dataset, we use the same class definition as Cityscapes and BDD. For BDD and Cityscapes datasets, we keep the original setting. The merged dataset UDS totally has 34,968 images for training and 6,500 images for testing. The details of the UDS dataset are shown in Table 2. Moreover, we find that several recent self-attention based methods [4,95,96] cannot perform well than previous method DeeplabV3+ [8]. This implies a better generalized method is needed for this setting. We provide the code and model on the github pages.

Discussion. Note that despite designing more balanced sampling methods or including domain generalization based method can improve the results on UDS, the goal of this work is only to verify the effectiveness of our SFNet and SFNet-Lite on this challenging setting. Both GD-GAM and FAM perform image feature level alignment, which are *not sensitive* to the domain variations. Moreover, we also show the effectiveness of SFNet on domain generation settings using RobustNet [32]. More details can be found in experiment part.

3.5 Improvement Details and Extension.

Improvement Details. We use deeply supervised loss [2] to supervise intermediate outputs of the decoder for easier optimization. In addition, following [62], online hard example mining [97] is also used by only training on the 10% hardest pixels sorted by cross-entropy loss. During the inference, we only use the results from the main head. We also use uniform sampling methods to balance the rare class during training for all benchmarks. For the Cityscapes dataset, we also use the coarse boosting training tricks [5] to boost rare classes on Cityscapes. For backbone design, we also deploy the latest advanced backbone STDC [66] to speed up the inference speed on the device.

Extending SFNet into Panoptic Segmentation. Panoptic Segmentation unifies both semantic segmentation and instance segmentation, which is a more challenging task. We also explore the proposed SFNet on such task with the proposed panoptic segmentation baseline K-Net [57]. K-Net is a state-of-the-art panoptic segmentation method where each thing and stuff is represented by kernels in its decoder head. In particular, we replace the backbone part of K-Net with our proposed SFNet backbone and aligned decoder. Then we train the modified model using the same setting as K-Net.

4 Experiment

4.1 Experiment Settings

Overview. We first review the dataset and training setting for SFNet. Then, we present the result comparison on five road-driving datasets, including the original SFNet and the newly proposed SFNet-lite. After that, we give detailed ablation studies and analysis on our SFNet. Finally, we present the generalization ability of SFNet on the Cityscapes Panoptic Segmentation dataset.

DataSets. We mainly carry out experiments on the road driving datasets, including Cityscapes, Mapillary, IDD, BDD, and our proposed merged driving dataset. We also report panoptic segmentation results on the Cityscapes validation set. Cityscapes [25] is a benchmark densely annotated for 19 categories of urban scenes, which contains 5,000 fine annotated images in total and is divided into 2,975, 500, and 1,525 images for training, validation, and testing, respectively. In addition, 20,000 coarse-labeled images are also provided to enrich the training data. Images are all with the same high resolution in the road driving scene, i.e., 1024×2048. Note that we use the fine-annotated dataset for ablation study and comparison with previous methods. We also use the coarse data to boost the final results of SFNet-Lite. Mapillary [27] is a large-scale road-driving dataset, which is more challenging than Cityscapes since it contains more classes and various scenes. It contains 18,000 images for training and 2,000 images for validation. IDD [26] is another road-driving dataset that mainly contains the India scene. It contains more images than Cityscapes. It has 6,993 training images and 981 validation images. To our knowledge, we are *the first* to benchmark the real-time segmentation models on Mapillary and IDD datasets. Another research group develops the BDD dataset, which mainly contains various scenes in American areas. It has 7,000 training images and 1,000 validation images. *All the datasets, including UDS dataset, are available online.*

Implementation Details. We use PyTorch [98] framework to carry out all the experiments. All networks are trained with the same setting, where stochastic gradient descent (SGD) with batch size of 16 is used as an optimizer, with a momentum of 0.9 and weight decay of 5e-4. All models are trained for 50K iterations with an initial learning rate of 0.01. As a common practice, the "poly" learning rate policy is adopted to decay the initial learning rate by multiplying $(1 - \frac{\text{iter}}{\text{total\_iter}})^{0.9}$ during training. Data augmentation contains random horizontal flip, random resizing with a scale range of $[0.75,\ 2.0]$, and random cropping with crop size of 1024×1024 for Cityscapes, Mapillary, BDD, IDD, and UDS datasets. For quantitative evaluation, the mean of class-wise Intersection-Over-Union (mIoU) is used for an accurate comparison, and the number of Floating-point Operations Per Second (FLOPs) and Frames Per Second (FPS) are adopted for speed comparison. Moreover, to achieve a stronger baseline, we also adopt the class-balanced sampling strategy proposed in [5], which obtains stronger baselines. For the Cityscapes dataset, we also adopt coarse annotated data boosting methods to improve rare class segmentation quality. Our code and model are available for reference. Also note that several non-real segmentation methods in Mapillary, BDD, IDD, and USD datasets are implemented using our codebase and trained under the same setting.

TensorRT Deployment Device. The testing environment is TensorRT 8.2.0 with CUDA 11.2 on a single TITAN-RTX GPU. In addition, we re-implement the grid sampling operator by CUDA to be used together with TensorRT. The operator is provided by PyTorch and used in warping operations in the Flow Alignment Module. We report an average time of inferencing 100 images. Moreover, we also deploy our SFNet and SFNet-Lite on different devices, including 1080-TI and RTX-3090. We report the results in the next part.

4.2 Main Results

Results On Cityscapes test set. We first report our SFNet on the Cityscapes dataset in Table 3. With ResNet-18 as the backbone, our method achieves 79.8% mIoU and even reaches the performance of accurate models, which will be discussed next. Adopting STDC net as the backbone, our method achieves 79.8% mIoU with full resolution inputs while running at 80 FPS. This suggests that our method can be benefited from a well-human-designed backbone. For the improved SFNet-Lite, our method can achieve even better results than the original SFNet while running faster using ResNet-18 as the backbone. For the STDC backbone, our method achieves much faster speed while maintaining similar accuracy. In particular, using STDC-v1, our method achieves 78.8% mIoU while running at 120 FPS, a new state-of-the-art result on balancing speed and accuracy. This indicates the effectiveness of our proposed GD-FAM.

Note that for fair comparison, in Table 3, following previous works [66, 63], we report the speed using Tensor-RT devices. For the results on the remaining datasets, we only report GPU average inference time. The Original SFNet with ResNet-18 achieves 78.9 % mIoU, and we adopt uniform sampling, coarse boosting, and long-time training, which leads to an extra 0.9 % gain on the test set. The details can be found in the following sections.

Results on Mapillary validation set. In Table 4, we report speed and accuracy results on a more challenging Mapillary dataset. Since this dataset contains huge resolution images and direct inference may raise the out-of-memory issue, we resize the short size of the image to 1,536 and crop the image and ground truth center following [5].

As shown in Table 4, our methods also achieve the best speed and accuracy trade-off for various backbones. Even though the Deeplabv3+ [8] and EMANet [96] achieve higher accuracy, their speed cannot reach the real-time standard. In particular, for the DFNet-based backbone [29], our SFNet achieves **almost 5-6% mIoU** improvements. For SFNet-Lite, our methods also achieve considerable results while running faster.

Results on IDD validation set. In Table 5, our methods achieve the best speed and accuracy trade-off. Compared with previous work STDCNet, our method achieves better accuracy and faster speed, as shown

Table 3: Comparison on Cityscapes *test* set with state-of-the-art real-time models. For a fair comparison, the input size is also considered, and all models use single-scale inference. The FPS of our SFNet is evaluated on TensorRT following [66].

Method	InputSize	mIoU (%)	#FPS	#Params
ESPNet [60]	512 × 1024	60.3	132	0.4M
ESPNetv2 [61]	512 × 1024	62.1	80	0.8M
ERFNet [99]	512 × 1024	69.7	41.9	-
BiSeNet(ResNet-18) [62]	768 × 1536	74.6	43	12.9M
BiSeNet(Xception-39) [62]	768 × 1536	68.4	72	5.8M
BiSeNetv2(ResNet-18) [63]	768 × 1536	75.3	47.3	-
BiSeNetv2(Xception-39) [63]	768 × 1536	72.6	156	-
ICNet [58]	1024 × 2048	69.5	34	26.5M
DF1-Seg [29]	1024 × 2048	73.0	80	8.6M
DF2-Seg [29]	1024 × 2048	74.8	55	18.9M
SwiftNet [22]	1024 × 2048	75.5	39.9	11.8M
SwiftNet-ens [22]	1024 × 2048	76.5	18.4	24.7M
DFANet [59]	1024 × 1024	71.3	100	7.8M
CellNet [65]	768 × 1536	70.5	108	-
STDC1-Seg75 [66]	768 × 1536	75.3	126.7	12.0M
STDC2-Seg75 [66]	768 × 1536	76.8	97.0	16.1M
HyperSeg-M [100]	512 × 1024	75.8	36.9	10.1M
HyperSeg-S [100]	768 × 1536	78.1	16.1	10.2M
DDRNet-23 [101]	1024 × 2048	77.4	108.1	5.7M
SFNet(DF1)	1024 × 2048	74.5	134.5	9.0M
SFNet(DF2)	1024 × 2048	77.8	103.1	19.6M
SFNet(ResNet-18)	1024 × 2048	79.8	33.3	12.9M
SFNet(STDC-1)	1024 × 2048	78.1	97.1	9.1M
SFNet(STDC-2)	1024 × 2048	79.8	79.9	13.1M
SFNet-Lite(ResNet-18)	1024 × 2048	80.1	48.9	12.3M
SFNet-Lite(STDC-1)	1024 × 2048	78.8	119.1	9.7M
SFNet-Lite(STDC-2)	1024 × 2048	79.0	92.3	13.7M

Table 4: Comparison on Mapillary *validation* set with state-of-the-art models. All the models are re-trained for a fair comparison and use single-scale inference with the same resolution input. The non-real-time models in the first sub-table use ResNet-50 as the backbone. The mIoU and FPS are measured input image size with 1536 × 1536. All the models are tested on a single TITAN-RTX.

Method	mIoU (%)	#FPS	#Params
PSPNet [2]	42.4	4.8	31.1M
Deeplabv3+ [8]	46.4	3.2	40.5M
DANet [4]	42.9	2.0	48.1M
OCRNet [95]	46.6	3.8	39.0M
EMANet [96]	47.5	4.2	34.8M
BiSeNet-V1(ResNet-18) [62]	43.2	24.3	12.9M
ICNet [58]	42.8	48.2	26.5M
DF1-Seg [29]	35.8	125.1	8.6M
DF2-Seg [29]	40.2	75.2	18.9M
STDC1 [66]	41.9	34.5	12.0M
STDC2 [66]	43.5	29.0	16.1M
SFNet(DF1)	41.4	102.2	9.0M
SFNet(DF2)	45.6	57.8	19.6M
SFNet(ResNet-18)	46.5	19.8	12.9M
SFNet-Lite(ResNet-18)	46.3	24.5	12.3M
SFNet-Lite(STDC-2)	45.8	35.8	13.7M

in the last row of Table 5. For DFNet backbone, our methods also achieve nearly 12% mIoU relative improvements. Such results indicate that the proposed FAM and GD-FAM accurately align the low-resolution feature into more accurate high-resolution and high-semantic feature maps.

Table 5: Comparison on IDD *validation* set with state-of-the-art models. For a fair comparison, all the models are re-trained and use single scale inference with the same resolution inputs (1080 × 1920 original size of IDD).

Method	mIoU (%)	#FPS	#Params
PSPNet [2]	77.6	5.2	31.1M
Deeplabv3+ [8]	78.9	3.5	40.5M
DANet [4]	76.6	3.2	48.1M
OCRNet [95]	78.1	4.2	39.0M
EMANet [96]	77.2	4.4	34.8M
BiSeNet-V1(ResNet-18) [62]	74.4	24.5	12.9M
ICNet [58]	73.8	37.5	26.5M
DF1-Seg [29]	63.4	79.2	8.55M
DF2-Seg [29]	67.9	50.8	18.88M
STDC1 [66]	75.5	30.8	12.0M
STDC2 [66]	76.3	24.8	16.1M
Bi-Align [102]	73.9	30.2	19.2M
SFNet(DF1)	75.8	65.8	9.03M
SFNet(DF2)	76.3	37.4	19.63M
SFNet(ResNet-18)	76.8	20.2	12.87M
SFNet-Lite(ResNet-18)	76.2	26.8	12.3M
SFNet-Lite(STDC-2)	76.8	26.2	13.7M

Table 6: Comparison on BDD *validation* set with state-of-the-art models. For a fair comparison, all the models are re-trained and use single scale inference with the same resolution inputs (720 × 1280 original size of BDD).

Method	mIoU (%)	#FPS	#Params
PSPNet [2]	62.3	11.2	31.1M
Deeplabv3+ [8]	63.6	7.3	40.5M
DANet [4]	62.8	6.6	48.1M
OCRNet [95]	60.1	7.1	39.0M
EMANet [96]	61.4	9.6	34.8M
BiSeNet-V1(ResNet-18) [62]	53.8	45.1	12.9M
ICNet [58]	52.4	39.5	26.5M
Bi-Align [102]	53.4	42.1	19.2M
DF1-Seg [29]	42.5	82.3	8.6M
DF2-Seg [29]	47.8	53.4	18.9M
STDC1 [66]	52.1	45.8	12.0M
STDC2 [66]	53.8	33.0	16.1M
SFNet(DF1)	55.4	70.3	9.0M
SFNet(DF2)	60.2	47.3	19.6M
SFNet(ResNet-18)	60.6	35.6	12.9M
SFNet-Lite(ResNet-18)	60.6	44.3	12.3M
SFNet-Lite(STDC-2)	59.4	29.8	13.7M

Results on BDD validation set. In Table 6, we further benchmark the representative works on BDD dataset. From that table, Deeplabv3+ [8] achieves the top performance but with a much slower speed. Again, our methods, including both original SFNet and improved SFNet-Lite achieve the best speed and accuracy trade-off. For the recent state-of-the-art method STDC-Net [66], our SFNet-Lite achieves 5% mIoU improvement

Table 7: Comparison on UDS *testing* set with state-of-the-art models. All the models are re-trained for fair comparison and use single scale inference with the same resolution inputs (1024×2048, on both resized images and ground truth).

Method	mIoU (%)	#FPS	#Params
PSPNet [2]	75.2	5.3	31.1M
Deeplabv3+ [8]	78.0	3.7	40.5M
DANet [4]	75.8	3.0	48.1M
OCRNet [95]	77.0	4.2	39.0M
EMANet [96]	76.8	4.4	34.8M
BiSeNet-V1(ResNet-18) [62]	73.8	24.5	12.9M
ICNet [58]	72.9	38.5	26.5M
Bi-Align [102]	73.9	30.1	19.2M
DF1-Seg [29]	62.6	75.2	8.6M
DF2-Seg [29]	66.8	45.1	18.9M
STDC1 [66]	74.0	30.2	12.0M
STDC2 [66]	75.2	24.5	16.1M
SFNet(DF1)	71.6	69.5	9.0M
SFNet(DF2)	75.5	37.5	19.6M
SFNet(ResNet-18)	76.5	19.4	12.9M
SFNet-Lite(ResNet-18)	75.3	24.5	12.3M
SFNet-Lite(STDC-1)	74.8	33.5	13.7M
SFNet-Lite(STDC-2)	75.6	30.2	13.7M

while running slower. When adopting the ResNet-18 backbone, our SFNet-Lite achieves 60.6% mIoU while running at 44.5 FPS without TensorRT acceleration.

Results on USD testing set. Finally, we benchmark the recent works on the merged USD dataset in Table 7. To fit the GPU memory, we resize both images and ground truth images to 1024×2048. From that table, we find Deeplabv3+ [8] achieves top performance. Several self-attention-based models [96,95,4] achieve even worse results than previous Deeplabv3+ on such domain variant datasets. This shows that the USD dataset still leaves a huge room to improve.

As shown in Table 7, our methods using DFNet backbones achieve *relatively 10% mIoU improvements* over DF-Seg baselines. When equipped with the ResNet-18 backbone, our SFNet achieves 76.5% mIoU while running at 20 FPS. When adopting the STDC-V2 backbone, our SFNet-Lite achieves the best speed and accuracy trade-off.

4.3 Ablation Studies

Effectiveness of FAM and GD-FAM. Table 8(a) reports the comparison results against baselines on the validation set of Cityscapes [25], where ResNet-18 [17] serves as the backbone. Compared with the naive FCN, dilated FCN improves mIoU by 1.1%. By appending the FPN decoder to the naive FCN, we get 74.8% mIoU by an improvement of 3.2%. By replacing bilinear upsampling with the proposed FAM, mIoU is boosted to 77.2%, which improves the naive FCN and FPN decoder by 5.7% and 2.4%, respectively. Finally, we append PPM (Pyramid Pooling Module) [2] to capture global contextual information, which achieves the best mIoU of 78.7 % together with FAM. Meanwhile, FAM is complementary to PPM by observing FAM improves PPM from 76.6% to 78.7%. In Table 10(a), we compare the effectiveness of GD-FAM and FAM. As shown in that table, our new proposed GD-FAM has better performance (0.4%) while running faster than the original FAM under the same settings.

Positions to insert FAM or GD-FAM: We insert FAM to different stage positions in the FPN decoder and report the results in Table 8(b). From the first three rows, FAM improves all stages and gets the greatest improvement at the last stage, demonstrating that misalignment exists in all stages of FPN and is more severe in coarse layers. This is consistent with the fact that coarse layers contain stronger semantics but with lower resolution and can greatly boost segmentation performance when they are appropriately upsampled to high resolution. The best result is achieved by adding FAM to all stages in the last row. For GD-FAM, we aim to align the high-resolution features and low-resolution directly. We choose to align F_3 and the output of PPM by default.

Ablation study on network architecture design: Considering current state-of-the-art contextual modules are used as heads on dilated backbone networks [35,36], we further try different contextual heads in our methods where the coarse feature map is used for contextual modeling. Table 8(c) reports the comparison results, where PPM [2] delivers the best result, while the more recently proposed methods such as non-Local-based heads [42] perform worse. Therefore, we choose PPM as our contextual head due to its better performance with lower computational cost.

Ablation on FAM design. We first explore the effect of upsampling in FAM in Table 9(a). Replacing the bilinear upsampling with deconvolution and nearest neighbor upsampling achieves 77.9% mIoU and 78.2% mIoU, respectively, which are similar to the 78.3% mIoU achieved by bilinear upsampling. We also try the various kernel sizes in Table 9(b). A larger kernel size of 5×5 is also tried, which results in a similar result (78.2%) but introduces more computation cost. In Table 9(c), replacing FlowNet-S with correlation in FlowNet-C also leads to slightly worse results (77.2%) but increases the inference time. The results show that it is enough to use lightweight FlowNet-S for aligning feature maps in FPN. In Table 9(d), we compare our results with DCN [92]. We apply DCN on the concatenated feature map of the bilinear upsampled feature map and the feature map of

Table 8: Ablation studies on **SFNet architecture design** using Cityscapes validation set.

(a) Ablation study on baseline model.

Method	Stride	mIoU (%)	Δa(%)
FCN	32	71.5	-
Dilated FCN	8	72.6	1.1 ↑
+FPN	32	74.8	3.3 ↑
+FAM	32	77.2	5.7 ↑
+FPN + PPM	32	76.6	5.1 ↑
+FAM + PPM	32	78.3	7.2 ↑

(b) Ablation study on insertion position.

Method	$\mathbf{F}_3$	$\mathbf{F}_4$	$\mathbf{F}_5$	mIoU(%)	Δa(%)
FPN+PPM	-	-	-	76.6	-
	✓	-	-	76.9	0.3 ↑
	-	✓	-	77.0	0.4 ↑
	-	-	✓	77.5	0.9 ↑
	-	✓	✓	77.8	1.2 ↑
	✓	✓	✓	78.3	1.7 ↑

(c) Ablation study on context module.

Method	mIoU(%)	Δa(%)	#GFLOPs
FAM	76.4	-	-
+PPM [2]	78.3	1.9↑	123.5
+NL [42]	76.8	0.4↑	148.0
+ASPP [35]	77.6	1.2↑	138.6
+DenseASPP [36]	77.5	1.1↑	141.5

Table 9: Ablation results on **FAM design** using Cityscapes validation set.

(a) Ablation study on Upsampling operation in FAM.

Method	mIoU (%)
bilinear upsampling	78.3
deconvolution	77.9
nearest neighbor	78.2

(b) Ablation study on kernel size k in FAM where 3 FAMs are involved.

Method	mIoU (%)	Gflops
$k = 1$	77.8	120.4
$k = 3$	78.3	123.5
$k = 5$	78.1	131.6
$k = 7$	78.0	140.5

(c) Ablation with FlowNet-C [90] in FAM.

Method	mIoU (%)	Δa(%)
FPN +PPM	76.6	-
correlation [90]	77.2	0.6 ↑
Ours	77.5	0.9 ↑

(d) Comparison with DCN [92].

Method	$\mathbf{F}_3$	$\mathbf{F}_4$	$\mathbf{F}_5$	mIoU(%)	Δa(%)
FPN +PPM	-	-	-	76.6	-
DCN	-	-	✓	76.9	0.3 ↑
Ours	-	-	✓	77.5	0.9 ↑
DCN	✓	✓	✓	77.2	0.6 ↑
Ours	✓	✓	✓	78.3	1.7 ↑

Table 10: Ablation experiment results on SFNet-Lite and **GD-FAM design** using Cityscapes validation set. DF: Dual Flow. Attn: Attention. G: Gate. US: Uniform Sampling. LT: Long Training. CB: Coarse Boosting.

(a) Effectiveness of GD-FAM. FPS is measuerd with 1024 × 2048 input.

Method	mIoU (%)	FPS
FCN	71.5	50.3
+FPN + PPM (baseline)	76.6	40.3
+ FAM + PPM	78.3	19.4
+ one GD-FAM + PPM	78.3	24.5

(b) Ablation study on components in GD-FAM.

Method	DF	Atten	G	mIoU(%)
FPN+PPM	-	-	-	76.6
	✓	-	-	77.8
	✓	✓	-	78.0
	✓	✓	✓	78.3

(c) Ablations study on Improving Tricks.

Method	US	LT	CB	mIoU(%)
SFNet-Lite	-	-	-	78.3
	✓	-	-	78.6
	✓	✓	-	79.0
	✓	✓	✓	79.7

Table 11: Generalization on Various Backbone. For SFNet series, the baseline models are without FAM or GD-FAM. Note, GD-FAM is only used once. The GFlops are calculated with 1024 × 2048 input.

Backbone	mIoU(%)	Δa(%)	#GFLOPs	Δb(%)
ResNet-50 [17]	76.8	-	332.6	-
w/ FAM	79.2	2.4 ↑	337.1	+4.5
ResNet-101 [17]	77.6	-	412.7	
w/ FAM	79.8	2.2↑	417.5	+4.8
w/ GD-FAM	80.2	2.6↑	415.3	+2.6
ShuffleNetv2 [80]	69.8	-	17.8	-
w/ FAM	72.1	2.3 ↑	18.1	+0.3
DF1 [29]	72.1	-	18.6	-
w/ FAM	74.3	2.2 ↑	18.7	+0.1
DF2 [29]	73.2	-	48.2	-
w/ FAM	75.8	2.6 ↑	48.5	+0.3
STDC-Net-v1 [66]	75.0	-	58.2	-
w/ FAM	76.7	1.7↑	59.8	+1.6
w/ GD-FAM	76.5	1.5↑	59.0	+0.8
STDC-Net-v2 [66]	75.6	-	85.0	-
w/ FAM	77.4	1.8↑	86.3	+1.3
w/ GD-FAM	77.5	1.9↑	86.2	+1.2

the next level. We first insert one DCN in higher layers $\mathbf{F}_5$ where our FAM is better than it. After applying DCN to all layers, the performance gap is much larger. This indicates that our method can also align low-level edges for better boundaries and edges in lower layers, which will be shown in the visualization part.

Ablation GD-FAM design. In Table 10(b), we explore the effect of each component in GD-FAM. In particular, adding Dual Flow (DF) design boosts about 1.2% improvement. Using Attention to generate gates rather than using convolution leads to 0.2% improvement. Finally, using the shared gate design also improves the strong baseline by 0.3%.

Ablation on Improving Details. In Table 10(c), we explore the training tricks, including Uniform Sampling (US), Long Training (LT) and Coarse Boosting (CB). Performing US leads to 0.3% improvements on our SFNet-Lite. Using LT (1000 epochs training) rather than short training (300 epochs training) results in another 0.4% mIoU improvement. Finally, adopting coarse data boosts on several rare classes leads to another 0.7% improvement.

generalization on Various Backbones. We further carry out experiments with different backbone networks, including both deep and light-weight networks, where the FPN decoder with PPM head is used as a strong baseline in Table 11. For heavy networks, we choose ResNet-50 and ResNet-101 [17] to extract representation. For light-weight networks, ShuffleNetv2 [80], DF1/DF2 [29] and STDC-Net [66] are employed. FAM significantly achieves better mIoU on all backbones with slightly extra computational cost. Both GD-FAM and

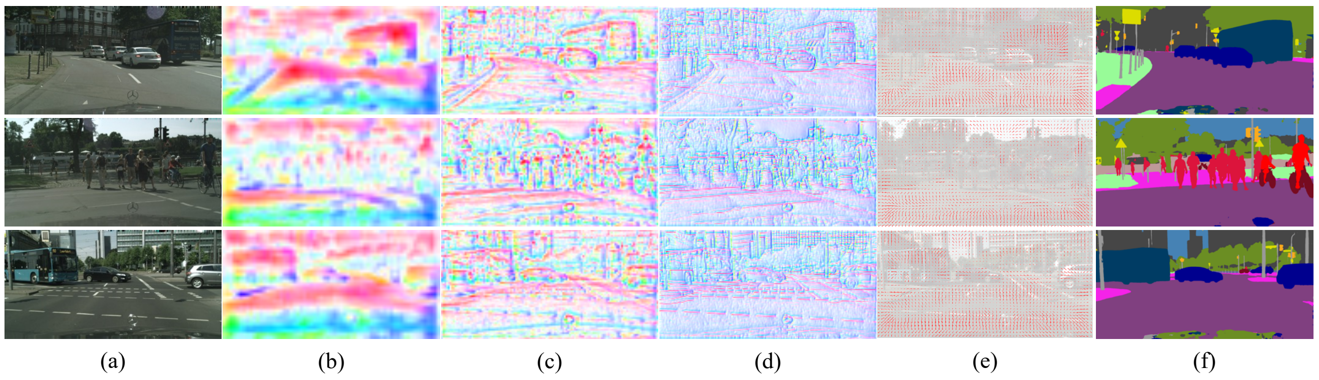

Fig. 6: Visualization of the learned semantic flow fields. Column (a) lists three exemplary images. Column (b)-(d) show the semantic flow of the three FAMs in ascending order of resolution during the decoding process, following the same color coding of Figure 3. Column (e) is the arrowhead visualization of flow fields in column (d). Column (f) contains the segmentation results.

FAM improve the results of different backbones significantly with little extra computation cost.

Aligned feature representation: In this part, we present more visualization on aligned feature representation as shown in Figure 7. We visualize the upsampled feature in the final stage of ResNet-18. It shows that compared with DCN [92], our FAM feature is more structural and has much more precise object boundaries, which is consistent with the results in Table 9(d). That indicates that FAM is **not** an attention effect on a feature similar to DCN, but aligns the feature towards a more precise shape than in red boxes.

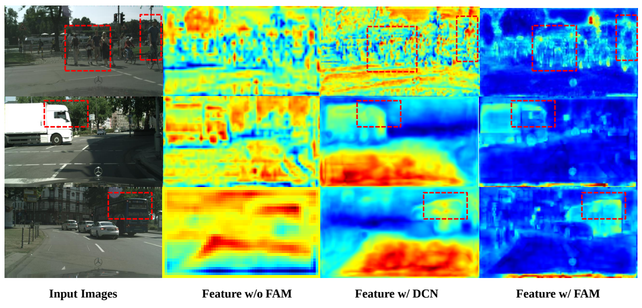

Fig. 7: Visualization of the aligned feature. Compared with DCN, our module outputs more structural feature representation.

4.4 More Detailed Analysis

Detailed Improvements. Table 12 compares the detailed results of each category on the validation set, where ResNet-101 is used as backbone, and FPN decoder with PPM head serves as the baseline. SFNet improves almost all categories, especially for 'truck' with more than 19% mIoU improvement. Adopting GD-FAM leads to more consistent improvement over FAM on each class.

Visualization of Semantic Flow. Figure 6 visualizes semantic flow from FAM in different stages. Similar to optical flow, semantic flow is visualized by color coding and is bilinearly interpolated to image size for a quick overview. Besides, vector fields are also visualized for detailed inspection. From the visualization, we observe that semantic flow tends to diffuse out from some positions inside objects. These positions are generally near the object centers and have better receptive fields to activate top-level features with pure and strong semantics. Top-level features at these positions are then propagated to appropriate high-resolution positions following the guidance of semantic flow. In addition, semantic flows also have coarse-to-fine trends from the top level to the bottom level. This phenomenon is consistent with the fact that semantic flows gradually describe offsets between gradually smaller patterns.

Visual Improvements on Cityscapes dataset. Figure 8(a) visualizes the prediction errors by both methods, where FAM considerably resolves ambiguities inside large objects (e.g., truck) and produces more precise boundaries for small and thin objects (e.g., poles, edges of wall). Figure8 (b) shows our model can better handle the small objects with shaper boundaries than dilated PSPNet due to the alignment on lower layers.

Visualization Comparison on Mapillary dataset. In Figure 9, we show the visual comparison results on the Mapillary dataset. As shown in that figure, compared with previous ICNet and BiSegNet, our SFNet-Lite using ResNet-18 as backbone has better segmentation results in cases of more accurate segmentation classification and structural output.

Table 12: Quantitative per-category comparison results on Cityscapes validation set, where ResNet-101 backbone with the FPN decoder and PPM head serves as the strong baseline. Obviously, Both FAM and GD-FAM boost the performance of almost all the categories.

Method	road	swalk	build	wall	fence	pole	tlight	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIoU
BaseLine	98.1	84.9	92.6	54.8	62.2	66.0	72.8	80.8	92.4	60.6	94.8	83.1	66.0	94.9	65.9	83.9	70.5	66.0	78.9	77.6
with FAM	98.3	85.9	93.2	62.2	67.2	67.3	73.2	81.1	92.8	60.5	95.6	83.2	65.0	95.7	84.1	89.6	75.1	67.7	78.8	79.8
with GD-FAM	98.3	86.7	93.5	63.4	67.1	68.2	73.5	81.9	92.7	64.4	95.4	84.2	68.4	95.4	85.7	91.2	83.2	67.7	79.3	81.0

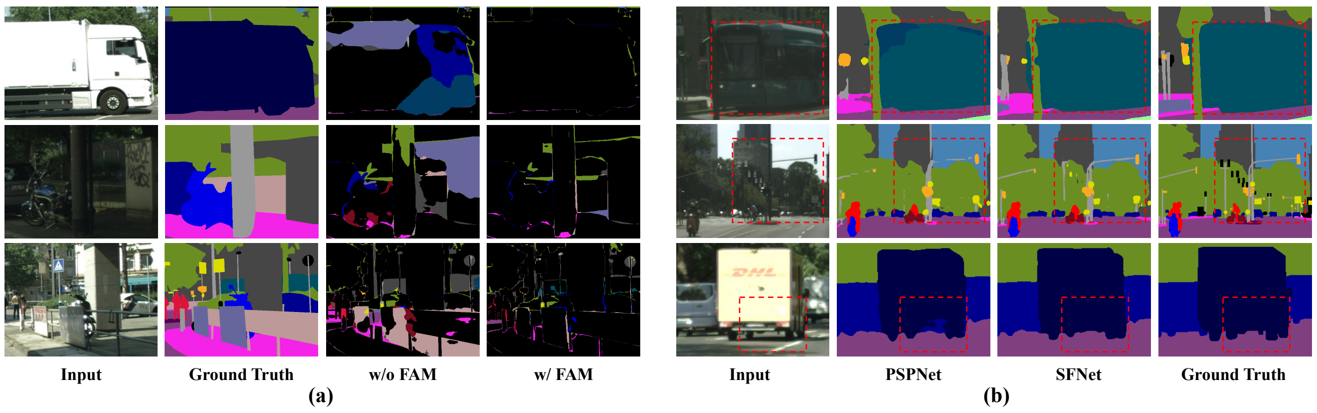

Fig. 8: (a), Qualitative comparison in terms of errors in predictions, where correctly predicted pixels are shown as black background while wrongly predicted pixels are colored with their ground truth label color codes. (b), Scene parsing results comparison against PSPNet [2], where the improved regions are marked with red dashed boxes. Our method performs better on both small scale and large scale objects.

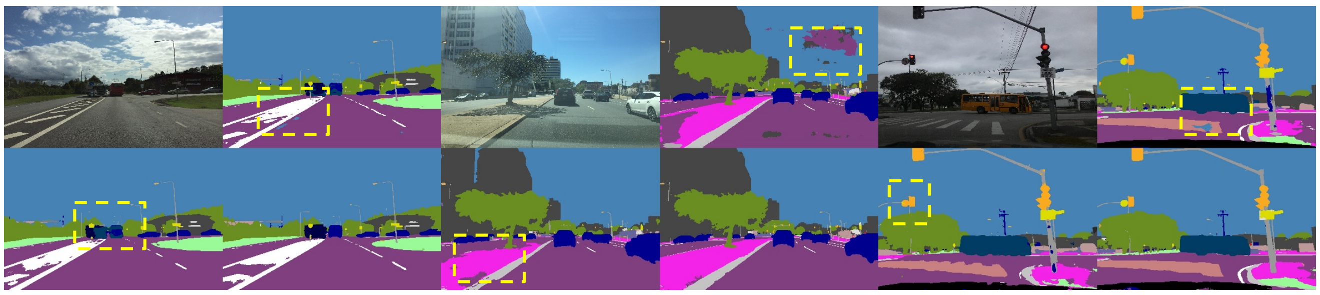

Fig. 9: Qualitative comparison on Mapillary dataset. Top-left: *Origin Images*. Top-Left: *Results of BiSegNet [62]*. Down-Left: *Results of ICNet [58]*. Down-Right: *Results of our SFNet-Lite*. Improvement regions are in yellow boxes. Best view it in color.

Visual Comparison on proposed USD dataset. In figure 10, we present several samples from different datasets. Compared with the original DFNet baseline, our method can achieve better segmentation results in terms of clear object boundaries and inner object consistency. We also show the SFNet-Lite with ResNet-18 backbone in the fourth row and overlapped images in the last row. The figure shows that our methods (SFNet with DFV2 backbone and SFNet-Lite with ResNet-18 backbone) achieve good segmentation quality for different domains.

Table 13: Speed Comparison on TensorRT Deployment testing with Different Devices. The FPS is measured with 1024×2048 input.

Network	1080-TI	TITAN-RTX	RTX-3090
SFNet(resnet-18)	26.8	34.2	50.5
SFNet(stdcv2)	56.2	78.0	202.1

Speed Effect on Different Devices. In Table 13, we explore the effect of deployment devices. In particular, compared with the original SFNet [30], which uses 1080-TI as a device, using a more advanced device leads to

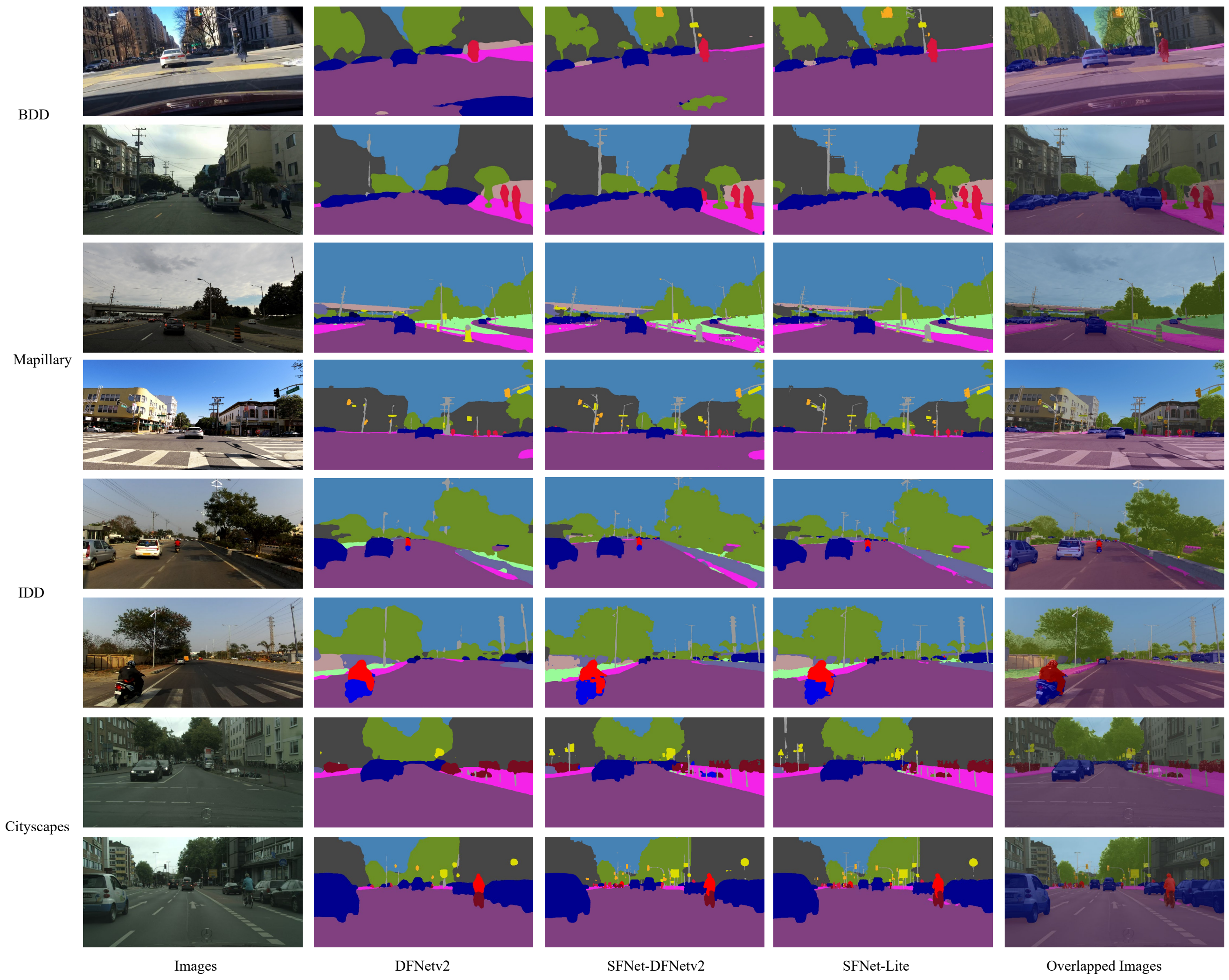

Fig. 10: Visualization results on UDS validation dataset including BDD, Maillary, IDD and Cityscapes. Our methods achieve the better visual results in cases of clear object boundary, inner object consistency and better structural outputs. We adopt singele scale inference and all the models are trained under the same setting. Best view it on screen and zoom in.

Table 14: Pretraining Effect of UDS dataset. The mIoU is evaluated on Camvid dataset [103].

Network	ImageNet	UDS	mIoU
SFNet(resnet-18)	✓	-	73.8
SFNet(stdcv2)	✓	-	72.9
SFNet(resnet-18)	-	✓	76.5
SFNet(stdcv2)	-	✓	75.6

UDS Used for Pre-training. We further show the effectiveness of our UDS dataset in table 14. Compared with ImageNet [93], adopting the pre-training with the UDS dataset can significantly boost SFNet results on the Camvid dataset [103], which leads to a significant margin (3-4% mIoU). This implies that the UDS dataset can be an excellent pre-train source to boost the model performance.

a much higher speed. For example, RTX-3090 results almost twice faster as 1080-TI using ResNet-18 and four times faster using STDCNet. Moreover, we also find that SFNet with STDCNet [66] backbone is more friendly to TensorRT deployment.

4.5 Extension on Efficient Panoptic Segmentation

Experiment Setting. In this section, we show the generalization ability of our Semantic Flow on more challenging task Panoptic Segmentation. We choose K-

Table 15: Experiment results on the Cityscapes Panoptic validation set. * indicates using DCN [92]. All the methods use single-scale inference. We prove the generalization ability of semantic flow. The FPS is measured on one V100 card with 1024 × 2048 input.

Method	Backbone	PQ	PQ_{th}	PQ_{st}	#FPS
UPSNet [73]	ResNet50	59.3	54.6	62.7	7.3
SOGNet [72]	ResNet50	60.0	56.7	62.5	6.7
Seamless [71]	ResNet50	60.2	55.6	63.6	-
Unifying [105]	ResNet50	61.4	54.7	66.3	-
Panoptic-DeepLab [76]	ResNet50	59.7	-	-	8.2
Panoptic FCN* [106]	ResNet50	61.4	54.8	66.6	-
K-Net [57]	ResNet50	61.2	52.4	66.8	10.2
STDCv1 + K-Net Head	STDCv1	58.0	50.3	62.4	23.3
SF-STDCv1 + K-Net Head	STDCv1	59.2	52.9	63.3	20.3
STDCv2 + K-Net Head	STDCv2	59.8	53.8	63.8	19.3
SF-STDCv2 + K-Net Head	STDCv2	60.3	54.4	64.8	18.6
SF-ResNet50 + K-Net Head	ResNet50	61.7	52.6	67.2	9.0

Net [57] as the prediction head, while our SFNet is the backbone and neck for the feature extractor. All the network is first trained on the COCO dataset and then on the Cityscapes dataset. For COCO [94] dataset pretraining, all the models are trained following detectron2 settings [104]. We adopt the multiscale training by resizing the input images such that the shortest side is at least 480 and 800 pixels, while the longest is at most 1333 pixels. We also apply random crop augmentations during training, where the training images are cropped with a probability of 0.5 to a random rectangular patch and then resized again to 800-1333 pixels. All the models are trained for 36 epochs. For Cityscape fine-tuning, we resize the images with a scale ranging from 0.5 to 2.0 and randomly crop the whole image during training with batch size 16. All the results are obtained via single-scale inference. We also report results using the ResNet50 backbone for reference. We report the FPS on V100 devices by averaging 100 input images. For FPS measurement, we also include the panoptic post-processing times.

Results on Various Baseline on Cityscapes Panoptic Segmentation. As shown in Table 15, our SFNet backbone improves the baseline models in terms of the Panoptic Quality metric by around 0.5-1.0%. The results show the generalization ability of the semantic flow because our aligned feature representation preserves more fine-grained information. Moreover, we compare our methods using a stronger ResNet50 backbone. Compared with K-Net [57], our methods still achieve 0.5% PQ improvements with 1.2 FPS drop. Our method with STDCv2 backbone achieves a strong speed and accuracy trade-off (60.3 PQ with 18.6 FPS).

Table 16: More detailed comparison between GF-FAM and DCN. We adopt ResNet18 as backbone.

Method	Cityscapes	UDS	#FPS
FCN +FPN + PPM (baseline)	76.6	72.3	40.3
w one DCN	76.9	73.5	22.8
w one FAM	77.5	74.8	23.3
w one GD-FAM	78.5	75.5	24.6

Table 17: Domain generalization experiments using SFNet and SFNet-Lite using RobustNet [32]. The baseline methods are SFNet series with RobustNet with no FAMs or GD-FAMs. Our methods also show better results on BDD and IDD when trained with Cityscapes dataset.

Method	Backbone	BDD	IDD	#FPS
baseline	ResNet18	43.2	46.2	25.8
SFNet	ResNet18	45.2	48.1	24.0
SFNet-Lite	ResNet18	46.0	48.5	24.8
baseline	STDC-2	39.2	41.4	27.5
SFNet-Lite	STDC-2	41.8	44.5	26.7

4.6 More Analysis on SFNet and SFNet-Lite

Experiment Setting. In this section, we perform more extensive experiments using SFNets. (1), We first conduct more experiments with DCN [92] using Cityscapes and UDS by adding one DCN layer and one GD-FAM. (2), Then, we perform domain generalization experiments using RobustNet with different SFNet baselines, where we train the model on the Cityscapes dataset and test the model on BDD and IDD datasets. (3), Next, we present the results on ADE20k datasets using different baselines, including Semantic FPN [19] and SegFormer [55]. For the experiments on the ADE20k dataset, we follow the default settings from OCRNet [95], where the crop size is set to 512 with 160k iterations training. The GFlops are calculated with 512 × 512 inputs.

More Detailed Comparison with DCN. We carry out a more detailed comparison between DCN and our proposed GD-FAM. In particular, we replace GD-FAM or FAM with a simple concatenation followed by a deformable convolution, where GD-FAM and FAM are inserted in the last stage to align the last two features for comparison. The DCN directly replaces FAM or GD-FAM. As shown in Tab 16, our method achieves better results (1.0-2.0% mIoU gains) on both Cityscape dataset and UDS dataset, which share the same conclusion with the findings in Tab. 9 (d).

Domain Generalization Testing Using RobustNet [32]. We further prove the domain generalization ability of SFNet and SFNet-Lite. Our methods are based

Table 18: Effectiveness on ADE20k dataset on Semantic-FPN with different backbones.

Method	Backbone	mIoU	#GFLOP
Semantic-FPN (baseline)	ResNet50	37.6	72.8
SFNet	ResNet50	39.0	75.2
SFNet-Lite	ResNet50	38.8	74.2
Semantic-FPN (baseline)	DF2	34.5	35.2
SFNet	DF2	36.7	36.2
SFNet-Lite	DF2	36.8	35.9

Table 19: Effectiveness on ADE20k dataset on Transformer-based methods. B0 and B1 : backbones in SegFormer [55].

Method	Backbone	mIoU	#GFLops
SegFormer (baseline)	B0	37.4	8.4
SFNet-Lite	B0	38.2	9.4
SegFormer (baseline)	B1	40.9	16.0
SFNet-Lite	B1	42.2	17.6

on previous work RobustNet [32] and Semantic-FPN [19]. In particular, we follow the original open-source RobustNet code [1] and settings by the whitening operation in different backbones to build the baseline. As shown in Tab .17, our methods achieve consistent 2-3% mIoU improvements over the RobustNet baselines on both IDD and BDD datasets.

Experiment Results on ADE20k Dataset. In Tab. 18, we verify the effectiveness of FAM and GD-FAM on the more challenging dataset ADE20k. For a fair comparison, we re-implement the baseline in the same codebase and report our reproduced results for Semantic-FPN. As shown in that table, we find about 1.2%-2.2% improvements over different baselines. In particular, we find the improvements on the real-time model are stronger, which means the semantic gaps in small models are heavier. This finding is similar in the road driving scene datasets (see Tab. 6, Tab. 7).

Experiment Results on ADE20k Using Transformer-based Model. In Tab. 19, we also report the results using transformer-based model SegFormer [55]. We also find about 0.8-1.3% mIoU improvements over different backbones. These results indicate that our proposed approach can also be used in transformer-based segmenter.

5 Conclusion

In this paper, we propose to use the learned **Semantic Flow** to align multi-level feature maps generated by aligned feature pyramids for semantic segmentation. We propose a flow-aligned module to fuse high-level feature maps and low-level feature maps. Moreover, to speed up the inference procedure, we propose a novel Gated Dual flow alignment module to align both high and low-resolution feature maps directly. By discarding atrous convolutions to reduce computation overhead and employing the flow alignment module to enrich the semantic representation of low-level features, our network achieves the best trade-off between semantic segmentation accuracy and running time efficiency. Experiments on multiple challenging datasets illustrate the efficacy of our method. Moreover, we merge four challenging driving datasets into one Unified Driving Segmentation dataset (UDS), which contains various domains. We benchmark several works on the merged dataset. Experiment results show that our SFNet series can achieve the best speed and accuracy trade-off. In particular, our SFNet improves the original DFNet on the UDS dataset by a large margin (9.0% mIoU). These results indicate that our SFNet can be a faster and accurate baseline for Semantic Segmentation.

Acknowledgement We gratefully acknowledge the support of SenseTime Research for providing the computing resources to carry out this research. Without it, we could not perform benchmarking. This work was supported by the National Key Research and Development Program of China (No.2020YFB2103402). We also thank Qi Han (SenseTime) and Houlong Zhao (XiaoMi Car) for discussing the TensorRT deployment.

Data Availability Statement All the datasets used in this paper are available online. Cityscapes [2], BDD [3], IDD [4], Mapillary [5], and Camvid [6] can be downloaded from their official website accordingly. The UDS dataset is merged from these datasets.

References

1. Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
2. Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.
3. Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *ECCV*, 2018.
4. Jun Fu, Jing Liu, Haijie Tian, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, 2019.

[1] https://github.com/shachoi/RobustNet

[2] https://www.cityscapes-dataset.com/benchmarks/
[3] https://bdd-data.berkeley.edu/
[4] https://idd.insaan.iiit.ac.in/
[5] https://www.mapillary.com/dataset/vistas
[6] http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/

5. Yi Zhu, Karan Sapra, Fitsum A. Reda, Kevin J. Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *CVPR*, 2019.
6. Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.
7. Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018.
8. Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.
9. Xiangtai Li, Zhao Houlong, Han Lei, Tong Yunhai, and Yang Kuiyuan. Gff: Gated fully fusion for semantic segmentation. In *AAAI*, 2020.
10. Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, 2021.
11. Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In *CVPR*, 2021.
12. Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021.
13. Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. *arXiv preprint arXiv:1608.05442*, 2016.
14. Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *CVPR*, 2018.
15. Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.
16. Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.
17. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
18. Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.
19. Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollar. Panoptic feature pyramid networks. In *CVPR*, 2019.
20. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.
21. Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 2017.
22. Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *CVPR*, 2019.
23. Juncai Peng, Yi Liu, Shiyu Tang, Yuying Hao, Lutao Chu, Guowei Chen, Zewu Wu, Zeyu Chen, Zhiliang Yu, Yuning Du, et al. Pp-liteseg: A superior real-time semantic segmentation model. *arXiv preprint arXiv:2204.02681*, 2022.
24. Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *CVPR*, 2017.
25. Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.
26. Girish Varma, Anbumani Subramanian, Anoop Namboodiri, Manmohan Chandraker, and CV Jawahar. Idd: A dataset for exploring problems of autonomous navigation in unconstrained environments. In *WACV*, 2019.
27. Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017.
28. Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, 2020.
29. Xin Li, Yiming Zhou, Zheng Pan, and Jiashi Feng. Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In *CVPR*, 2019.
30. Xiangtai Li, Ansheng You, Zeping Zhu, Houlong Zhao, Maoke Yang, Kuiyuan Yang, and Yunhai Tong. Semantic flow for fast and accurate scene parsing. In *ECCV*, 2020.
31. Shihua Huang, Zhichao Lu, Ran Cheng, and Cheng He. Fapn: Feature-aligned pyramid network for dense image prediction. In *ICCV*, 2021.
32. Sungha Choi, Sanghun Jung, Huiwon Yun, Joanne T Kim, Seungryong Kim, and Jaegul Choo. Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In *CVPR*, 2021.
33. Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.
34. Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015.
35. Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
36. Maoke Yang, Kun Yu, Chi Zhang, Zhiwei Li, and Kuiyuan Yang. Denseaspp for semantic segmentation in street scenes. In *CVPR*, 2018.
37. Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2018.
38. Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *CVPR*, 2018.
39. Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *ECCV*, 2018.
40. Kaiyu Yue, Ming Sun, Yuchen Yuan, Feng Zhou, Errui Ding, and Fuxin Xu. Compact generalized non-local network. In *NeurIPS*, 2018.
41. Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Learning a discriminative feature network for semantic segmentation. In *CVPR*, 2018.

42. Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.
43. Yuhui Yuan and Jingdong Wang. Ocnet: Object context network for scene parsing. *IJCV*, 2021.
44. Hang Zhang, Han Zhang, Chenguang Wang, and Junyuan Xie. Co-occurrent features in semantic segmentation. In *CVPR*, 2019.
45. Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019.
46. Xiangtai Li, Hao He, Xia Li, Duo Li, Guangliang Cheng, Jianping Shi, Lubin Weng, Yunhai Tong, and Zhouchen Lin. Pointflow: Flowing semantics through points for aerial image segmentation. In *CVPR*, 2021.
47. Hao He, Xiangtai Li, Guangliang Cheng, Jianping Shi, Yunhai Tong, Gaofeng Meng, Véronique Prinet, and LuBin Weng. Enhanced boundary learning for glass-like object segmentation. In *ICCV*, 2021.
48. Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017.
49. Henghui Ding, Xudong Jiang, Bing Shuai, Ai Qun Liu, and Gang Wang. Context contrasted feature and gated multi-scale aggregation for scene segmentation. In *CVPR*, 2018.
50. Towaki Takikawa, David Acuna, Varun Jampani, and Sanja Fidler. Gated-scnn: Gated shape cnns for semantic segmentation. In *ICCV*, 2019.
51. Zilong Huang, Yunchao Wei, Xinggang Wang, Humphrey Shi, Wenyu Liu, and Thomas S Huang. Alignseg: Feature-aligned segmentation networks. *TPAMI*, 2021.
52. Hanzhe Hu, Yinbo Chen, Jiarui Xu, Shubhankar Borse, Hong Cai, Fatih Porikli, and Xiaolong Wang. Learning implicit feature alignment function for semantic segmentation. *ECCV*, 2022.
53. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.
54. Haobo Yuan, Xiangtai Li, Yibo Yang, Guangliang Cheng, Jing Zhang, Yunhai Tong, Lefei Zhang, and Dacheng Tao. Polyphonicformer: Unified query learning for depth-aware video panoptic segmentation. *ECCV*, 2022.
55. Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *NeurIPS*, 2021.
56. Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021.
57. Wenwei Zhang, Jiangmiao Pang, Kai Chen, and Chen Change Loy. K-net: Towards unified image segmentation. In *NeurIPS*, 2021.
58. Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *ECCV*, 2018.
59. Hanchao Li, Pengfei Xiong, Haoqiang Fan, and Jian Sun. Dfanet: Deep feature aggregation for real-time semantic segmentation. In *CVPR*, 2019.
60. Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *ECCV*, 2018.
61. Sachin Mehta, Mohammad Rastegari, Linda Shapiro, and Hannaneh Hajishirzi. Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In *CVPR*, 2019.
62. Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *ECCV*, 2018.
63. Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *IJCV*, 2021.
64. Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *CVPR*, 2019.
65. Yiheng Zhang, Zhaofan Qiu, Jingen Liu, Ting Yao, Dong Liu, and Tao Mei. Customizable architecture search for semantic segmentation. In *CVPR*, 2019.
66. Mingyuan Fan, Shenqi Lai, Junshi Huang, Xiaoming Wei, Zhenhua Chai, Junfeng Luo, and Xiaolin Wei. Rethinking bisenet for real-time semantic segmentation. In *CVPR*, 2021.
67. Haiyang Si, Zhiqiang Zhang, Feifan Lv, Gang Yu, and Feng Lu. Real-time semantic segmentation via multiply spatial fusion network. *arXiv preprint arXiv:1911.07217*, 2019.
68. Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019.
69. Yanwei Li, Xinze Chen, Zheng Zhu, Lingxi Xie, Guan Huang, Dalong Du, and Xingang Wang. Attention-guided unified network for panoptic segmentation. In *CVPR*, 2019.
70. Yifeng Chen, Guangchen Lin, Songyuan Li, Omar Bourahla, Yiming Wu, Fangfang Wang, Junyi Feng, Mingliang Xu, and Xi Li. Banet: Bidirectional aggregation network with occlusion handling for panoptic segmentation. In *CVPR*, 2020.
71. Lorenzo Porzi, Samuel Rota Bulo, Aleksander Colovic, and Peter Kontschieder. Seamless scene segmentation. In *CVPR*, 2019.
72. Yibo Yang, Hongyang Li, Xia Li, Qijie Zhao, Jianlong Wu, and Zhouchen Lin. Sognet: Scene overlap graph network for panoptic segmentation. In *AAAI*, 2020.
73. Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *CVPR*, 2019.
74. Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *CVPR*, 2021.
75. Rui Hou, Jie Li, Arjun Bhargava, Allan Raventos, Vitor Guizilini, Chao Fang, Jerome Lynch, and Adrien Gaidon. Real-time panoptic segmentation from dense detections. In *CVPR*, 2020.
76. Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020.
77. Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*, 2020.
78. Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient

convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
79. Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.
80. Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018.
81. John Lambert, Zhuang Liu, Ozan Sener, James Hays, and Vladlen Koltun. MSeg: A composite dataset for multi-domain semantic segmentation. In *CVPR*, 2020.
82. Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Simple multi-dataset detection. In *CVPR*, 2022.
83. Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. *ICLR*, 2022.
84. Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE TKDE*, 2022.
85. Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *ECCV*, 2018.
86. Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *ICCV*, 2019.
87. Jin Kim, Jiyoung Lee, Jungin Park, Dongbo Min, and Kwanghoon Sohn. Pin the memory: Learning to generalize semantic segmentation. In *CVPR*, 2022.
88. Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 2011.
89. Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. *ECCV*, 2004.
90. Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *CVPR*, 2015.
91. Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *NeurIPS*, 2015.
92. Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.
93. Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 2015.
94. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.
95. Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. *ECCV*, 2020.
96. Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin, and Hong Liu. Expectation-maximization attention networks for semantic segmentation. *ICCV*, 2019.
97. Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016.
98. Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NeurIPS*, 2017.
99. Eduardo Romera, Jose M. Alvarez, Luis Miguel Bergasa, and Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE-TITS*, 2018.
100. Yuval Nirkin, Lior Wolf, and Tal Hassner. Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation. In *CVPR*, 2021.
101. Yuanduo Hong, Huihui Pan, Weichao Sun, and Yisong Jia. Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes. *IEEE TITS*, 2022.
102. Yanran Wu, Xiangtai Li, Chen Shi, Yunhai Tong, Yang Hua, Tao Song, Ruhui Ma, and Haibing Guan. Fast and accurate scene parsing via bi-direction alignment networks. In *ICIP*, 2021.
103. Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2008.
104. Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.
105. Qizhu Li, Xiaojuan Qi, and Philip HS Torr. Unifying training and inference for panoptic segmentation. In *CVPR*, 2020.
106. Yanwei Li, Hengshuang Zhao, Xiaojuan Qi, Liwei Wang, Zeming Li, Jian Sun, and Jiaya Jia. Fully convolutional networks for panoptic segmentation. In *CVPR*, 2021.