SlimSeg: Slimmable Semantic Segmentation with Boundary Supervision

Danna Xue
danna_xue@mail.nwpu.edu.cn
Northwestern Polytechnical University & Computer Vision Center, Universitat Autònoma de Barcelona
Xi'an, China

Fei Yang
fyang@cvc.uab.es
Computer Vision Center, Universitat Autònoma de Barcelona
Barcelona, Spain

Pei Wang
wangpei23@mail.nwpu.edu.cn
Northwestern Polytechnical University
Xi'an, China

Luis Herranz
lherranz@cvc.uab.es
Computer Vision Center, Universitat Autònoma de Barcelona
Barcelona, Spain

Jinqiu Sun*
sunjinqiu@nwpu.edu.cn
Northwestern Polytechnical University
Xi'an, China

Yu Zhu
yuzhu@nwpu.edu.cn
Northwestern Polytechnical University
Xi'an, China

Yanning Zhang
ynzhang@nwpu.edu.cn
Northwestern Polytechnical University
Xi'an, China

ABSTRACT

Accurate semantic segmentation models typically require significant computational resources, inhibiting their use in practical applications. Recent works rely on well-crafted lightweight models to achieve fast inference. However, these models cannot flexibly adapt to varying accuracy and efficiency requirements. In this paper, we propose a simple but effective slimmable semantic segmentation (SlimSeg) method, which can be executed at different capacities during inference depending on the desired accuracy-efficiency tradeoff. More specifically, we employ parametrized channel slimming by stepwise downward knowledge distillation during training. Motivated by the observation that the differences between segmentation results of each submodel are mainly near the semantic borders, we introduce an additional boundary guided semantic segmentation loss to further improve the performance of each submodel. We show that our proposed SlimSeg with various mainstream networks can produce flexible models that provide dynamic adjustment of computational cost and better performance than independent models. Extensive experiments on semantic segmentation benchmarks, Cityscapes and CamVid, demonstrate the generalization ability of our framework.

*Corresponding author.

MM '22, October 10–14, 2022, Lisboa,Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3548191

CCS CONCEPTS

• **Computing methodologies** → **Scene understanding**; **Image segmentation**.

KEYWORDS

Efficient semantic segmentation; Slimmable neural network; Knowledge distillation; Boundary detection

ACM Reference Format:
Danna Xue, Fei Yang, Pei Wang, Luis Herranz, Jinqiu Sun, Yu Zhu, and Yanning Zhang. 2022. SlimSeg: Slimmable Semantic Segmentation with Boundary Supervision. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3503161.3548191

1 INTRODUCTION

Semantic segmentation predicts the semantic category corresponding to each pixel in an image. Various applications have benefited from advances towards more accurate results, such as autonomous driving [3, 11, 13, 17, 26–29, 32, 52, 53, 59, 60], image synthesis and manipulation [37, 47], and medical imaging [25, 39]. Based on the pioneering fully convolutional network [35], previous studies have made important achievements by greatly increasing the performance on various challenging semantic segmentation benchmarks [1, 5, 9, 64]. Despite their superiority, these powerful models, built upon heavy deep neural networks, suffer from the low inference speed and strict requirements for computing devices.

Most of the existing works mainly address efficient semantic segmentation through (i) designing compact backbone architectures [11, 17, 26, 27, 40, 44, 53, 61], (ii) effective model compression methods [3, 28, 30, 32, 38, 60], (iii) exploiting reliable context and boundary information [13, 29, 34, 41, 52]. However, those methods mainly speed up the inference with fixed network structures, while

Table 1: The FLOPs and number of parameters of semantic segmentation networks (except for the backbone) and their proportions of the whole model, with image size 1024×2048.

Networks	SFNet [29]		DeepLabv3+ [2]	
Backbone	ResNet50	ResNet18	ResNet50	MobileNetv2
GFLOPs	436.3 | 72%	107.5 | 55%	663.5 | 45%	6.3 | 34%
Params	7.7M | 25%	1.5M | 12%	16.8M | 40%	2.7M | 59.3%

in practice, the equipped resources are quite different across diverse devices. Even for the same device, the availability of hardware resources varies over time. Suppose we want to switch between models of different sizes according to the ideal accuracy-efficiency tradeoff. One straightforward way is to train multiple independent models with different structures and parameters and load a specific one during inference. However, it requires a longer training time and more memory for storage. Unlike previous works, we focus on improving the flexibility of the semantic segmentation model.

The recent work [57] proposed a slimmable neural network that can adjust the width of the network for different inference speeds. However, they mainly focus on image classification and only apply their slimmable models as backbones on instance segmentation tasks, while the other parts (*e.g.*, the decoder) are non-slimmable. Due to the resolution of the output image, even if a relatively simple structure is used in the decoder part, including up-sampling and multi-level feature aggregation *etc.*, the decoder still requires a large amount of computation during inference. We show the computation cost (in FLOPs) and the number of parameters of several mainstream segmentation models, including SFNet [29] and DeepLabv3+ [2], in Table 1. In these models, the Pyramid Pooling Module (PPM) [62] and the decoder account for more than one-third of the overall calculation, while the parameters for most of them are the minority of the whole model. Based on [57], we focus on semantic segmentation and aim to lower computational cost from the perspective of reducing the overall size of the network, rather than just backbones. Motivated by this, we propose a slimmable semantic segmentation network (SlimSeg) that leverages the slimming mechanism to dynamically adjust the channel of features on every single layer. The network's capacity can be switched with the size of width according to the computational requirements, thereby controlling the trade-off between accuracy and inference time. In addition, we apply stepwise downward inplace distillation for training smaller subnetworks, which means that smaller subnetworks are learned from the larger ones. This leads to consistent results between different submodels.

Moreover, we also found that the differences between the predicted results of slimmable subnetworks with different widths mainly exist along the semantic boundaries. Previous works [58, 66] also report that most existing segmentation models fail to make right predictions along the semantic boundaries. To further improve the segmentation quality on the boundary and narrow the accuracy gap between each subnetwork, we introduce a semantic boundary detection head on the low-level features and additional supervision named semantic boundary guided loss. This loss leverages the predicted boundaries as guidance to calculate a weighted bootstrapped cross-entropy. The boundary detection head can be removed during inference, so it does not introduce any additional computation.

Our SlimSeg is a general scheme that can adapt the existing segmentation models to width switchable models without any new structural design. The experimental results on Cityscapes [5] and CamVid [1] based on SFNet [29] and DeepLabv3+ [2] demonstrate the slimmable model has comparable accuracy to independent models. Furthermore, our method shows higher accuracy on smaller subnetworks with the stepwise downward distillation and proposed boundary guided loss. The contributions are summarized as follows:

- We propose a simple but effective slimmable semantic segmentation method (SlimSeg) which can adjust the capacity of the model depending on the desired trade-off between accuracy and efficiency.
- We present the boundary supervision, including a low-level boundary detection head and a boundary guided loss to improve the accuracy of semantic segmentation in boundary regions, especially for the smaller subnetworks.
- Extensive experiments and analysis indicate the efficacy and generalization ability of our proposed method, both quantitatively and qualitatively.

2 RELATED WORKS

2.1 Generic Semantic Segmentation

A typical semantic segmentation architecture generally includes two parts: encoder and decoder. The encoder module extracts image features through convolution and downsampling. Generally, the encoder is adapted from image classification models trained on ImageNet [6], such as VGG19 [42], ResNet [2], *etc.* Since semantic segmentation conduct pixel-level classification, the typical fully connected layers are replaced by convolutional layers [35]. To utilize the global context, the Pyramid Pooling Module (PPM) [2, 62] is employed to increase the receptive field without an increase in parameters. However, massive computations are introduced by PPM and other feature fusion modules performed on high-resolution features neighbor to the output. To pursue better global and local feature fusion, models [43, 63] based on more powerful backbones, such as HRNet [46] and ViT [8], have been proposed. These models have achieved higher accuracy, but are limited by the hardware requirements in practice. Our approach takes advantage of the sophisticated models and achieves variable capacity through width slimming, enabling fast inference while maintaining accuracy.

2.2 Efficient Semantic Segmentation

Efficient semantic segmentation needs to consider both accuracy and computational cost. Existing methods trade accuracy and speed along three different lines.

Hand-crafted compact backbone architecture. An effective backbone can greatly improve the upper bound of performance. The works [11, 17, 26, 27, 40, 44, 53, 61] design lightweight backbone architectures from scratch to pursue more efficient inference. Some works [17, 27, 61] devised multiscale image cascades and feature fusion mechanisms to achieve a good accuracy-speed trade-off. Others [26] improve existing network layers to create sufficient receptive field and densely utilize the contextual information. BiSeNet

[53] introduced a shallow spatial branch to process full resolution images while learning context information by a deep branch.

Machine-driven architecture optimization. Neural Architecture Search (NAS) [67] is an effective technique to switch the labor-intensive architecture design to an automatic machine-driven optimization process, and this technique has been applied to semantic segmentation in recent years. From repeated cell structure [38, 60] to more flexible network structure [28], different types of network (*e.g.*, graph convolution network [32]), or explicitly taking latency into consideration [3, 30]. FasterSeg [3] introduces the teacher-student co-searching and flexible multi-resolution branches aggregation structure. Although the latitude of the search space is continuously improved [59], it still requires longer training time and more effective search strategies.

Feature mining and aggregation. By exploiting the potential of existing lightweight models, rather than building new architectures, these methods learn more favorable context information. Knowledge distillation [16] has shown its effectiveness on segmentation tasks by improving the accuracy of a lightweight student model and speed-up its convergence by transferring learned knowledge from a sophisticated teacher network. Liu *et al.* [34, 41] provide a comprehensive analysis of feature distillation at different levels, from various cumbersome models to compact models. Others investigate multi-level feature aggregation to alleviate the side effects of up and down sampling [29] or enlarge the receptive field of lightweight networks [13, 52].

Although these efficient semantic segmentation approaches improve the accuracy-efficiency tradeoff from different perspectives, the resulting model is still limited to fixed size and operating at a single tradeoff. Unlike these methods, we enable adjustable computation with one single model and ensures good accuracy for each submodel of different size.

2.3 Dynamic Neural Networks

Dynamic neural networks [14] reduce average inference cost by adaptively changing characteristics of the computational graph, including the resolution, depth, and width. Reducing the **resolution** of the input image is the most straightforward way to lower computational costs. For images with relatively simple context, equivalent prediction accuracy can be achieved with lower resolutions. Some works [49, 51, 65] propose parallel training for multi-resolution inference with a single model. Networks with dynamic **depth** speed up inference by skipping residual blocks adaptively [31, 45, 48] or early exiting when shallower subnetworks have high enough confidence [21, 23, 51]. The number of feature channels, *i.e.* **width**, is also a key factor of efficiency. One way of enabling various channel inference is dynamic pruning. By identifying and skipping the insignificant channels during inference [12, 20, 24] or training a hypernetwork to select the filters [4], the channel complexity can be lessened. Moreover, [55–57] propose slimmable neural networks with embedded submodels sharing parameters that are executable at different widths, allowing immediate and adaptive accuracy-efficiency trade-offs at runtime. Based on the success of slimmable neural network, Liang *et al.* [24] improve the hardware efficiency by introducing a dynamic slimming gate that adaptively adjusts the network width with negligible extra computation cost. Although dynamic neural networks have shown their effectiveness on strategically allocating appropriate computational resources, most works still focus on image classification and some other low-level vision tasks, such as image compression [50], denoising [22] and image generation [18]. Different from previous works, we study dynamic semantic segmentation models through our analysis.

3 METHOD

3.1 Slimmable Segmentation Framework

Image semantic segmentation requires assigning a category label to each pixel in the image from several semantic categories. Given an image x, a segmentation network $\mathcal{S}$ parameterized by θ implements a mapping $p = \mathcal{S}(x;\theta)$, where each spatial element of p is a probability vector indicating the probability of each semantic category, from which the most probable is selected. Ideally, it should correspond to the category indicated in the corresponding ground truth segmentation map y (coded as one-hot probability vectors per pixel). During training, the loss minimized is the cross-entropy $\mathcal{L}_{CE}(p, y; \theta)$ between the predicted probability and the ideal one-hot label. In practice, this loss is averaged over the pixels in the image and the image-segmentation pairs (x, y) in the training dataset.

In this work, we propose a flexible semantic segmentation framework, named as SlimSeg, which can adapt its model capacity during inference via the slimming mechanism to accommodate various levels of computing power. More specifically, we define different sets of widths (*i.e.* number of channels in each convolutional layer) of the segmentation network. Thus, the segmentation network contains N subnetworks with parameters $\{\theta_{w_1}, \theta_{w_2}, ..., \theta_{w_N}\}$ with N increasing widths $w_1 < w_2 < ... < w_N$, respectively. For every convolutional layer implementing slimming, the parameters are built as subsets of larger (sub)networks as $\theta_{w_1} \subset \theta_{w_2} \subset ... \subset \theta_{w_N} = \theta$. Then, the objective of our task becomes optimizing all the subnetworks with $\sum_{n=1}^{N} \mathcal{L}_{CE}(p^n, y; \theta_{w_n})$, where p^n is the predicted category probability vector of the n^{th} subnetworks with parameters θ_{w_n}. The loss is also averaged over pixels and training data, and then minimized over the parameters θ. Note that we could also replace the (one-hot) ground truth label y with the soft label $p_{n'}$ predicted by larger subnetworks to distill its knowledge. We describe our loss functions in more detail in Section 3.2 and 3.3. Henceforth, we will also omit the explicit dependencies on the model parameters for the sake of simplicity.

The overall pipeline of our SlimSeg is illustrated in Figure 1. We deploy width slimming on the entire network, including the encoder for feature extraction, the Pyramid Pooling Module [62] and the decoder for feature aggregation and classification. The number of channels is adjusted through the slimmable convolutional layer [57], which produces different output feature channels by adjusting the number of filters. The slimmable convolution will result in a different output feature distribution. Following [57], we use independent batch normalization (BN) layers for each width, which only introduces very few parameters to the overall model.

3.2 Stepwise Downward Distillation

To utilize the knowledge learned by large submodels to guide the learning of the smaller submodels, we apply inplace knowledge

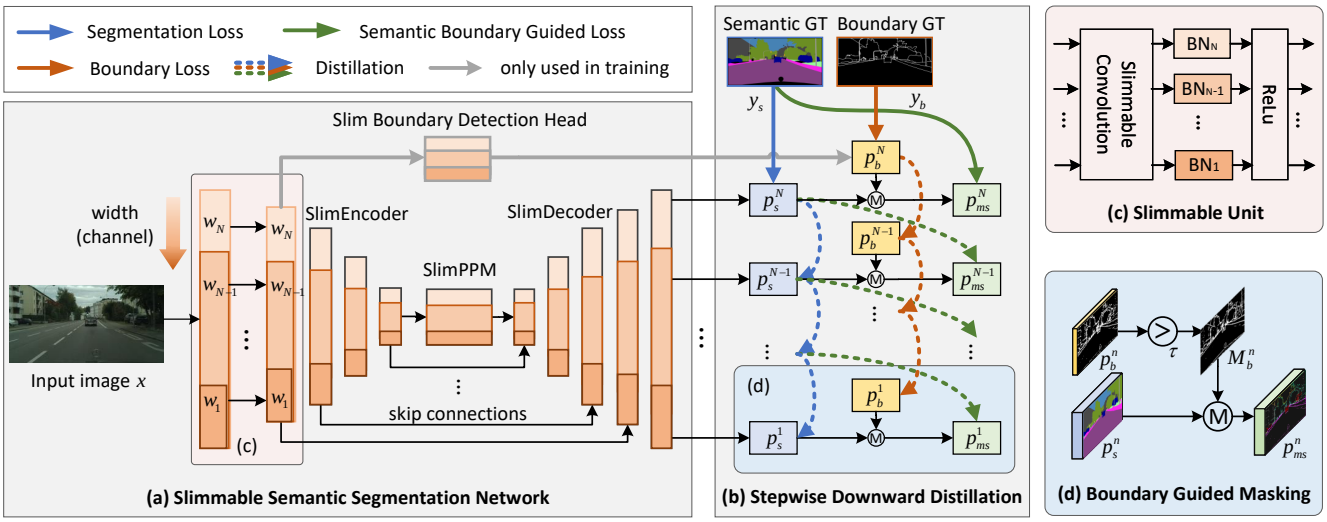


Figure 1: Overview of our slimmable semantic segmentation framework. (a) The whole network, including the encoder, PPM, decoder and boundary detection head, is slimmable. The boundary detection head can be removed during inference. (c) Each slimmable unit includes a slimmable convolution layer, independent BNs for each width and a ReLu layer. (b) The largest network with width w_N is supervised by the ground truth labels, and the smaller models with width w_n are learned from larger models with width w_{n+1} by stepwise distillation. (d) The predicted boundaries are used to generate boundary masked probability maps for calculating the boundary guided loss.

distillation from larger (sub)networks to smaller ones. Unlike previous knowledge distillation on segmentation [34, 41], we do not learn from an already trained (fixed) sophisticated model to improve another independent compact model. We introduce stepwise downward inplace distillation, where class probabilities estimated from the larger subnetwork are used as soft targets for training the next smaller subnetwork. The largest subnetwork is supervised by the ground truth labels. Note that the parameters of a smaller subnetwork are also a subset of larger ones, which means that the smallest subnetwork will learn the most important features implicitly to guarantee the accuracy of larger submodels. This leads to the following loss function:

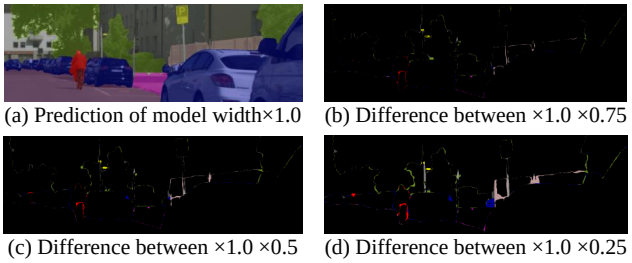
$$\mathcal{L}_{seg} = \mathcal{L}_{CE}(p_s^N, y_s) + \sum_{n=1}^{N-1} \mathcal{L}_{KD}(p_s^n, p_s^{n+1}), \tag{1}$$

where $\mathcal{L}_{CE}$ denotes the cross entropy loss, and p_s^n, y_s are the segmentation probability map predicted by the n^{th} submodel and the ground truth semantic label, respectively. Instead of computing the Kullback-Leibler divergence between two probabilities, we use soft target cross-entropy loss (we denote it as $\mathcal{L}_{KD}$ to distinguish it from $\mathcal{L}_{CE}$, which applied with ground truth supervision). We found that the cross-entropy between two probabilities is more stable during training than the Kullback-Leibler divergence, which is also a common setting for the knowledge distillation in [34, 55–57].

In practice, stopping the gradients of the supervising tensor predicted by the larger width is necessary, so that the loss of a subnetwork will never back-propagate through the computation graph to larger subnetworks. We performed experiments on the effectiveness of distillation and the type of optimal teachers. The results show that using the probability map predicted by previous subnetworks as the soft target can lead to better performance. For more details, see Section 4.3.

3.3 Semantic Boundary Guided Loss

Based on the training framework and distillation method presented above, we can already obtain varying amounts of computation of multiple subnetworks with partially shared parameters. To further improve the performance, especially for the smaller subnetworks, we compare the semantic labels predicted by different subnetworks trained only with the loss $\mathcal{L}_{seg}$. As illustrated in Figure 2, the differences between the segmentation results of subnetworks with different widths are mainly near the borders between different semantic categories. Moreover, as the width decreases, the gap between the predictions gets larger.



(a) Prediction of model width×1.0 (b) Difference between ×1.0 ×0.75

(c) Difference between ×1.0 ×0.5 (d) Difference between ×1.0 ×0.25

Figure 2: Difference between submodels. (a) Predicted semantic map of the ×1.0 model. (b)-(d) Difference map between the smaller submodels and the ×1.0 model, where the consistent (inconsistent) predicted pixels are shown as black background (ground truth color codes). Better view in color.

Motivated by this observation, we introduce extra boundary supervision to improve the accuracy in those regions, especially for small subnetworks. Specifically, we introduce an additional boundary detection head with a simple structure, including a slimmable unit (Figure 1 (c)) and a slimmable convolution layer with kernel size 1 followed by a sigmoid layer, on the low-level features. The output of this head p_b^N is supervised by the binary boundary masks generated by the semantic segmentation ground truth labels y_b. The pixels within 3 pixels from the semantic border are marked as boundary regions. We apply binary cross-entropy loss to constrain boundary detection with:

$$\mathcal{L}_b = \mathcal{L}_{BCE}(p_b^N, y_b) + \sum_{n=1}^{N-1} \mathcal{L}_{KD}(p_b^n, p_b^{n+1}), \tag{2}$$

where we also leverage knowledge distillation to subnetworks with the soft boundary labels predicted by the larger one, except for the largest width that uses the boundary ground truth y_b. Unlike [7], our boundary detection head is used only on training and can be removed during inference, so it does not introduce any extra computation. The head helps enhance the low-level features of boundary regions.

Besides, the estimated boundary also perform as a reference to resample the misclassified pixels on the border to calculate the boundary guided segmentation loss, which can be regarded as a hard sample mining strategy. As shown in Figure 1 (d), taking the boundary probability map p_b predicted by the boundary detection head, we generate a confidence binary mask M_b to locate those pixels which might be situated near to semantic boundaries:

$$M_b(u,v) = \begin{cases} valid, & p_b(u,v) > \tau \\ invalid, & otherwise \end{cases}. \tag{3}$$

The values in M_b are element-wise calculated by comparing the boundary confidential score p_b at each location (u, v) with a predefined threshold τ. We empirically set τ to 0.7 in our experiments. Only valid pixels are included in the loss calculation. Similar to $\mathcal{L}_{seg}$, the cross-entropy loss and the knowledge distillation loss of the masked semantic probabilities $p_{ms}^n = M_b^n(p_s^n), n \in \{1, 2, \cdots, N\}$ are calculated with:

$$\mathcal{L}_g = \mathcal{L}_{CE}(p_{ms}^N, y_s) + \sum_{n=1}^{N-1} \mathcal{L}_{KD}(p_{ms}^n, p_s^{n+1}). \tag{4}$$

Then, the loss function for training our SlimSeg is calculated as a summation of the semantic segmentation loss $\mathcal{L}_{seg}$, boundary detection loss $\mathcal{L}_b$ and the boundary guided segmentation loss $\mathcal{L}_g$:

$$\mathcal{L}_{full} = \mathcal{L}_{seg} + \lambda_1 \mathcal{L}_b + \lambda_2 \mathcal{L}_g \tag{5}$$

where λ_1, λ_2 are hyperparameters, which are set to 10 and 1 in our experiments, respectively.

Finally, to clarify the training procedure of our proposed SlimSeg, we provide a Pytorch-style pseudo-code in Algorithm 1.

Algorithm 1 Slimmable semantic segmentation

Ensure: Dataset $\mathcal{D}$, width list $\mathcal{W} = \{w_1, w_2, ..., w_N\}$
Require: Slimmable semantic segmentation network $\mathcal{S}$
for $i = 1, 2, ..., iteration$ **do**
  Get a mini-batch of image x, semantic label y_s, boundary label y_b from $\mathcal{D}$.
  Clear gradients of weights, $optimizer.zerograd()$.
  for w in $sorted(\mathcal{W}, reverse = True)$ **do**
    Switch the BN layers to current width.
    Execute current subnetwork, $p_s, p_b = \mathcal{S}(x; \theta_w)$.
    Compute the masked probability, $p_{ms} = M_b(p_s)$.
    if $w = w_N$ **then**
      Compute loss with ground truth,
      $loss = CE(p_s, y_s) + BCE(p_b, y_b) + CE(p_{ms}, y_s)$.
    else
      Compute distillation loss,
      $loss = KD(p_s, y_s^t) + KD(p_b, y_b^t) + KD(p_{ms}, y_s^t)$.
    end if
    if $w > w_1$ **then**
      Save predicted probability p_s, p_b as teachers y_s^t, y_b^t.
    end if
    Compute gradients, $loss.backward()$.
  end for
  Update weights, $optimizer.step()$.
end for
return $\mathcal{S}$

4 EXPERIMENTS

4.1 Benchmarks and Evaluation Metrics

Cityscapes. Cityscapes [5] is a first-person perspective street-scene dataset with 19 semantic categories, 5000 fine annotated images with 2,975, 500 and 1,525 images for training, validation and testing, respectively. The high resolution of the images (1024×2048 pixels) poses a great challenge to real-time semantic segmentation. For a fair comparison, we only use the fine annotated images for training.

CamVid. CamVid [1] is a road scene dataset from the perspective of a driving automobile. It consists of 367, 101 and 233 images for training, validation and testing with resolution 720×960. Following the pioneering work [10, 53], we use the subset of 11 semantic classes from the 32 provided categories for a fair comparison with existing methods. The pixels out of the selected classes are ignored.

Evaluation Metrics. For quantitative evaluation, we report the mean of class-wise intersection-over-union (mIoU) for accuracy comparison. The floating-point operations per second (FLOPs) and frames per second (FPS) are adopted for efficiency comparison. Besides, we also give the number of parameters for model size.

4.2 Implementation Details

Training. We use the stochastic gradient descent (SGD) algorithm to train our models with the batch size of 8, stochastic momentum of 0.9 and weight decay of 5e-4. As a common practice, the "poly" learning rate strategy in which the initial rate is multiplied by $\left(1 - \frac{iter}{iter_{max}}\right)^{power}$ at each iteration with the power of 0.9. All the models are trained for 100K iterations with an initial learning rate of 0.01 and Online Hard Example Mining (OHEM) [33] on two NVIDIA GeForce 3090Ti GPUs with CUDA 11.0, CUDNN 8.0 by Pytorch 1.7. Data augmentation includes random horizontal flip, random resizing with the scale range of [0.5, 2.0], and random cropping to 768×768 for Cityscapes and 720×720 for CamVid.

Inference. For inference, we use the whole image as an input to report performance, unless explicitly mentioned. Evaluation tricks such as sliding window inference and multiscale testing are not adopted. The measurement of inference time is executed on a single NVIDIA GeForce 2080Ti with CUDA 10.1, CUDNN 7.0, and we report the FPS without TensorRT acceleration.

Architectures. We conduct the experiments based on two mainstream semantic segmentation networks: SFNet [29] and DeepLabv3+ [2]. SFNet is based on the Feature Pyramid Network [33] architecture with a backbone network pre-trained on ImageNet classification [6] as encoder, a pyramid pooling module and a decoder aggregating multi-level features from the encoder. Similarly, DeepLabv3+ [2] includes a feature encoder, an atrous spatial pyramid pooling module and a simple decoder with only several convolutional layers and upsampling. For SFNet, we use the slimmable ResNet50 [57] pre-trained on ImageNet [6], and slimmable ResNet18, DFNetV1, DFNetV2 [30] without pre-training as encoder. For DeepLabv3+, we report the results using the slimmable ResNet50 and MobileNetv2 [57] (both are pre-trained on ImageNet) as encoder. The input of the boundary detection head is the low level features output by the second stage of the backbones. The resolution of the input features are down-sampled 4 times compared to the original image. We apply four width multipliers $[0.25, 0.5, 0.75, 1.0]$ in our experiments, except for Deeplabv3+-MobileNetv2 with $[0.35, 0.5, 0.75, 1.0]$.

4.3 Ablation Study

We conduct ablation experiments to validate the effectiveness of our width slimming training scheme, knowledge distillation method and the proposed boundary guided loss.

Width Slimming Training Scheme. We compare the slimmable model with their independently trained counterparts to demonstrate the effectiveness of the width slimming segmentation training scheme. The independent models have the same architecture as the slimmable subnetworks, but can only operate on a single width. Note that both the independent and slimmable models are trained with the loss $\mathcal{L}_{full}$ in Eq.5 for fair comparison, and the independent models are supervised by ground truth. We report the mIoU, number of parameters (M) and FLOPs (GMac) in Table 2. The slimmable models outperform the independent models of all width on SFNet (ResNet50, ResNet18) and DeepLabv3+ (ResNet50, MobileNetv2), while for SFNet (DFNetv, DFNetv2), the larger independent models are better than the slimmable one. We think this is because DFNet [19] is a compact backbone designed for best speed accuracy trade-off by neural architecture search, which have very little space to be compressed. Therefore, the gap between slimmable SFNet-DFNets submodels with different widths is also larger than ResNets. In terms of the amount of computation, with about 56% of the whole FLOPs, the submodel with width×0.75 achieves comparable performance as the full model. Besides, a slimmable model saves about 50% memories for storing the parameters compared with several independent models, and number will increase if we have more switchable width.

Stepwise Downward Distillation. To make the most of the knowledge learned by larger submodels, we test different distillation settings and demonstrate the effectiveness of our distillation method.

Table 2: Comparison of independent and slimmable models on Cityscapes *val*. Bold numbers indicate the better mIoUs.

Network	Width	Independent mIoU	Independent Param	Slimmable mIoU	Slimmable Param	FLOPs
SFNet ResNet50	×1.0	78.3	31.20	**78.4** (0.1↑)	31.29	607.9
	×0.75	77.3	17.57	**77.9** (0.6↑)		343.4
	×0.5	76.3	7.82	**77.4** (1.1↑)		153.9
	×0.25	73.2	1.97	**74.4** (1.2↑)		39.4
SFNet ResNet18	×1.0	75.0	12.87	**75.6** (0.6↑)	12.89	243.4
	×0.75	74.0	7.24	**74.8** (0.8↑)		137.4
	×0.5	71.4	3.22	**72.5** (1.1↑)		61.5
	×0.25	65.5	0.79	**67.3** (1.8↑)		15.7
SFNet DFNetv2	×1.0	**73.6**	17.88	73.1 (0.5↓)	17.91	80.2
	×0.75	**71.4**	10.06	71.1 (0.3↓)		45.2
	×0.5	**70.0**	4.48	69.8 (0.2↓)		20.2
	×0.25	62.5	1.12	**64.2** (1.7↑)		5.2
SFNet DFNetv1	×1.0	**70.0**	8.42	69.4 (0.6↓)	8.44	32.8
	×0.75	**67.8**	4.74	67.0 (0.8↓)		18.6
	×0.5	65.0	2.11	**65.3** (0.3↑)		8.4
	×0.25	57.8	0.52	**59.8** (2.0↑)		2.2
DeepLabv3+ ResNet50	×1.0	78.0	40.35	**78.4** (0.4↑)	40.44	1463
	×0.75	77.6	22.71	**78.2** (0.6↑)		824.3
	×0.5	76.7	10.11	**77.6** (0.9↑)		347.6
	×0.25	74.0	2.54	**75.6** (1.6↑)		92.9
DeepLabv3+ MobileNetv2	×1.0	66.9	4.53	**67.9** (1.0↑)	4.58	18.5
	×0.75	63.3	2.57	**67.0** (3.7↑)		12.2
	×0.5	58.6	1.16	**64.3** (5.7↑)		5.7
	×0.35	56.1	0.57	**61.1** (5.0↑)		3.3

Does inplace knowledge distillation work? We compare the mIoUs of training the slimmable model with and without stepwise downward distillation in Table 3. For the smallest subnetwork with width×0.25, the mIoUs consistently improve with distillation under all combinations of loss functions. With the distillation strategy proposed by our work, mIoUs improve on all subnetworks, and among them, the smallest subnetwork with width×0.25 has the largest increase (0.8%) from 73.6% to 74.4%.

Which is the best teacher for small submodels? We train our slimmable model with soft targets predicted by different models as teachers in knowledge distillation. For the student subnetwork $\mathcal{S}(\theta_{w_n})$, 'prev', 'largest', 'mean' indicates that the soft target is the predicted probability p^{n+1} of the last larger subnetwork $\mathcal{S}(\theta_{w_{n+1}})$, p^N of the largest subnetwork $\mathcal{S}(\theta_{w_N})$ [56] and the average of all the predictions $\frac{1}{N-n}\sum_{j=n+1}^{N} p^j$ by the subnetwork larger than the current model $\mathcal{S}(\theta_{w_{n+1}}), ..., \mathcal{S}(\theta_{w_N})$, respectively. Different from the setting of 'mean', 'larger' represents using the average loss of all the larger submodels' distillation. The mIoU of our slimmable model under different teacher settings are reported in Table 4. Note that all the models are trained with the sum of the three losses proposed. Our 'prev' setting, the stepwise downward distillation, outperform others by higher mIoU 74.4% and 77.37% on width ×0.25 and ×0.5. Using the average loss of all larger submodels results in better

mIoUs on the larger submodels with width ×0.75 and ×1.0, but even lower mIoU than models trained without distillation on width ×0.25 and ×0.5. The results are consistent with the phenomenon that student network's performance degrades when the gap between student and teacher is too large [36].

Table 3: Ablation of knowledge distillation (KD) with different loss function by Slim-SFNet-ResNet50 on Cityscapes *val*.

KD	GT			Soft Target			mIoU (%)			
	$\mathcal{L}_{seg}$	$\mathcal{L}_b$	$\mathcal{L}_g$	$\mathcal{L}_{seg}$	$\mathcal{L}_b$	$\mathcal{L}_g$	×0.25	×0.5	×0.75	×1.0
w/o	✓						71.82	75.97	76.92	77.90
	✓	✓					73.08	76.34	77.12	78.14
	✓		✓				72.49	76.47	77.82	78.35
	✓	✓	✓				73.63	76.92	77.77	78.26
w	✓			✓			71.94	75.86	76.64	77.55
	✓	✓		✓	✓		73.12	76.04	77.21	78.21
	✓		✓	✓		✓	72.94	76.16	77.41	78.37
	✓	✓	✓	✓	✓	✓	**74.40**	**77.37**	**77.87**	**78.43**

Table 4: Ablation of different knowledge distillation (KD) strategies with Slim-SFNet-ResNet50 on Cityscapes *val*. Bold numbers and *italic* numbers indicate the best and second best results.

KD	Teacher	Loss	mIoU (%)			
			×0.25	×0.5	×0.75	×1.0
w/o	-	$\mathcal{L}_{CE/BCE}(p^n, y)$	73.63	76.92	77.77	78.26
w	prev	$\mathcal{L}_{KD}(p^n, p^{n+1})$	**74.40**	**77.37**	*77.87*	*78.43*
	largest	$\mathcal{L}_{KD}(p^n, p^N)$	73.64	76.72	77.04	78.38
	mean	$\mathcal{L}_{KD}(p^n, \frac{1}{N-n}\sum_{j=n+1}^{N} p^j)$	73.24	76.25	77.53	77.85
	larger	$\frac{1}{N-n}\sum_{j=n+1}^{N} \mathcal{L}_{KD}(p^n, p^j)$	73.25	75.87	**78.02**	**78.61**

Boundary Supervision. As shown in Table 3, with boundary detection loss $\mathcal{L}_b$, the mIoUs on all widths are improved, especially for the smallest submodels, with 1.2% increase from 71.94% to 73.12%. For slimmable models trained without $\mathcal{L}_b$ but with the boundary guided segmentation loss $\mathcal{L}_g$, we use the binary boundary ground truth label as a mask to generate a masked probability map p_{ms}. The boundary guided segmentation loss with ground truth labels also helps on improving the mIoUs on all width. With all the losses together, we get the best performance on all the submodels.

To demonstrate the improvements on semantic borders, we illustrate the histogram of the error pixels in Figure 3. It shows the statistics of error pixels numbers and their Euclidean distances to the nearest boundaries on 500 Cityscapes *val* images. Overall, the improved pixels are mainly distributed on the semantic borders. The improvement number of pixels within the range of 5 pixels along the borders accounts for about 50% of the total. Some qualitative results on Cityscapes *val* are shown in Figure 4. With the boundary supervision, the predicted segmentation maps of each width model are more consistent, especially on the boundary regions. Segmentation results for some interior regions are also improved.

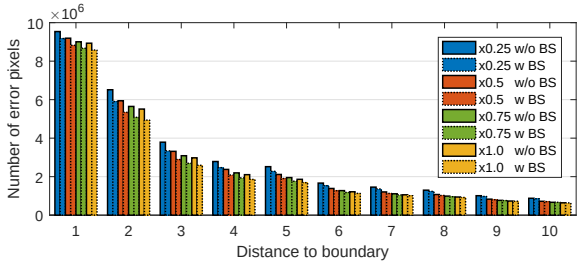
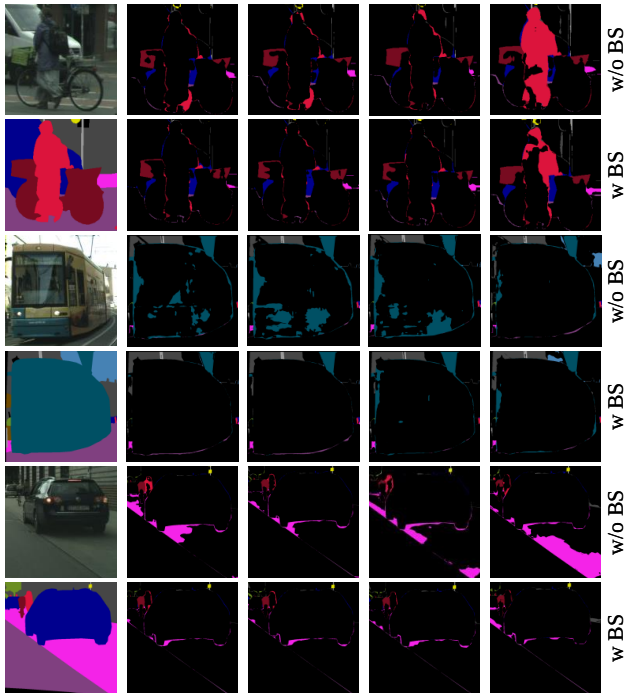


Figure 3: Comparison of the distribution of error pixels between slimmable models trained with and without boundary supervision (BS) on Cityscapes *val*. The model with boundary supervision has less error predictions on the boundary.



(a) Image/GT (b) ×1.0 (c) ×0.75 (d) ×0.5 (e) ×0.25

Figure 4: Visual comparison of our boundary supervision (BS) on Cityscapes *val*, 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. Submodels with boundary supervision perform better on small objects and semantic borders.

4.4 Comparisons with Real-time Models

We compare our method with other existing state-of-the-art real-time methods on Cityscapes and CamVid.

Results on Cityscapes. We present the mIoU and inference speed of our slimmable SFNet-ResNet50 and SFNet-ResNet18 (both

backbones are pretrained on ImageNet) and other real-time segmentation methods in Table 5. Our Slim-SFNet-ResNet50 achieves result (77.3%) with FPS 23.8. With ResNet18 as backbone, our method achieves 74.3% mIoU with 51.4 FPS.

Table 5: Comparison with state-of-the-art on Cityscapes *val*. ‡indicates the model is not pretrained on ImageNet.

Method	Resolution	Backbone	mIoU	FLOPs	FPS	Param
BiSeNetV1[54]	768×1536	Xception39	69.0	14.8	105.8	5.8
BiSeNetV1[54]	768×1536	ResNet18	74.8	55.3	65.5	49
CAS‡[60]	768×1536	Searched	71.6	-	108	-
GAS‡[32]	767×1537	Searched	72.4	-	163.9	-
DF1-Seg[30]	1024×2048	DFNetv1	74.1	-	106.4	-
DF2-Seg1[30]	1024×2048	DFNetv2	75.9	-	67.2	-
DF2-Seg2[30]	1024×2048	DFNetv2	76.9	-	56.3	-
SFNet[29]	1024×2048	ResNet18	78.7	247	18	12.9
BiSeNetV2‡[53]	1024×2048	None	73.4	21.3	-	-
BiSeNetV2-L‡[53]	512×1024	None	75.8	118.5	47.3	4.6
FasterSeg‡[3]	1024×2048	Searched	73.1	28.2	108.4	4.4
STDC2-Seg75[10]	768×1536	STDC2	77.0	54.9	97†	16.1
MSFNet[13]	1024×2048	ResNet18	77.2	96.8	41	-
CABiNet[52]	1024×2048	MBNetv3-s	76.6	12	76.5	2.64
CABiNet[52]	1024×2048	ResNet18	76.7	66.4	54.5	9.2
DDRNet-Seg[17]	1024×2048	DDRNet-23	79.5	143.1	37.1	20.1
Slim-SFNet ×[0.25, 0.5, 0.75, 1.0] (Ours)	1024×2048	ResNet50	74.4	39.4	46.2	2.0
			77.3	153.9	23.8	7.8
			77.8	343.4	13.2	17.6
			78.4	607.9	9.0	31.2
Slim-SFNet ×[0.25, 0.5, 0.75, 1.0] (Ours)	1024×2048	ResNet18	70.4	15.7	74.9	0.8
			74.3	61.5	51.4	3.2
			76.7	137.4	30.8	7.2
			77.9	243.6	21.8	12.9

Results on CamVid. Since the inference speed of different models is measured under different conditions, we list the corresponding GPU type. Table 6 shows the comparison results on CamVid between our method and SoTA methods. Our network achieves competitive trade-off between performance and speed by 80.1% (72.5% without ImageNet pretraining) mIoU with 55.7 FPS, which outperforms the original independently trained SFNet.

Discussion. Our work tackles the design of efficient and adjustable segmentation methods. In contrast to the SoTA real-time semantic segmentation methods, the performance of our methods do not rely on well-crafted compact network architectures. The experimental results demonstrated that our method can be directly applied to the mainstream segmentation frameworks and turn the fixed-computation models into adjustable ones. In this work, we use globally consistent width multipliers, but the optimal width of can be different for each layer, so we believe that the accuracy-efficiency tradeoff still has room for improvement. Furthermore, combining with image content, input resolution and depth of the network, the dynamic inference can be further explored.

5 CONCLUSION

In this paper, we propose a general slimmable semantic segmentation method, which enables adjustable accuracy-efficiency tradeoff through a width-swicthable segmentation network. We demonstrate the effectiveness of stepwise downward distillation on improving the performance of smaller subnetworks, and with less amount of features saved during training compared with other distillation strategies. Based on the observation of the difference between the predictions of each subnetwork, we introduce boundary supervision on low-level features of the network and propose a boundary guided loss to further improve the segmentation results of pixels along semantic borders. We demonstrate the effectiveness of the proposed method through extensive experiments with several different mainstream semantic segmentation networks on the Cityscapes and CamVid. Our proposed method improves the accuracy of the smaller submodels without great accuracy drops on large submodels.

Table 6: Comparison with state-of-the-art on CamVid *test* with image size 720×960. IM and CS represent using extra data, ImageNet and Cityscapes, for pretraining, respectively. †indicates the FPS is measured with TensorRT acceleration.

Method	Extra	Backbone	mIoU	FPS	GPU
BiSeNetV1[54]	IM	Xception39	65.6	175	GTX1080Ti
BiSeNetV1[54]	IM	ResNet18	68.7	116.3	GTX1080Ti
CAS[60]	None	Searched	71.2	169	TitanXp
GAS[32]	None	Searched	72.8	153.1	TitanXp
SFNet[29]	IM	ResNet18	73.8	36	GTX1080Ti
MSFNet[13]	None	None	75.4	91	GTX2080Ti
STDC1-Seg[10]	IM	STDC1	73.0	198†	GTX1080Ti
STDC2-Seg[10]	IM	STDC2	73.9	152†	GTX1080Ti
BiSeNetV2[53]	CS	None	76.7	124.5	GTX1080Ti
BiSeNetV2-L[53]	CS	None	78.5	32.7	GTX1080Ti
DDRNet-Seg[17]	CS	DDRNet-23	80.6	94	GTX2080Ti
Slim-SFNet ×[0.25, 0.5, 0.75, 1.0] (Ours)	CS	ResNet50	78.0	57.1	GTX2080Ti
			80.6	47.9	
			81.6	31.7	
			81.7	21.8	
Slim-SFNet ×[0.25, 0.5, 0.75, 1.0] (Ours)	IM	ResNet18	71.0	102.8	GTX2080Ti
			73.6	98	
			74.8	72.6	
			75.2	55.7	
Slim-SFNet ×[0.25, 0.5, 0.75, 1.0] (Ours)	CS	ResNet18	75.0	102.8	GTX2080Ti
			77.9	98	
			79.5	72.6	
			80.1	55.7	

ACKNOWLEDGMENTS

This work was partially supported by National Science Foundation of China under Grant No.U19B2037 and No.61901384, Natural Science Basic Research Program of Shaanxi Province (Program No.2021JCW-03), Grant PID2021-128178OB-I00 funded by MCIN/AEI/10.13039/501100011033, ERDF "A way of making Europe" and the Ramón y Cajal grant RYC2019-027020-I'. D.X. and P.W. thank the funding from China Scholarship Council (No.202006290209, No.201906290067).

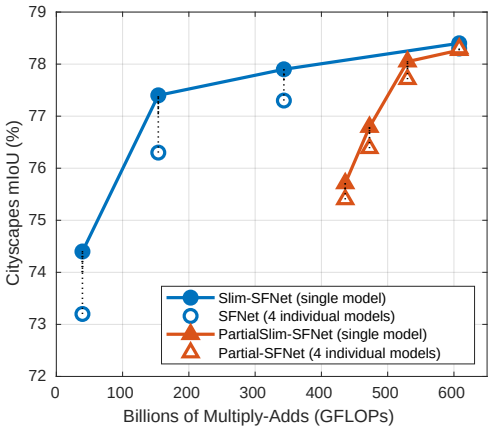
A SLIMMABLE SEGMENTATION FRAMEWORK

In this section, we present specific experimental results to illustrate why we choose to slim the entire segmentation framework instead of just a part of it. In addition to computational considerations, it is also because the use of more complex decoders cannot significantly improve the accuracy of submodels.

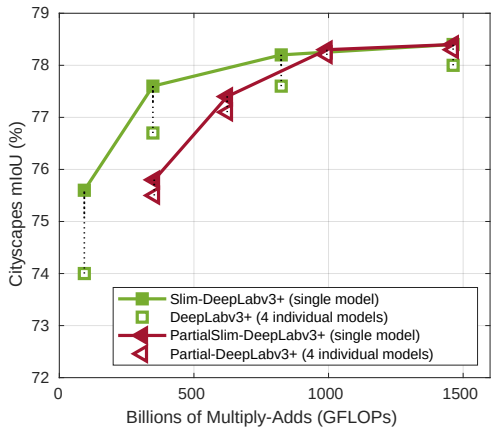
A.1 Globally Slimmable v.s. Partially Slimmable

Yu et al. [57] has applied their slimmable ResNet50 backbone on instance segmentation task, but except for the slimmable ResNet50, the other parts of the Mask-RCNN (i.e. the lateral layers and the decoder) are non-slimmable. While in our work, we set the entire network to be adjustable in width, even the for the Pyramid Pooling Module, the lateral layers and the decoder. We report the mIoU and FLOPs of the **globally slimmable** models, the **partially slimmable** models (with only the slimmable backbone), and their independent counterparts in Table 7. Two kinds of structures, including SFNet [29] and Deeplabv3+ [2], both with slimmable ResNet50 [57] pretrained on ImageNet as backbone, are tested. For the partially slimmable models, the number of channels in non-slimmable parts is fixed and the same as that of subnetwork with width ×1.0 in globally slimmable model. As illustrated in Figure 5, the computation reduction brought by seldom slimming the backbone is relatively small. For partially slimmable models, the mIoU gap between submodels of different width is smaller than the gap between globally slimmable submodels, and the range of FLOPs is narrower, due to the fixed non-slimmable parts in partially slimmable submodels. At the same time, compared with partially slimmable models, globally slimmable models have more obvious advantages on mIoU than corresponding independent models. Since the number of parameters of the non-backbone part in SFNet accounts for higher percentage than that in DeepLabv3 (shown in Table 1 of the main paper), the difference on mIoU and FLOPs between globally and partially slimmable models is also larger.

Overall, comparing the mIoU-FLOPs curves of the globally and partially slimmable model, the globally slimmable model can achieve higher mIoU than the partially slimmable model with the same amount of computation, especially for smaller submodels. Therefore, we believe that globally slimmable semantic segmentation network leads to a better accuracy-efficiency tradeoff.



(a) (Slim-)SFNet-ResNet50



(b) (Slim-)DeepLabv3plus-ResNet50

Figure 5: FLOPs-mIoU spectrum of globally and partially slimmable networks on Cityscapes *val*.

B BOUNDARY SUPERVISION

In this section, we conduct more experiments about the boundary supervision and show some visualization results of features and predicted segmentation maps.

B.1 Boundary Groundtruth

Boundary segmentation ground truth labels are generated based on the semantic segmentation ground truth labels, since we only focus on the boundaries between different semantic categories rather than the obvious image edges inside the regions with the same semantic category. In the experiments presented in the main paper, we set the radius of the boundary region to 3 pixels. Here we compare the mIoUs of the Slim-SFNet-ResNet50 models using different boundary ground truth labels in Table 8. When radius equals to 3 pixels, the mIoU of each subnetwork is the optimal. Smaller boundary regions benefit to exploit hard samples, but when the number of boundary samples is too small, it is not conducive to the network to fully learn the characteristics of boundary samples.

B.2 Input of the Boundary Detection Head

The input of our boundary detection head is the low-level features extracted by the first few layers of the backbone networks. We report the mIoUs of models trained with different low-level features as boundary detection input. As shown in Table 10, 'conv1', 'conv2_x', 'conv3_x' represent the first three stages of ResNet50 [15], where 'x' indicates the numbers of the residual blocks. According to Table 9, slimmable models trained with boundary supervision, including the boundary detection head and the loss functions $\mathcal{L}_b$ and $\mathcal{L}_g$, outperform the slimmable model without boundary supervision. Using the features output by layer 'conv2_3' of ResNet50 lead to higher overall mIoU.

The layer 'conv1' contains only one convolutional layer. Although the extracted features can identify image edges, what we need is boundaries between different semantic categories, which

Table 7: Comparison between globally slimmable models and partially slimmable models on Cityscapes *val*. *Note that both the independent and slimmable models are trained with the sum of the three losses in Equation (5) (in the main paper) for fair comparison.

Network	Slimmable Part	Width	Independent*		Slimmable*		GFLOPs
			mIoU (%)	Param (M)	mIoU (%)	Param (M)	
SFNet ResNet50	Backbone (Partially slimmable)	×1.0	78.3	31.2	**78.3** (0.0↑)	31.3	608.0
		×0.75	77.8	20.2	**78.0** (0.2↑)		529.9
		×0.5	76.4	12.1	**76.8** (0.4↑)		472.6
		×0.25	75.4	6.9	**75.7** (0.3↑)		436.0
	Backbone+PPM+Decoder (Globally slimmable)	×1.0	78.3	31.3	**78.4** (0.1↑)	31.3	607.9
		×0.75	77.3	17.6	**77.9** (0.6↑)		343.4
		×0.5	76.3	7.8	**77.4** (1.1↑)		153.9
		×0.25	73.2	2.0	**74.4** (1.2↑)		39.4
DeepLabv3+ ResNet50	Backbone (Partially slimmable)	×1.0	78.3	40.4	**78.4** (0.1↑)	40.4	1462.8
		×0.75	78.2	26.3	**78.3** (0.1↑)		993.7
		×0.5	77.1	15.1	**77.4** (0.3↑)		623.7
		×0.25	75.5	6.9	**75.8** (0.3↑)		352.7
	Backbone+PPM+Decoder (Globally slimmable)	×1.0	78.0	40.4	**78.4** (0.4↑)	40.4	1462.8
		×0.75	77.6	22.7	**78.2** (0.6↑)		824.3
		×0.5	76.7	10.1	**77.6** (0.9↑)		347.6
		×0.25	74.0	2.5	**75.6** (1.6↑)		92.9

Table 8: mIoUs of slimmable models trained with different boundary detection groundtruth.

Radius (pixels)	mIoU (%)			
	×0.25	×0.5	×0.75	×1.0
1	*74.10*	*76.63*	*77.59*	*77.91*
3	**74.40**	**77.37**	**77.86**	**78.43**
5	73.63	76.02	76.93	77.38

Table 9: mIoUs slimmable models trained with different low-level features as the input of boundary detection head.

Boundary Head	Input Features	mIoU (%)				
		×0.25	×0.5	×0.75	×1.0	Ave
w/o	-	71.94	75.86	76.64	77.55	75.50
w	conv1	73.56	*76.99*	77.54	78.18	76.57
	conv2_3	**74.40**	**77.37**	**77.86**	**78.43**	**77.02**
	conv3_4	*74.12*	76.72	*77.57*	*78.32*	76.68

contains semantic information to a certain extent. Moreover, as our main task is to perform semantic segmentation, in addition to exploiting the boundary pixels, the context information of the object itself is more important. If edge constraint is added to the feature output by the layer 'conv1', it will have a greater impact on subsequent features, so we add the boundary supervision on the deeper features. The features output by the layer 'conv3_4' have been processed by three stages of convolutions, and contain some semantic information, so the overall mIoU outperforms the model using 'conv1'. However, since the resolution of the features output by the layer 'conv3_4' are down-sampled by 8 times compared to the original image, boundaries and details have been lost, so using the output features of the layer 'conv3_4' as input for boundary detection is worse than layer 'conv2_3'.

Table 10: Architectures of ResNet50 [15]. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2. The input image size is 3×1024×2048.

Layer Name	Output Size (C×H×W)	ResNet50
conv1	64×512×1024	7×7, stride 2
conv2_x	256×256×512	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	512×128×256	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4_x	1024×64×128	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5_x	2048×32×64	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$

B.3 More Visualization Results

To show the effectiveness of the boundary supervision, we present more visualization results of both features and segmentation results on Cityscapes *val* [5]. As shown in Figure 6, the features in the regions near the boundary and the textured details of the objects, such as the head of the truck, are enhanced with boundary supervision, so the corresponding segmentation results in these areas have fewer errors. In Figure 7, we compare the error maps of segmentation predictions between slimmable models with and without boundary supervision. As we can see, not only the semantic boundaries are improved with boundary supervision, but also the thin small objects, such as the pole, fence, traffic sign, have better results especially for small submodels. The gaps between the predictions of submodels with different width are narrowed with boundary supervision.

REFERENCES

[1] Gabriel J Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. 2008. Segmentation and recognition using structure from motion point clouds. In *European conference on computer vision*. Springer, 44–57.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*. 801–818.

[3] Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. 2020. FasterSeg: Searching for Faster Real-time Semantic Segmentation. In *International Conference on Learning Representations*.

[4] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. 2019. You look twice: Gaternet for dynamic filter selection in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9172–9180.

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3213–3223.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[7] Henghui Ding, Xudong Jiang, Ai Qun Liu, Nadia Magnenat Thalmann, and Gang Wang. 2019. Boundary-aware feature propagation for scene segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6819–6829.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[9] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88, 2 (2010), 303–338.

[10] Mingyuan Fan, Shenqi Lai, Junshi Huang, Xiaoming Wei, Zhenhua Chai, Junfeng Luo, and Xiaolin Wei. 2021. Rethinking bisenet for real-time semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9716–9725.

[11] Roland Gao. 2021. Rethink Dilated Convolution for Real-time Semantic Segmentation. *arXiv preprint arXiv:2111.09957* (2021).

[12] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. 2018. Dynamic Channel Pruning: Feature Boosting and Suppression. In *International Conference on Learning Representations*.

[13] Feng Lu Haiyang Si, Zhiqiang Zhang. 2020. Real-Time Semantic Segmentation via Multiply Spatial Fusion Network. In *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press.

[14] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. 2021. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[16] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* 2, 7 (2015).

[17] Yuanduo Hong, Huihui Pan, Weichao Sun, and Yisong Jia. 2021. Deep Dual-resolution Networks for Real-time and Accurate Semantic Segmentation of Road Scenes. *arXiv preprint arXiv:2101.06085* (2021).

[18] Liang Hou, Zehuan Yuan, Lei Huang, Huawei Shen, Xueqi Cheng, and Changhu Wang. 2020. Slimmable generative adversarial networks. *arXiv preprint arXiv:2012.05660* (2020).

[19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1314–1324.

[20] Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G Edward Suh. 2019. Channel gating neural networks. *Advances in Neural Information Processing Systems* 32 (2019).

[21] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. 2018. Multi-Scale Dense Networks for Resource Efficient Image Classification. In *International Conference on Learning Representations*.

[22] Zutao Jiang, Changlin Li, Xiaojun Chang, Jihua Zhu, and Yi Yang. 2021. Dynamic Slimmable Denoising Network. *arXiv preprint arXiv:2110.08940* (2021).

[23] Alexandros Kouris, Stylianos I Venieris, Stefanos Laskaridis, and Nicholas D Lane. 2021. Multi-Exit Semantic Segmentation Networks. *arXiv preprint arXiv:2106.03527* (2021).

[24] Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. 2021. Dynamic slimmable network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8607–8617.

[25] Daiqing Li, Junlin Yang, Karsten Kreis, Antonio Torralba, and Sanja Fidler. 2021. Semantic segmentation with generative models: Semi-supervised learning and strong out-of-domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8300–8311.

[26] G Li and J Kim. 2020. DABNet: Depth-wise asymmetric bottleneck for real-time semantic segmentation. In *30th British Machine Vision Conference 2019, BMVC 2019*. BMVA Press.

[27] Hanchao Li, Pengfei Xiong, Haoqiang Fan, and Jian Sun. 2019. Dfanet: Deep feature aggregation for real-time semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9522–9531.

[28] Peike Li, Xuanyi Dong, Xin Yu, and Yi Yang. 2020. When Humans Meet Machines: Towards Efficient Segmentation Networks.. In *BMVC*.

[29] Xiangtai Li, Ansheng You, Zhen Zhu, Houlong Zhao, Maoke Yang, Kuiyuan Yang, Shaohua Tan, and Yunhai Tong. 2020. Semantic flow for fast and accurate scene parsing. In *European Conference on Computer Vision*. Springer, 775–793.

[30] Xin Li, Yiming Zhou, Zheng Pan, and Jiashi Feng. 2019. Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9145–9153.

[31] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. 2020. Learning dynamic routing for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8553–8562.

[32] Peiwen Lin, Peng Sun, Guangliang Cheng, Sirui Xie, Xi Li, and Jianping Shi. 2020. Graph-guided architecture search for real-time semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4203–4212.

[33] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2117–2125.

[34] Yifan Liu, Changyong Shu, Jingdong Wang, and Chunhua Shen. 2020. Structured knowledge distillation for dense prediction. *IEEE transactions on pattern analysis and machine intelligence* (2020).

[35] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3431–3440.

[36] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5191–5198.

[37] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. GauGAN: semantic image synthesis with spatially adaptive normalization. In *ACM SIGGRAPH 2019 Real-Time Live!* 1–1.

[38] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. 2016. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147* (2016).

[39] Dian Qin, Jia-Jun Bu, Zhe Liu, Xin Shen, Sheng Zhou, Jing-Jun Gu, Zhi-Hua Wang, Lei Wu, and Hui-Fen Dai. 2021. Efficient medical image segmentation based on knowledge distillation. *IEEE Transactions on Medical Imaging* 40, 12 (2021), 3820–3831.

[40] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. 2017. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems* 19, 1 (2017), 263–272.

[41] Changyong Shu, Yifan Liu, Jianfei Gao, Yan Zheng, and Chunhua Shen. 2021. Channel-wise Knowledge Distillation for Dense Prediction. In *ICCV*.

[42] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[43] Ke Sun, Yang Zhao, Borui Jiang, Tianheng Cheng, Bin Xiao, Dong Liu, Yadong Mu, Xinggang Wang, Wenyu Liu, and Jingdong Wang. 2019. High-resolution representations for labeling pixels and regions. *arXiv preprint arXiv:1904.04514* (2019).

[44] Michael Treml, José Arjona-Medina, Thomas Unterthiner, Rupesh Durgesh, Felix Friedmann, Peter Schuberth, Andreas Mayr, Martin Heusel, Markus Hofmarcher, Michael Widrich, et al. 2016. Speeding up semantic segmentation for autonomous driving. (2016).

[45] Andreas Veit and Serge Belongie. 2018. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 3–18.

[46] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. 2020. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence* 43, 10 (2020), 3349–3364.

[47] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8798–8807.

[48] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. 2018. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 409–424.

[49] Yikai Wang, Fuchun Sun, Duo Li, and Anbang Yao. 2020. Resolution switchable networks for runtime efficient image recognition. In *European Conference on Computer Vision*. Springer, 533–549.

[50] Fei Yang, Luis Herranz, Yongmei Cheng, and Mikhail G Mozerov. 2021. Slimmable compressive autoencoders for practical neural image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4998–5007.

[51] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. 2020. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2369–2378.

[52] Michael Ying Yang, Saumya Kumaar, Ye Lyu, and Francesco Nex. 2021. Real-time semantic segmentation with context aggregation network. *ISPRS Journal of Photogrammetry and Remote Sensing* 178 (2021), 124–134.

[53] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. 2021. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *International Journal of Computer Vision* 129, 11 (2021), 3051–3068.

[54] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. 2018. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*. 325–341.

[55] Jiahui Yu and Thomas Huang. 2019. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728* (2019).

[56] Jiahui Yu and Thomas S Huang. 2019. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1803–1811.

[57] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. 2018. Slimmable neural networks. *arXiv preprint arXiv:1812.08928* (2018).

[58] Yuhui Yuan, Jingyi Xie, Xilin Chen, and Jingdong Wang. 2020. Segfix: Model-agnostic boundary refinement for segmentation. In *European Conference on Computer Vision*. Springer, 489–506.

[59] Xiong Zhang, Hongmin Xu, Hong Mo, Jianchao Tan, Cheng Yang, Lei Wang, and Wenqi Ren. 2021. Dcnas: Densely connected neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13956–13967.

[60] Yiheng Zhang, Zhaofan Qiu, Jingen Liu, Ting Yao, Dong Liu, and Tao Mei. 2019. Customizable architecture search for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11641–11650.

[61] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. 2018. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European conference on computer vision (ECCV)*. 405–420.

[62] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2881–2890.

[63] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. 2021. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 6881–6890.

[64] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. 2017. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 633–641.

[65] Mingjian Zhu, Kai Han, Enhua Wu, Qiulin Zhang, Ying Nie, Zhenzhong Lan, and Yunhe Wang. 2021. Dynamic Resolution Network. *Advances in Neural Information Processing Systems* 34 (2021).

[66] Yi Zhu, Karan Sapra, Fitsum A Reda, Kevin J Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. 2019. Improving semantic segmentation via video propagation and label relaxation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8856–8865.

[67] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).

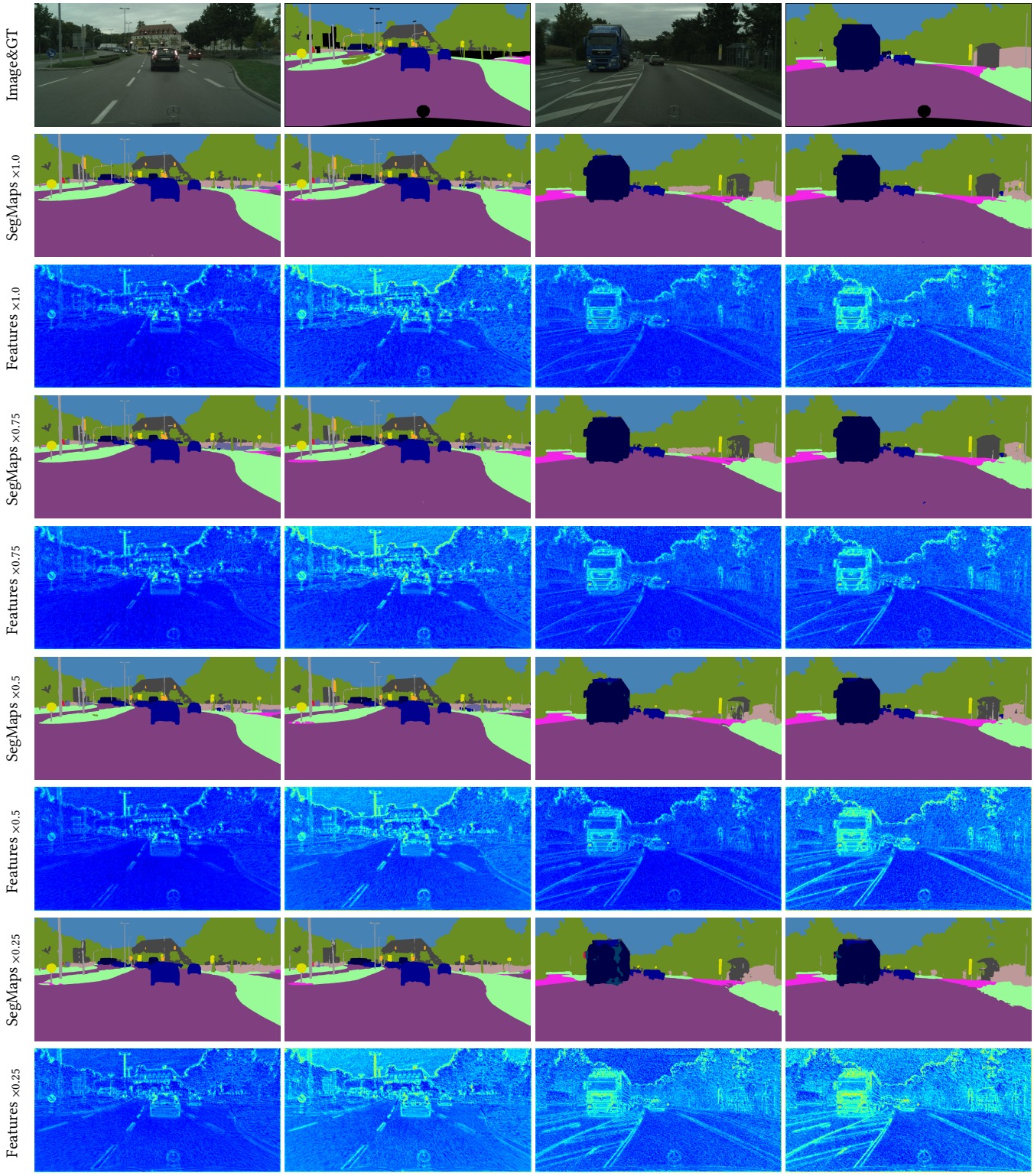


Figure 6: Visual comparison of our boundary supervision on Cityscapes *val*, in terms of the average feature maps of the output of layer 'conv2_3' in ResNet50. Column 1 and 3 are the colored semantic segmentation maps and average features predicted by slimmable submodels without boundary supervision. The brighter color indicates the larger number of features. Column 2 and 4 are the results with boundary supervision. With boundary supervision, the features in boundary and textured regions are enhanced, which results in better segmentation results of these area.

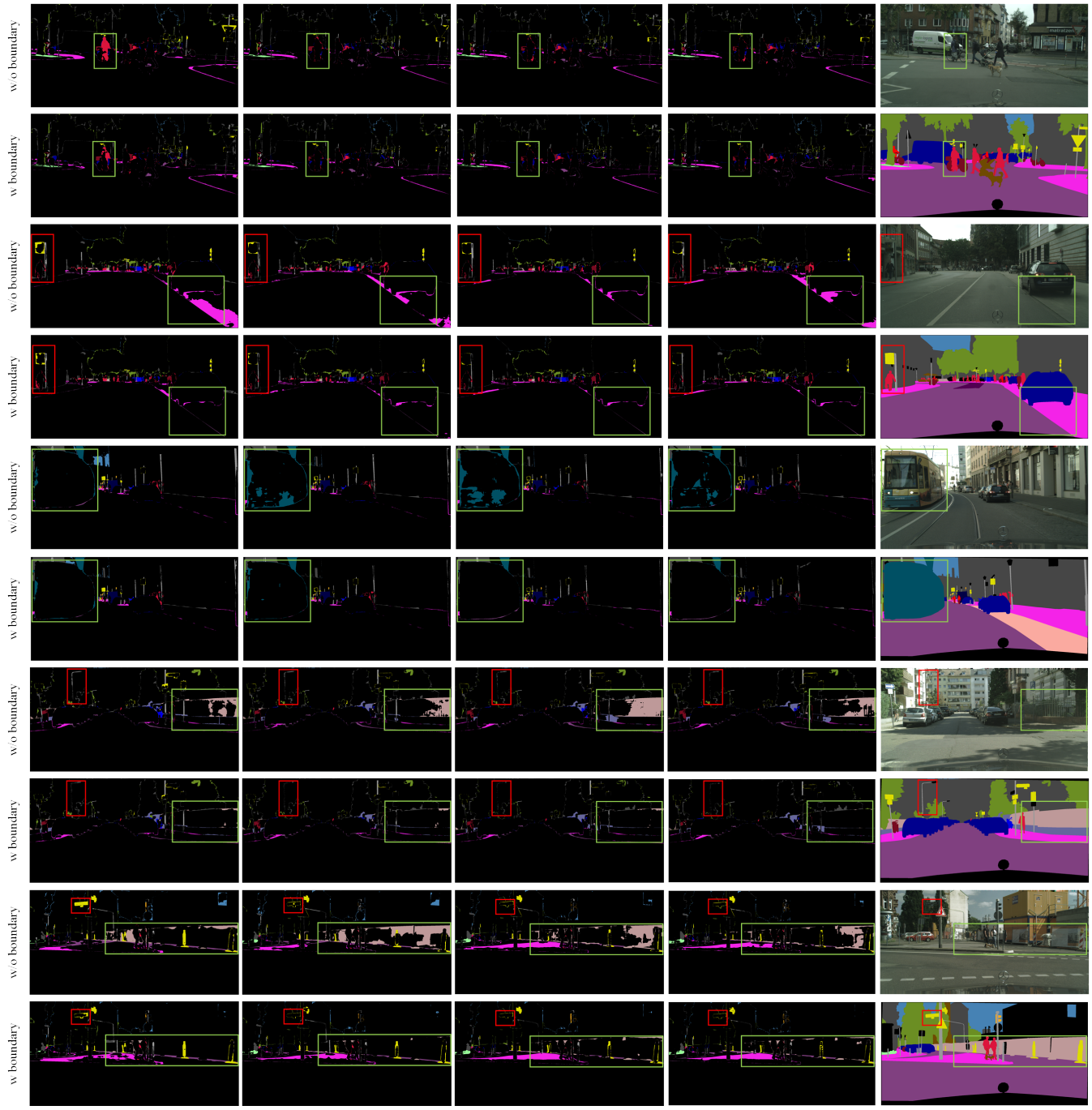


Figure 7: Visual comparison of our boundary supervision on Cityscapes *val*, in terms of errors in predictions, where correctly predicted pixels are shown as black background while wrongly predicted pixels are colored with their ground truth labels color codes. Models with boundary supervision performs better on small objects, such as poles and traffic signs, and semantic borders. Please zoom in for better viewing.