

BCSNet: Boundary-Centric Fusion and Selective Semantic Pyramid Injection Network for Real-Time Semantic Segmentation

Min Li^{1,2}, QingPei Liu^{1,2}, Yuan Gao^{1,2}, Mingle Zhou^{1,2}, Yang Chen^{3,1,2}, and Delong Han^{1,2} 

¹ Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan, China

² Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing, Shandong Fundamental Research Center for Computer Science, Jinan, China

³ School of Computer Science and Engineering, Southeast University, Nanjing, China
handl@sdas.org

Abstract. Real-time semantic segmentation requires accurate dense prediction under strict latency and computational constraints. Although dual-branch architectures provide an effective accuracy–efficiency trade-off, they remain susceptible to boundary degradation during cross-branch fusion, insufficient scale diversity for small and medium objects, and feature misalignment introduced by content-agnostic upsampling. These limitations are particularly evident around class transitions, thin structures, and distant objects. To address these issues, we propose the Boundary-Centric Fusion and Selective Semantic Pyramid Injection Network (BCSNet). Its central design is a shared boundary representation that coordinates three boundary-sensitive operations within a lightweight two-branch architecture. BCFR uses this representation to regulate bidirectional cross-branch interaction and local logit refinement. SL-FPN-SI performs per-pixel semantic-scale selection through a compact feature pyramid. BCRAU restricts content-adaptive reassembly to boundary-related regions while retaining bilinear interpolation elsewhere. Under a controlled RTX 4090 evaluation protocol, BCSNet-L achieves 79.5% mIoU at 107.0 FPS on Cityscapes, while BCSNet-S obtains 76.5% mIoU at 189.8 FPS. On CamVid, BCSNet reaches 77.1% mIoU at 156.8 FPS. The results indicate that BCSNet provides a competitive accuracy–efficiency trade-off under the evaluated desktop-GPU setting. Embedded deployment and hardware-specific operator optimization remain subjects for further investigation.

Keywords: Real-time semantic segmentation, Boundary-aware learning, Multi-scale representation, Adaptive upsampling

1 Introduction

Semantic segmentation assigns a semantic label to each image pixel and is a fundamental component of dense scene understanding in autonomous driving, robotic navigation, and related perception tasks. In latency-sensitive applications, segmentation models must identify drivable areas, vehicles, pedestrians, traffic signs, and other safety-relevant objects while satisfying limited computation and memory budgets. Despite recent progress, real-time semantic segmentation remains challenging because high-resolution prediction requires both fine spatial detail and sufficiently strong semantic context. Improving one of these properties often increases computational cost or weakens the other, resulting in a persistent trade-off among regional accuracy, boundary quality, and inference efficiency.

To improve this trade-off, recent methods have adopted efficiency-oriented backbones, asymmetric resolution processing, and lightweight feature interaction. A representative solution is the dual-branch paradigm, in which a high-resolution branch preserves spatial information and a low-resolution branch captures semantic context. BiSeNetV2 [24], DDRNet [5], and GCNet [23], among others, illustrate the effectiveness of this design. Nevertheless, prediction errors remain concentrated around object contours, thin structures, and small or distant instances. Accurate recognition in these regions depends not only on high-resolution features, but also on cross-scale alignment, semantic scale diversity, and controlled interaction between detail and semantic representations.

We identify three related limitations in existing real-time segmentation pipelines. First, cross-branch fusion is commonly implemented by bilinear upsampling followed by convolutional mixing, which may attenuate high-frequency details and weaken boundary responses. Second, the semantic branch often provides limited scale diversity for small and medium objects, whereas introducing heavy pyramid decoders increases latency and memory cost. Third, interpolation-based upsampling is efficient but content-agnostic, and thus may introduce feature misalignment near class transitions. Although dense adaptive upsampling can improve local alignment, applying it to the entire feature map is usually unfavorable for real-time inference. These limitations are not isolated; they are most harmful in boundary-critical regions, where fusion, semantic injection, and cross-scale alignment jointly determine the final prediction quality. These findings suggest a boundary-centric design principle: modeling capacity should be adaptively allocated to error-prone boundary regions instead of being uniformly expanded over the entire spatial domain.

Based on this principle, we propose **BCSNet** (Boundary-Centric Fusion and Selective Semantic Pyramid Injection Network), an efficient two-branch framework in which a shared boundary representation coordinates three boundary-sensitive operations. **BCFR** regulates asymmetric bidirectional interaction between semantic and detail features and performs local logit refinement. **SL-FPN-SI** constructs a compact semantic pyramid and predicts location-dependent scale weights for selective semantic injection. **BCRAU** combines bilinear interpolation with regional content-adaptive reassembly, where the adaptive path is evaluated only in boundary-related regions. These

components are designed as coordinated uses of a common boundary signal rather than independent feature-enhancement modules.

The main contributions of this work are summarized as follows:

- We propose BCSNet, a boundary-centric real-time semantic segmentation framework that selectively allocates additional modeling capacity to boundary-critical regions, providing a focused design for improving boundary quality under a constrained efficiency budget.
- We introduce BCFR, a shared-boundary-guided fusion and refinement module that uses boundary cues to regulate bidirectional cross-branch interaction and enhance local contour preservation with limited computational overhead.
- We design SL-FPN-SI, a lightweight semantic pyramid with selective injection, to introduce scale-adaptive semantic cues into the high-resolution stream, improving the representation of small and medium objects without relying on a heavy decoder.
- We propose BCRAU, a boundary-conditioned region-adaptive upsampling operator that restricts content-aware reassembly to boundary regions for efficient cross-scale alignment. Experiments on Cityscapes, CamVid, and COCO-Stuff10K, together with controlled ablation studies, indicate that BCSNet achieves a practical accuracy–efficiency trade-off.

2 Related Work

2.1 Real-Time Semantic Segmentation

Real-time semantic segmentation seeks a balance between dense prediction accuracy and deployment efficiency [11]. Representative CNN-based methods such as BiSeNetV2 [24] and DDRNet [5] improve this trade-off through dual-branch feature decomposition and repeated cross-resolution fusion, while PIDNet [21] introduces explicit boundary-aware modeling to alleviate detail oversmoothing. More recent methods explore stronger context modeling under real-time constraints, including RTFormer [17], SeaFormer [16], AFFormer [3], SCTNet [22], and GCNet [23]. In contrast to methods that rely on an additional heavy semantic pathway or an explicit auxiliary branch, BCSNet employs a shared boundary representation to guide cross-branch interaction and boundary-sensitive refinement within a lightweight two-branch framework [24].

2.2 Multi-Scale Context and Boundary Modeling

Multi-scale semantic representation and boundary preservation remain two central challenges in urban-scene parsing, especially for thin structures and small objects. Recent efficient architectures enhance contextual representation through token pyramids, mobile-friendly attention, or training-time semantic guidance, as exemplified by TopFormer [27], SeaFormer [16], AFFormer [3], and SCTNet [22]. Meanwhile, boundary-aware designs such as PIDNet [21] show that explicitly modeling contour

information is beneficial for detail recovery. However, existing methods typically treat semantic enrichment and boundary correction as loosely coupled processes [20]. BCSNet addresses them jointly by combining selective semantic pyramid injection with shared boundary guidance [15], thereby improving contextual diversity and contour precision without introducing a heavy decoding stage [13].

2.3 Adaptive Upsampling and Feature Alignment

Accurate cross-scale fusion further depends on feature alignment during upsampling. FPN [6] establishes a standard top-down multi-scale aggregation scheme, but interpolation-based alignment may still blur object boundaries. CARAFE [18] demonstrates that content-aware reassembly can improve feature upsampling, although applying such adaptive operators densely over the full image is often unfavorable for real-time deployment. Our boundary-constrained adaptive upsampling follows the same content-aware motivation, but restricts adaptive reassembly to narrow boundary regions while retaining bilinear interpolation elsewhere, improving boundary alignment with a lower computational cost [8].

3 Methodology

3.1 Overall Network Architecture

Given an input image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, BCSNet adopts a two-branch architecture. The semantic branch progressively reduces the spatial resolution to output strides 16, 32, and 64 to capture contextual information. The detail branch maintains an output stride of $r = 8$ to preserve spatial structures and class transitions. As shown in Fig. 1, the branches exchange information at multiple stages. Semantic features are aligned and injected into the detail stream, while selected detail features are fed back to the semantic stream. After the final semantic stage, a lightweight context aggregation module produces a compact semantic representation. This representation is fused with the detail stream to obtain $\mathbf{F}_{hi} \in \mathbb{R}^{\frac{H}{r} \times \frac{W}{r} \times C}$. A lightweight segmentation head predicts initial logits $\mathbf{Z} \in \mathbb{R}^{\frac{H}{r} \times \frac{W}{r} \times K}$ for K classes.

Throughout the paper, r denotes the output stride of the high-resolution detail branch, whereas k denotes a semantic-branch stride. $\mathcal{U}_{k \rightarrow r}$ and $\mathcal{D}_{r \rightarrow k}$ denote spatial upsampling and downsampling alignment, respectively. $\mathbf{B}$ is the soft boundary probability, $\mathbf{G}$ is the continuous routing gate, $\mathbf{M}$ is the thresholded and dilated boundary-band mask, and $\mathbf{M}_{align}$ is the corresponding mask resized to the target alignment resolution. The operator $\odot$ denotes element-wise multiplication with spatial broadcasting over channels.

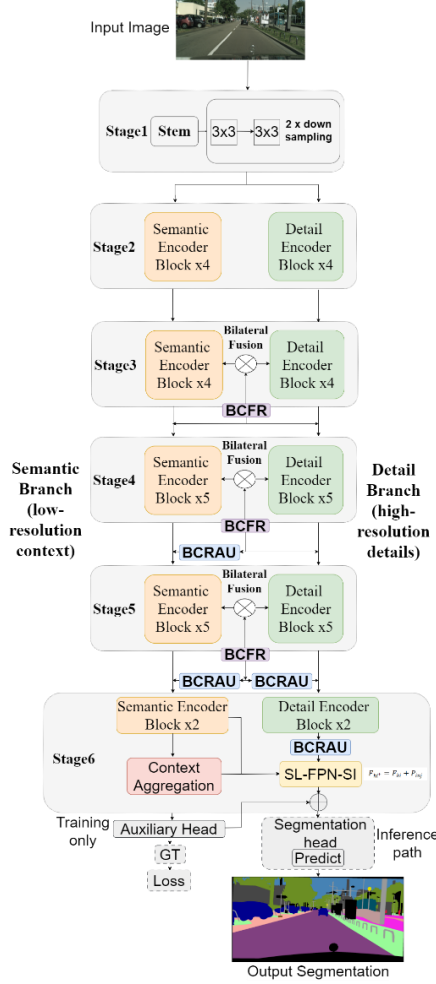


Fig. 1. Overall architecture of the proposed BCSNet. The network follows an efficient two-branch design and integrates BCFR, SL-FPN-SI, and BCRAU for boundary-aware fusion, multi-scale semantic enhancement, and boundary-sensitive feature alignment.

3.2 Boundary-Centric Cross-Branch Fusion and Refinement

As illustrated in Fig. 2, BCFR derives a shared boundary representation from the high-resolution detail branch and uses it for cross-branch feature routing and local logit refinement. The module consists of boundary estimation, asymmetric bidirectional routing, and boundary-band refinement.

Boundary representation. Following the structure shown in Fig. 2, we first predict a boundary probability map from the high-resolution detail feature $\mathbf{F}_{\text{det}}$:

$$\mathbf{B} = \sigma(g_{\text{bnd}}(\mathbf{F}_{\text{det}})), \mathbf{B} \in [0,1]^{\frac{H}{r} \times \frac{W}{r}} \quad (1)$$

where $g_{\text{bnd}}(\cdot)$ is a lightweight boundary head implemented by a depthwise separable convolution. For numerical stability, the boundary probability map is clipped as

$$\mathbf{G} = \text{clip}(\mathbf{B}, 10^{-6}, 1 - 10^{-6}) \quad (2)$$

where $\mathbf{G}$ is used as the boundary gate.

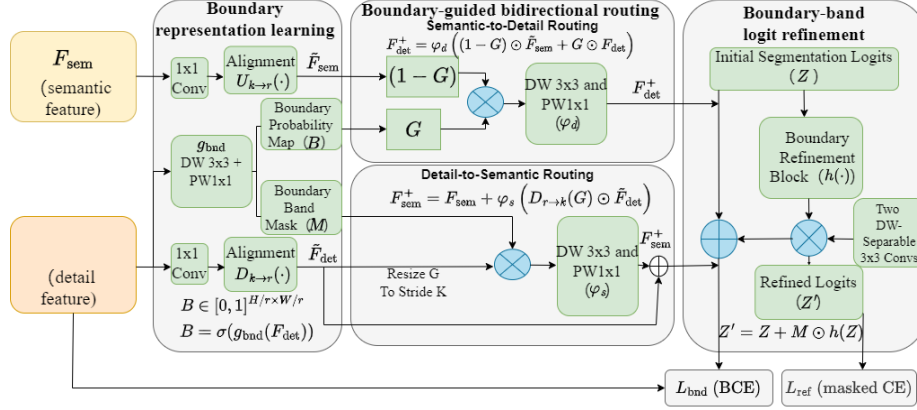


Fig. 2. Illustration of the proposed BCFR module. BCFR uses a shared boundary representation to guide bidirectional cross-branch interaction and boundary-focused refinement.

Boundary-guided bidirectional fusion. Let $\mathbf{F}_{\text{sem}}$ and $\mathbf{F}_{\text{det}}$ denote semantic and detail features at a fusion stage, with output strides k and r , respectively. We first align their spatial resolutions and channel dimensions:

$$\tilde{\mathbf{F}}_{\text{sem}} = \mathcal{U}_{k \rightarrow r}(\mathcal{C}(\mathbf{F}_{\text{sem}})), \tilde{\mathbf{F}}_{\text{det}} = \mathcal{D}_{r \rightarrow k}(\mathcal{E}(\mathbf{F}_{\text{det}})) \quad (3)$$

where $\mathcal{C}(\cdot)$ and $\mathcal{E}(\cdot)$ are 1×1 convolutions, $\mathcal{D}_{r \rightarrow k}(\cdot)$ denotes downsampling alignment, and $\mathcal{U}_{k \rightarrow r}(\cdot)$ denotes upsampling alignment.

For semantic-to-detail injection, the boundary gate modulates feature fusion:

$$\mathbf{F}_{\text{det}}^+ = \phi_d((1 - \mathbf{G}) \odot \tilde{\mathbf{F}}_{\text{sem}} + \mathbf{G} \odot \mathbf{F}_{\text{det}}) \quad (4)$$

where $\phi_d(\cdot)$ is a 3×3 depthwise separable convolution. For detail-to-semantic feedback, boundary-localized detail cues are injected into the semantic branch:

$$\mathbf{F}_{\text{sem}}^+ = \mathbf{F}_{\text{sem}} + \phi_s(\mathcal{D}_{r \rightarrow k}(\mathbf{G}) \odot \tilde{\mathbf{F}}_{\text{det}}) \quad (5)$$

where $\phi_s(\cdot)$ is also implemented by a 3×3 depthwise separable convolution. The two routing directions serve different purposes. In semantic-to-detail routing, the continuous gate retains high-resolution responses around predicted boundaries and introduces semantic information mainly in spatially homogeneous regions. In detail-to-semantic routing, only boundary-weighted detail features are propagated, which limits the transfer of unrelated low-level responses into the semantic stream.

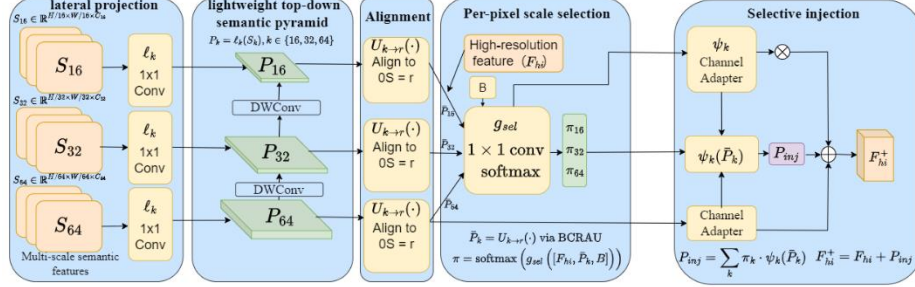


Fig. 3. Illustration of the proposed SL-FPN-SI module. It constructs a lightweight semantic pyramid and selectively injects scale-adaptive semantic cues into the high-resolution stream.

Boundary-aware refinement and loss. A boundary band mask $\mathbf{M}$ is obtained from $\mathbf{B}$:

$$\mathbf{M} = \delta(\mathbf{B}; \tau) \quad (6)$$

where $\delta(\cdot)$ denotes thresholding at $\tau = 0.5$ followed by 2-pixel dilation with a 3×3 square structuring element. The initial logits are refined only inside the boundary band:

$$\mathbf{Z}' = \mathbf{Z} + \mathbf{M} \odot h(\mathbf{Z}) \quad (7)$$

where $h(\cdot)$ is a lightweight residual refinement block with two DW-separable 3×3 convolutions.

The overall objective is

$$\mathcal{L} = \mathcal{L}_{\text{seg}} + \alpha \mathcal{L}_{\text{aux}} + \beta \mathcal{L}_{\text{bnd}} + \gamma \mathcal{L}_{\text{ref}} \quad (8)$$

Here, $\mathcal{L}_{\text{seg}}$ is the main segmentation loss, $\mathcal{L}_{\text{aux}}$ is the auxiliary segmentation loss, and $\mathcal{L}_{\text{bnd}}$ supervises the boundary map using ground-truth boundaries. The refinement loss is computed only within the boundary band:

$$\mathcal{L}_{\text{ref}} = \frac{1}{\sum_p \mathbf{M}(p) + \epsilon} \sum_p \mathbf{M}(p) \cdot \mathcal{L}_{\text{CE}}(\mathbf{Z}'(p), \mathbf{Y}_{\text{gt}}(p)) \quad (9)$$

where $\mathbf{Y}_{\text{gt}}$ is downsampled to stride r by nearest-neighbor interpolation, and $\epsilon = 10^{-6}$. We set $\alpha = 0.4$, $\beta = 0.2$, and $\gamma = 0.1$ in all experiments.

3.3 Semantic Lightweight Feature Pyramid with Selective Injection

As shown in Fig. 3, SL-FPN-SI introduces multi-scale semantic information without constructing a heavy high-resolution decoder. It first forms a compact pyramid from semantic-branch features and then predicts location-dependent scale weights for injection into the high-resolution stream.

Lightweight semantic pyramid. Given semantic features $\{\mathbf{S}_k\}_{k \in \mathcal{K}}$ with output strides $\mathcal{K} = \{16, 32, 64\}$, where $\mathbf{S}_k \in \mathbb{R}^{\frac{H}{k} \times \frac{W}{k} \times C_k}$, lateral embeddings are obtained by

$$\mathbf{P}_k = \ell_k(\mathbf{S}_k), k \in \mathcal{K} \quad (10)$$

where all pyramid features share $C_{\text{pyramid}} = 64$ channels. Coarse-to-fine fusion is then performed as

$$\mathbf{P}_{k_{\text{fine}}} \leftarrow \rho(\mathbf{P}_{k_{\text{fine}}} + \mathcal{U}_{\text{bil}}(\mathbf{P}_{k_{\text{coarse}}})) \quad (11)$$

where $k_{\text{fine}} < k_{\text{coarse}}$, and $\rho(\cdot)$ is a 3×3 depthwise separable convolution.

Selective injection. Each pyramid feature is aligned to stride $r = 8$:

$$\bar{\mathbf{P}}_k = \mathcal{U}_{k \rightarrow r}(\mathbf{P}_k), k \in \mathcal{K} \quad (12)$$

Per-pixel scale weights are computed as

$$\boldsymbol{\pi} = \text{softmax}(g_{\text{sel}}([\mathbf{F}_{\text{hi}}, \{\bar{\mathbf{P}}_k\}_{k \in \mathcal{K}}, \mathbf{B}])) \quad (13)$$

where g_{sel} is a 1×1 convolution that outputs $|\mathcal{K}|$ channels, and the softmax is applied along the scale dimension. Therefore, for every spatial location p , the weights satisfy $\pi_k(p) \geq 0$ and $\sum_{k \in \mathcal{K}} \pi_k(p) = 1$. The injected feature is consequently a normalized mixture of semantic scales rather than an unconstrained accumulation of pyramid features.

The injected semantic feature is

$$\mathbf{P}_{\text{inj}} = \sum_{k \in \mathcal{K}} \pi_k \odot \psi_k(\bar{\mathbf{P}}_k), \mathbf{F}_{\text{hi}}^+ = \mathbf{F}_{\text{hi}} + \mathbf{P}_{\text{inj}} \quad (14)$$

where $\psi_k(\cdot)$ is a 1×1 channel adapter.

3.4 Boundary-Conditioned Region-Adaptive Alignment Upsampling

As illustrated in Fig. 4, BCRAU combines a full-map bilinear path with a regionally activated adaptive path. Bilinear interpolation provides the default aligned feature, while content-adaptive reassembly replaces the bilinear result only within boundary-related regions.

Regional mixture. The two-path mixture shown in Fig. 4 is formulated as follows. Given a source feature map $\mathbf{X} \in \mathbb{R}^{\frac{H}{k} \times \frac{W}{k} \times C}$ to be upsampled from stride k to r , the bilinear result is

$$\tilde{\mathbf{X}} = \mathcal{U}_{\text{bil}}(\mathbf{X}) \quad (15)$$

Let $\tilde{\mathbf{X}}$ denote the adaptive reassembly result, and let $\mathbf{M}_{\text{align}}$ denote the binary boundary band $\mathbf{M}$ resized to the target resolution by nearest-neighbor interpolation. The final aligned feature is

$$\mathcal{U}_{k \rightarrow r}(\mathbf{X}) = \mathbf{M}_{\text{align}} \odot \hat{\mathbf{X}} + (1 - \mathbf{M}_{\text{align}}) \odot \tilde{\mathbf{X}} \quad (16)$$

where $\mathbf{M}_{\text{align}}$ is broadcast along the channel dimension.

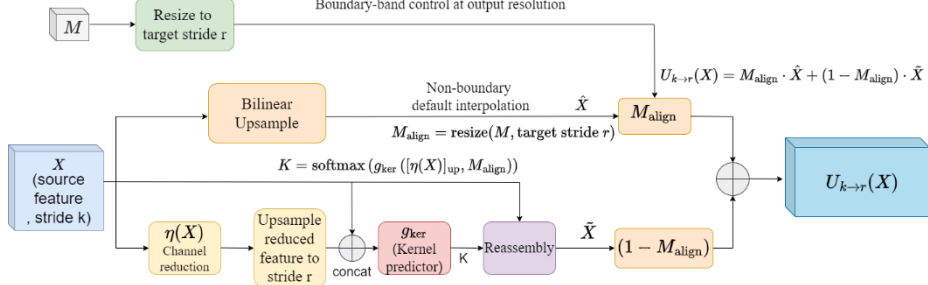


Fig. 4. Illustration of the proposed BCRAU module. BCRAU applies content-adaptive upsampling only within boundary regions while preserving bilinear interpolation elsewhere.

Content-adaptive reassembly. Reassembly weights over an $N \times N$ neighborhood are predicted as

$$\mathbf{K} = \text{softmax}\left(g_{\text{ker}}([u_{\text{bil}}(\eta(\mathbf{X})), \mathbf{M}_{\text{align}}])\right) \quad (17)$$

where $N = 3$ by default, $\eta(\cdot)$ reduces the channel number to $C/4$, and $g_{\text{ker}}(\cdot)$ is implemented by two depthwise separable 3×3 convolutions. The softmax is applied along the neighborhood dimension. For target location p , the adaptive output is

$$\hat{\mathbf{X}}(p) = \sum_{q \in \mathcal{N}(p)} \mathbf{K}(p, q) \tilde{\mathbf{X}}(q) \quad (18)$$

where $\mathcal{N}(p)$ denotes the $N \times N$ target-grid neighborhood centered at p .

Implementation. BCRAU is implemented with an optimized tile-level strategy. The target feature map is partitioned into regular tiles, and a tile is activated if it overlaps with $\mathbf{M}_{\text{align}}$. Adaptive reassembly is evaluated only for activated tiles, while inactive tiles reuse the bilinear result. For reproducibility, we also provide a dense PyTorch fallback that implements bilinear upsampling, kernel prediction, adaptive reassembly, mask processing, and output mixing with standard dense tensor operations. Unless otherwise specified, the reported BCRAU latency refers to the optimized tile-level implementation and includes all operator costs.

4 Experiments

We evaluate BCSNet on three semantic segmentation benchmarks and conduct controlled ablation studies on Cityscapes to analyze the contribution of each proposed component. To make the accuracy–efficiency comparison more interpretable, we explicitly specify the evaluation protocol.

Protocol. The main accuracy–efficiency comparison is reported on Cityscapes, while all ablation studies are conducted on the Cityscapes validation set under identical training and inference settings. In Table 1, “*” denotes a model reimplemented and measured by us on a single NVIDIA RTX 4090 GPU under the listed input resolution. FPS values of unmarked methods are taken from the corresponding publications and

are not directly hardware-normalized. The proposed BCSNet variants and all marked baselines are evaluated using the same timing procedure. Unless otherwise specified, BCRAU latency is measured using the optimized tile-level implementation and includes all mask-processing and adaptive-reassembly operations.

4.1 Datasets

Cityscapes. Cityscapes [2] provides 5,000 finely annotated images with a native resolution of 1024×2048 , including 2,975 images for training, 500 for validation, and 1,525 for testing. The test annotations are withheld for online evaluation, and 19 semantic classes are used.

CamVid. CamVid [1] contains 701 labeled frames at a native resolution of 960×720 . We follow the standard 367/101/233 train/validation/test split and report results under the 11-class protocol.

COCO-Stuff10K. COCO-Stuff10K [7] contains 10,000 densely annotated images, with 9,000 images for training and 1,000 images for validation. Unlabeled pixels are ignored during training and evaluation. We use an input resolution of 512×512 for both training and evaluation on this dataset.

4.2 Evaluation Metrics

We use mean Intersection over Union (mIoU) as the primary accuracy metric. For each class k , IoU and mIoU are computed as

$$\text{IoU}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FP}_k + \text{FN}_k}, \text{mIoU} = \frac{1}{K} \sum_{k=1}^K \text{IoU}_k \quad (19)$$

where TP_k , FP_k , and FN_k denote true-positive, false-positive, and false-negative pixels for class k , respectively, and K is the number of semantic classes. Inference efficiency is measured by frames per second (FPS):

$$\text{FPS} = \frac{1000}{\bar{t}} \quad (20)$$

where $\bar{t}$ denotes the average inference latency in milliseconds per image. We also report GFLOPs to reflect the computational cost of a single forward pass under the evaluated input resolution. To assess boundary quality, we report Boundary F-score (BF), where contour maps are extracted from segmentation masks following [26] and evaluated with a tolerance of 2 pixels. BF is used in the ablation study to evaluate whether the boundary-centric design improves contour-oriented behavior in addition to global region accuracy.

4.3 Main Results

Table 1 presents the accuracy–efficiency comparison on Cityscapes. Under the matched RTX 4090 evaluation protocol, BCSNet-S achieves 76.5% mIoU at 189.8 FPS with 8.7M parameters and 27.2 GFLOPs, demonstrating a favorable balance between

segmentation accuracy and computational efficiency. Compared with the reimplemented DDRNet-23-Slim baseline, BCSNet-S improves mIoU by 0.2 percentage points while requiring fewer GFLOPs. GCNet-S reports slightly higher throughput but lower segmentation accuracy; however, its FPS is taken from the original publication and is therefore not directly comparable under the same hardware setting. With ImageNet-1K pretraining, BCSNet-S (IN-1K) further improves mIoU from 76.5% to 77.2% without changing the parameter count or computational complexity, while maintaining nearly identical inference speed because the network architecture remains unchanged.

Increasing the model capacity yields further accuracy improvements. BCSNet-M achieves 77.5% mIoU at 134.0 FPS, whereas BCSNet-L reaches 79.5% mIoU at 107.0 FPS, attaining the highest mIoU among the methods listed in Table 1 while retaining real-time inference speed under the measured setting.

Table 1. Accuracy efficiency comparison on Cityscapes. “*” denotes models reimplemented and retested by us on a single RTX 4090 GPU under the listed input resolution. “ImageNet” indicates whether ImageNet pretrained weights are used.

Method	Resolution	FPS ↑	Params (M) ↓	GFLOPs (G) ↓	Image Net	mIoU (%) ↑
ICNet [28]	1024 × 2048	181.3	24.9	74.3	×	71.6
Fast-SCNN* [14]	1024 × 2048	325.9	1.4	7.3	×	71.0
SwiftNetRN-18 [10]	1024 × 2048	39.9	11.8	104.0	✓	75.4
CGNet [19]	1024 × 2048	73.9	0.5	27.5	×	68.3
BiSeNetV1* [25]	1024 × 2048	116.8	13.3	118.0	✓	74.4
BiSeNetV2* [24]	1024 × 2048	132.4	3.4	98.4	×	73.6
STDC1* [4]	1024 × 2048	183.7	8.3	67.5	×	71.8
STDC2* [4]	1024 × 2048	132.8	12.3	93.8	×	74.9
HyperSeg-M [9]	512 × 1024	59.1	10.1	7.5	✓	76.2
HyperSeg-S [9]	768 × 1536	45.7	10.2	17.0	✓	78.2
PP-LiteSeg-T2 [12]	768 × 1536	96.0	–	–	✓	76.0
PP-LiteSeg-B2 [12]	768 × 1536	68.2	–	–	✓	78.2
DDRNet-23-Slim* [5]	1024 × 2048	166.4	5.7	36.3	×	76.3
DDRNet-23* [5]	1024 × 2048	106.0	20.3	143.0	×	78.0
PIDNet-S* [21]	1024 × 2048	128.7	7.7	47.3	×	75.4
PIDNet-M* [21]	1024 × 2048	78.2	28.7	177.0	×	77.2
PIDNet-L* [21]	1024 × 2048	64.2	37.3	275.0	×	77.8
SCTNet-S-Seg75* [22]	768 × 1536	168.7	4.6	16.0	×	74.7
SCTNet-B-Seg75* [22]	768 × 1536	157.3	17.4	63.2	×	78.5
GCNet-S [23]	1024 × 2048	193.3	9.2	45.2	×	75.3
GCNet-M [23]	1024 × 2048	105.0	34.2	178.0	×	77.0

Method	Resolution	FPS ↑	Params (M) ↓	GFLOPs (G) ↓	Image Net	mIoU (%) ↑
GCNet-L [23]	1024 × 2048	88.0	45.2	232.0	×	78.6
BCSNet-S	1024 × 2048	189.8	8.7	27.2	×	76.5
BCSNet-S (IN-1K)	1024 × 2048	188.8	8.7	27.2	✓	77.2
BCSNet-M	1024 × 2048	134.0	32.3	171.2	×	77.5
BCSNet-L	1024 × 2048	107.0	42.6	221.2	×	79.5

Table 2 presents the cross-dataset comparison and cumulative transfer ablation results on CamVid and COCO-Stuff10K. On CamVid, the dataset-specific BCSNet baseline achieves 75.8% mIoU at 166.7 FPS. Sequentially introducing BCFR and SL-FPN-SI increases the mIoU to 76.4% and 76.8%, respectively, while the complete model with BCRAU reaches 77.1% mIoU at 156.8 FPS. Compared with the baseline, the full configuration improves mIoU by 1.3 percentage points with a throughput reduction of 9.9 FPS. Among the listed methods, BCSNet achieves accuracy close to PIDNet-S, with 77.1% versus 77.2% mIoU, while reporting higher throughput of 156.8 versus 142.8 FPS. GCNet-S provides higher reported throughput but lower accuracy, obtaining 76.6% mIoU at 210.6 FPS.

Table 2. Cross-dataset comparison and cumulative transfer ablation on CamVid and COCO-Stuff10K. The BCSNet rows progressively add BCFR, SL-FPN-SI, and BCRAU to the corresponding dataset-specific baseline. All BCSNet variants within each dataset use identical training and evaluation settings.

Dataset	Method	Resolution	FPS↑	mIoU ↑
CamVid	DDRNet-23-Slim [5]	960 × 720	189.	76.2
	PIDNet-S [21]	960 × 720	142.	77.2
	GCNet-S [23]	960 × 720	210.	76.6
	BCSNet baseline	960 × 720	166.	75.8
	+ BCFR	960 × 720	163.	76.4
	+ SL-FPN-SI	960 × 720	160.	76.8
	+ BCRAU	960 × 720	156.	77.1
COCO-Stuff10K	AFFormer-B [3]	512 × 512	46.5	35.1
	RTFormer-B [17]	512 × 512	90.9	35.3
	SCTNet-B [22]	512 × 512	141.	35.9
	BCSNet baseline	512 × 512	155.	33.9
	+ BCFR	512 × 512	152.	34.4
	+ SL-FPN-SI	512 × 512	148.	34.9

Dataset	Method	Resolution	FPS↑	mIoU ↑
	+ BCRAU	512×512	145.	35.2

A consistent cumulative improvement is observed on COCO-Stuff10K. Starting from 33.9% mIoU at 155.7 FPS, BCFR and SL-FPN-SI progressively improve the result to 34.4% and 34.9%, respectively, and the complete configuration reaches 35.2% mIoU at 145.3 FPS. This corresponds to a 1.3-point improvement over the baseline with a throughput reduction of 10.4 FPS. BCSNet also outperforms AFFormer-B and approaches RTFormer-B and SCTNet-B in accuracy, while maintaining higher reported throughput than these methods in the listed comparison. Overall, the consistent gains across both datasets indicate that BCFR and SL-FPN-SI provide the primary accuracy improvements, whereas BCRAU contributes an additional 0.3-point gain on each dataset.

Overall, BCSNet achieves a competitive accuracy–efficiency trade-off across Cityscapes, CamVid, and COCO-Stuff10K. The consistent gains over the dataset-specific baselines further demonstrate the effectiveness and cross-dataset applicability of the proposed modules.

4.4 Ablation Studies

Table 3 reports the cumulative component ablation, boundary-mask sensitivity, and BCRAU implementation comparison on the Cityscapes validation set. Starting from the baseline, BCFR improves mIoU from 75.6% to 76.4% and BF from 79.1 to 79.8, demonstrating the effectiveness of boundary-guided cross-branch interaction and local refinement. Adding SL-FPN-SI further increases the performance to 77.0% mIoU and 80.2 BF, confirming the benefit of selective multi-scale semantic injection. With BCRAU, the complete model achieves 77.2% mIoU and 80.5 BF at 189.8 FPS. Compared with the baseline, the full configuration provides gains of 1.6 mIoU points and 1.4 BF points, while reducing throughput from 201.4 to 189.8 FPS.

The sensitivity results support the use of $\tau = 0.5$ and a 2-pixel dilation as the default boundary-mask settings. Reducing the threshold to 0.3 enlarges the activated boundary region, decreasing throughput from 189.8 to 186.8 FPS without improving accuracy. Increasing it to 0.7 raises throughput to 191.2 FPS but reduces mIoU and BF to 76.9% and 80.1, respectively. Similarly, a 1-pixel dilation provides slightly higher throughput but lower accuracy, whereas a 3-pixel dilation expands the processed region and reduces FPS to 187.6 without additional performance gains. The default configuration therefore provides the most favorable accuracy–efficiency trade-off among the evaluated settings.

The final comparison evaluates the dense PyTorch fallback and the optimized tile-level implementation of BCRAU. Both implementations achieve identical results of 77.2% mIoU and 80.5 BF, indicating that the execution strategy does not affect prediction quality. The tile-level implementation increases throughput from 171.8 to 189.8 FPS and reduces peak memory consumption from 2876 MB to 2412 MB. These results

demonstrate that regional execution improves the efficiency of BCRAU while preserving its segmentation performance.

Table 3. Component ablation, parameter sensitivity, and BCRAU implementation comparison on the Cityscapes validation set. Each sensitivity experiment varies one factor at a time while keeping the remaining settings fixed. Default settings are shown in bold. GFLOPs denote the dense-equivalent arithmetic cost, whereas FPS reflects the actual regional execution strategy.

Setting	mIoU $\uparrow$	BF $\uparrow$	FPS $\uparrow$	GFLOPs	Peak Mem.
<i>Component ablation</i>					
Baseline	75.6	79.1	201.4	20.4	—
+ BCFR	76.4	79.8	197.0	21.8	—
+ SL-FPN-SI	77.0	80.2	193.1	24.3	—
+ BCRAU	77.2	80.5	189.8	27.2	2412 MB
<i>Boundary-threshold sensitivity</i>					
$\tau = 0.3$	77.1	80.4	186.8	27.2	—
$\tau = 0.5$	77.2	80.5	189.8	27.2	—
$\tau = 0.7$	76.9	80.1	191.2	27.2	—
<i>Boundary-band dilation sensitivity</i>					
$d = 1$ pixel	77.0	80.2	190.5	27.2	—
$d = 2$ pixels	77.2	80.5	189.8	27.2	—
$d = 3$ pixels	77.1	80.4	187.6	27.2	—
<i>BCRAU implementation</i>					
Dense PyTorch fallback	77.2	80.5	171.8	27.2	2876 MB
Tile-level BCRAU ($T = 8$)	77.2	80.5	189.8	27.2	2412 MB

Fig. 5 analyzes the internal design choices of the proposed modules. For BCFR, both the routing-only and refinement-only variants improve over the baseline, while the complete BCFR design achieves the best result in this group. This indicates that boundary-guided routing and local refinement provide complementary effects. For SL-FPN-SI, naive concatenation improves performance, but selective injection with $K = \{16, 32, 64\}$ achieves a better accuracy–efficiency balance. Adding the stride-8 level further increases mIoU to 77.1%, but reduces FPS to 190.6, suggesting that the default scale set is more suitable for the targeted real-time setting.

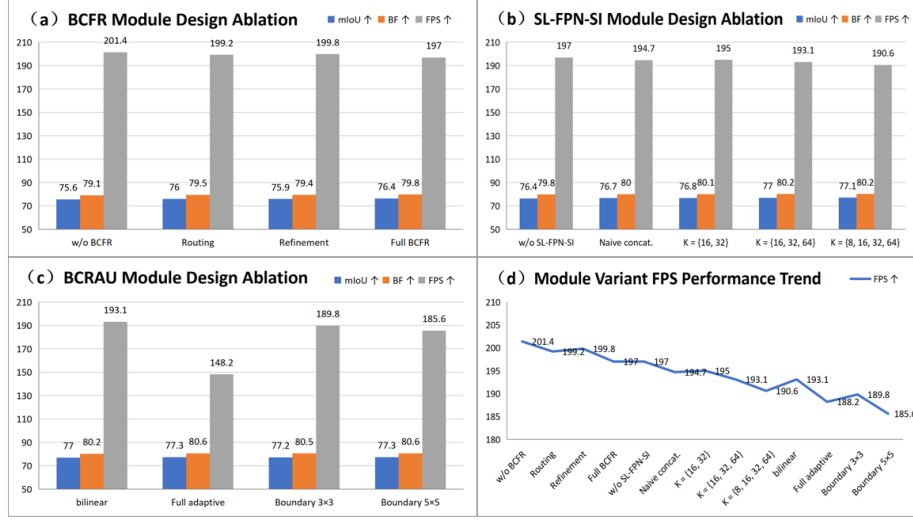


Fig. 5. Ablation results of the proposed modules on the Cityscapes validation set. (a) Design ablation of BCFR. (b) Design ablation of SL-FPN-SI. (c) Design ablation of BCRAU. (d) FPS trend across representative module variants. mIoU and BF measure segmentation accuracy and boundary quality, respectively, while FPS reflects inference efficiency.

For BCRAU, full-image adaptive upsampling improves mIoU from 77.0% to 77.3% and BF from 80.2 to 80.6. In contrast, the proposed boundary-only adaptive strategy preserves most of the accuracy gain while maintaining real-time throughput. The boundary-only 3×3 variant obtains 77.2% mIoU and 80.5 BF at 189.8 FPS, while the 5×5 variant reaches 77.3% mIoU and 80.6 BF at 185.6 FPS. These results confirm that dense adaptive reassembly is the main source of latency and that restricting adaptive alignment to boundary regions provides a more favorable trade-off between boundary-sensitive alignment and inference efficiency.

4.5 Qualitative Visualizations

Fig. 6 presents qualitative comparisons among PIDNet, GCNet, and the proposed BCSNet on representative Cityscapes validation images. The highlighted regions mainly cover boundary-sensitive areas, including sidewalks, poles, traffic signs, road-side structures, and distant small objects. Compared with the competing methods, BCSNet produces more continuous object contours and more stable class transitions around these regions. This suggests that the proposed boundary-centric design helps reduce the boundary blurring and local semantic confusion commonly observed in high-resolution urban-scene segmentation.

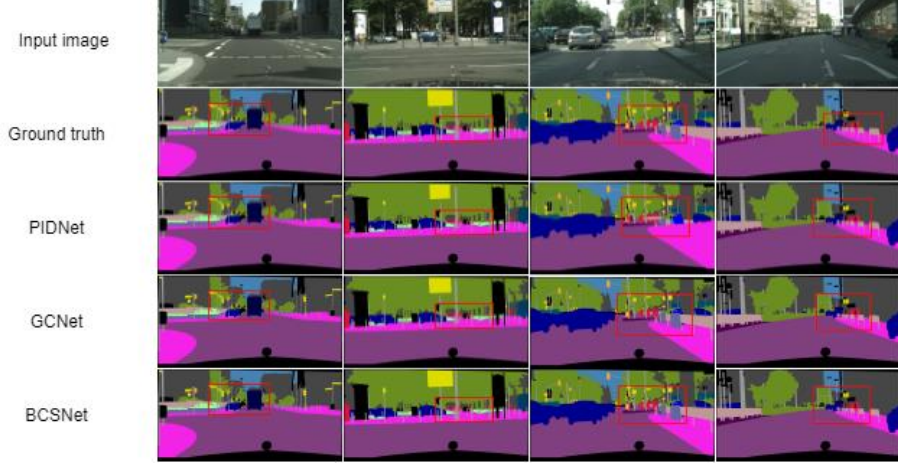


Fig. 6. Qualitative comparison on representative Cityscapes validation images among PIDNet, GCNet, and the proposed BCSNet. Red boxes highlight boundary-sensitive regions and small or thin objects, where accurate contour preservation and semantic consistency are challenging.

In addition to contour preservation, BCSNet maintains more coherent predictions for large semantic regions such as roads, buildings, and vegetation, while better retaining thin structures and small objects. These observations are consistent with the quantitative ablation results, where the proposed modules improve both mIoU and BF. Overall, the qualitative results provide visual evidence that BCFR, SL-FPN-SI, and BCRAU jointly contribute to boundary-aware fusion, multi-scale semantic enhancement, and boundary-sensitive feature alignment.

5 Conclusion

This paper introduced BCSNet, a boundary-centric framework for real-time semantic segmentation. The framework uses a shared boundary representation to coordinate cross-branch interaction, semantic-scale selection, and regional adaptive alignment within a two-branch architecture. BCFR performs boundary-guided feature routing and local logit refinement, SL-FPN-SI forms a normalized mixture of semantic scales, and BCRAU restricts content-adaptive reassembly to boundary-related regions. Experiments on Cityscapes, CamVid, and COCO-Stuff10K show that BCSNet provides competitive segmentation accuracy and throughput under the reported settings. The ablation results further indicate that the proposed components are associated with improvements in both mIoU and boundary F-score. The current study has several limitations. First, latency is evaluated primarily on a desktop GPU and does not establish performance on embedded or low-power hardware. Second, the cross-dataset evaluation does not directly measure robustness to adverse illumination, weather, occlusion, or domain shift. Third, the boundary analysis relies mainly on an aggregate BF metric and does not separate errors by class, object size, or boundary distance. Future work will investigate

hardware-specific optimization, model compression and quantization, adverse-condition robustness, and more detailed boundary-error analysis.

Acknowledgments. This study was funded by the Key R&D Program of Shandong Province, China (2024CXGC010501), National Natural Science Foundation of China (62501338).

Disclosure of Interests. The authors declare that they have no competing interests.

References

1. Brostow, G.J., Fauqueur, J., Cipolla, R.: Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters* **30** (2), 88–97 (2009)
2. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The Cityscapes dataset for semantic urban scene understanding. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 3213–3223 (2016)
3. Dong, B., Wang, P., Wang, F.: Head-free lightweight semantic segmentation with linear transformer. In: *Proceedings of the AAAI conference on artificial intelligence*. vol. 37, pp. 516–524 (2023)
4. Fan, M., Lai, S., Huang, J., Wei, X., Chai, Z., Luo, J., Wei, X.: Rethinking BiSeNet for real-time semantic segmentation. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 9716–9725 (2021)
5. Hong, Y., Pan, H., Sun, W., Jia, Y.: Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes. *arXiv preprint arXiv:2101.06085* (2021)
6. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 2117–2125 (2017)
7. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: *European conference on computer vision*. pp. 740–755. Springer (2014)
8. Lo, S.Y., Hang, H.M., Chan, S.W., Lin, J.J.: Efficient dense modules of asymmetric convolution for real-time semantic segmentation. *arXiv preprint arXiv:1809.06323* (2018)
9. Nirkin, Y., Wolf, L., Hassner, T.: Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 4061–4070 (2021)
10. Orsic, M., Kreso, I., Bevandic, P., Segvic, S.: In defense of pre-trained ImageNet architectures for real-time semantic segmentation of road-driving images. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 12607–12616 (2019)
11. Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: ENet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147* (2016)
12. Peng, J., Liu, Y., Tang, S., Hao, Y., Chu, L., Chen, G., Wu, Z., Chen, Z., Yu, Z., Du, Y., et al.: PP-LiteSeg: A superior real-time semantic segmentation model. *arXiv preprint arXiv:2204.02681* (2022)
13. Poudel, R.P.K., Bonde, U., Liwicki, S., Zach, C.: ContextNet: Exploring context and detail for semantic segmentation in real-time. In: *Proceedings of the British Machine Vision Conference (BMVC)* (2018)
14. Poudel, R.P., Liwicki, S., Cipolla, R.: Fast-SCNN: Fast semantic segmentation network. *arXiv preprint arXiv:1902.04502* (2019)

15. Takikawa, T., Acuna, D., Jampani, V., Fidler, S.: Gated-SCNN: Gated shape cnns for semantic segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 5229–5238 (2019)
16. Wan, Q., Huang, Z., Lu, J., Yu, G., Zhang, L.: SeaFormer: Squeeze-enhanced axial transformer for mobile semantic segmentation. In: The eleventh international conference on learning representations (2023)
17. Wang, J., Gou, C., Wu, Q., Feng, H., Han, J., Ding, E., Wang, J.: RTFormer: Efficient design for real-time semantic segmentation with transformer. *Advances in neural information processing systems* **35**, 7423–7436 (2022)
18. Wang, J., Chen, K., Xu, R., Liu, Z., Loy, C.C., Lin, D.: CARAFE: Content-aware reassembly of features. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 3007–3016 (2019)
19. Wu, T., Tang, S., Zhang, R., Cao, J., Zhang, Y.: CGNet: A light-weight context guided network for semantic segmentation. *IEEE Transactions on Image Processing* **30**, 1169–1179 (2020)
20. Xie, S., Tu, Z.: Holistically-nested edge detection. In: Proceedings of the IEEE international conference on computer vision. pp. 1395–1403 (2015)
21. Xu, J., Xiong, Z., Bhattacharyya, S.P.: PIDNet: A real-time semantic segmentation network inspired by pid controllers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 19529–19539 (2023)
22. Xu, Z., Wu, D., Yu, C., Chu, X., Sang, N., Gao, C.: SCTNet: Single-branch cnn with transformer semantic information for real-time segmentation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 38, pp. 6378–6386 (2024)
23. Yang, G., Wang, Y., Shi, D., Wang, Y.: Golden cudgel network for real-time semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 25367–25376 (2025)
24. Yu, C., Gao, C., Wang, J., Yu, G., Shen, C., Sang, N.: BiSeNet V2: Bilateral network with guided aggregation for real-time semantic segmentation. *International journal of computer vision* **129** (11), 3051–3068 (2021)
25. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: BiSeNet: Bilateral segmentation network for real-time semantic segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 325–341 (2018)
26. Yuan, Y., Xie, J., Chen, X., Wang, J.: SegFix: Model-agnostic boundary refinement for segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 489–506 (2020)
27. Zhang, W., Huang, Z., Luo, G., Chen, T., Wang, X., Liu, W., Yu, G., Shen, C.: TopFormer: Token pyramid transformer for mobile semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12083–12093 (2022)
28. Zhao, H., Qi, X., Shen, X., Shi, J., Jia, J.: ICNet for real-time semantic segmentation on high-resolution images. In: Proceedings of the European conference on computer vision (ECCV). pp. 405–420 (2018)