machines

MDPI

Article

ERSGNet: An Edge-Guided Residual Spatial Gating Fusion Network for Real-Time Semantic Segmentation

Zheming Wu [1], Lin Jiang [1,2,*] and Tete Wang [1]

[1] School of Information Science and Engineering, Northeastern University, Shenyang 110819, China
[2] State Key Laboratory of Synthetical Automation for Process Industries, Shenyang 110819, China
* Correspondence: jianglin@ise.neu.edu.cn

Abstract

With the development of autonomous driving and general real-time vision applications, semantic segmentation is increasingly expected to provide reliable pixel-level scene understanding under strict latency and computational constraints. Existing real-time semantic segmentation methods still struggle to effectively balance contextual semantics and local details during feature fusion. This difficulty may lead to boundary blur and the loss of fine-grained structures, and is mainly related to insufficient selectivity and boundary friendliness in the fusion process. To address this issue, the Edge-Guided Residual Spatial Gating Fusion Network (ERSGNet)is proposed as a three-branch decoupled framework for real-time semantic segmentation. In the ERSGNet, detail features F_d, contextual features F_c, and edge features F_e are separately extracted to reduce mutual interference among different types of information. Spatially Gated Interaction (SGI) is introduced to inject contextual semantics into the detail branch in a pixel-wise selective manner, the Parallel Context Pyramid (PCP) is used to enhance multi-scale contextual representation, and Edge-Guided Residual Spatial Gating Fusion (ERSG-Fuse) performs edge-guided adaptive fusion between F_d and F_c under the guidance of F_e. The ERSGNet is evaluated on the CamVid and Cityscapes datasets. On CamVid, the ERSGNet achieves 73.33% mean Intersection-over-Union (mIoU) at 166.27 frames per second (FPS). On Cityscapes, the ERSGNet achieves 69.21% mIoU at 53.93 FPS. These results suggest that the ERSGNet achieves a favorable accuracy–efficiency trade-off while supporting boundary-related and fine-grained structural representation.

Keywords: real-time semantic segmentation; feature fusion; edge-guided fusion; spatial gating; boundary refinement; lightweight networks

Academic Editor: Jiangjian Xiao

Received: 19 June 2026
Revised: 8 July 2026
Accepted: 10 July 2026
Published: 14 July 2026

1. Introduction

With the development of autonomous driving and general real-time vision applications, semantic segmentation is increasingly expected to provide reliable pixel-level scene understanding under strict latency and computational constraints. Therefore, maintaining segmentation accuracy while ensuring inference efficiency has become a key issue in real-time visual perception and road-scene semantic segmentation, where efficient feature extraction and robust representation under complex scene variations are important concerns [1–5]. Compared with image classification or object detection, semantic segmentation requires stable region-level semantic discrimination. It also requires accurate preservation of boundary information, small targets, and fine-grained structures to support more complete and reliable scene representations [6]. Recent boundary-guided real-time segmen-

tation studies further confirm the role of boundary-related information in preserving local details and reducing boundary ambiguity in lightweight segmentation models [7].

Although real-time semantic segmentation has made significant progress in recent years, the key limitation of existing methods lies not merely in whether high-level semantic features and local detail features are extracted simultaneously, but in the difficulty of effectively balancing contextual semantics and local details during feature fusion. Representative real-time methods have improved semantic representation and overall segmentation performance through dual-branch modeling, dual-resolution interaction, efficient context aggregation, attention-based information fusion, adaptive feature fusion, and multi-path guidance [4,5,8–13]. However, this coordination remains insufficient in boundary regions, fine-grained structures, and small-scale targets. This limitation is mainly related to insufficient selectivity, position sensitivity, and boundary friendliness in the fusion process.

To alleviate this feature-fusion difficulty in real-time semantic segmentation, the Edge-Guided Residual Spatial Gating Fusion Network (ERSGNet) is proposed as a three-branch decoupled framework. The three branches are responsible for local detail extraction, contextual semantic modeling, and boundary information encoding, thereby providing a clearer division of roles for subsequent feature interaction and fusion. At the intermediate interaction stage, a pixel-wise selective interaction mechanism is adopted to inject contextual semantics into the detail branch on demand, which helps preserve detail fidelity before final fusion. At the end of the context branch, parallel context aggregation is employed to enhance multi-scale semantic representation and provide a stronger contextual prior while maintaining deployment efficiency. Accordingly, the ERSGNet produces detail features F_d, contextual features F_c, and edge features F_e for subsequent feature fusion. At the final fusion stage, Edge-Guided Residual Spatial Gating Fusion (ERSG-Fuse) is proposed to generate a lightweight single-channel spatial gate from edge features. The spatial gate determines whether each pixel location should rely more on detail features or contextual semantics. Residual gated fusion is then combined with a learnable boundary residual injection, so that boundary information can guide the final fusion and enhance the final representation in a controllable manner. This design helps improve the segmentation quality of boundary regions and small targets while maintaining real-time performance.

The main contributions are summarized as follows:

- To address the problem that contextual semantics tend to overwhelm local details during feature fusion in real-time semantic segmentation, a three-branch decoupled framework is developed for detail, context, and edge modeling. The three branches output F_d, F_c, and F_e, respectively, thus providing a decoupled feature foundation for local detail preservation, contextual semantic modeling, and boundary enhancement.
- To alleviate the interference of strong semantic responses with detail representations during intermediate feature interaction, Spatially Gated Interaction (SGI) is introduced. SGI realizes on-demand injection of contextual semantics through a pixel-wise gating mechanism. In addition, a Parallel Context Pyramid (PCP) is introduced at the end of the context branch to aggregate multi-scale contextual semantics in parallel, thereby enhancing contextual representation while maintaining efficient deployment.
- To better balance contextual semantics and local details at the final fusion stage, Edge-Guided Residual Spatial Gating Fusion (ERSG-Fuse) is designed. Through a boundary-guided single-channel spatial gate, residual gated fusion, and a learnable boundary residual injection, ERSG-Fuse realizes position-sensitive and boundary-aware selective fusion between detail features and contextual semantics. This design helps improve the representation of boundary regions and small targets under a comparable computational cost.

2. Related Work

From the perspective of existing research routes, related methods can be grouped into three categories.

2.1. Semantic Representation Enhancement Methods

The first category, represented by SegFormer, Atrous Spatial Pyramid Pooling (ASPP)-based encoder–decoder semantic segmentation studies, recent DeepLabV3+-based studies for autonomous driving segmentation, and the Lightweight Real-Time Semantic Segmentation Network with Efficient Transformer and CNN (LETNet), focuses on enhancing high-level semantic representation through stronger semantic modeling, global modeling, and multi-scale representation [6,14–16]. In addition, recent studies on road-scene segmentation indicate that robust semantic representation and structured semantic organization remain important issues in semantic segmentation [5]. LETNet further shows that a U-shaped CNN can be combined with a Transformer to strengthen global modeling in lightweight real-time segmentation. Overall, these methods improve region-level category discrimination and confirm the effectiveness of stronger semantic modeling in complex driving scenes. However, their main focus is stronger semantic representation, while local detail preservation during feature fusion under real-time constraints is less explicitly addressed.

2.2. Lightweight Efficient Segmentation Methods

The second category, represented by Fast Semantic Segmentation Networks (Fast-SCNNs), Context-Guided Networks (CGNets), Efficient Dense Modules with Asymmetric Convolution (EDANet), the Efficient Spatial Pyramid Network v2 (ESPNetv2), and Efficient Bottleneck Unit Networks (EBUNets), together with recent hardware-aware lightweight models, mainly aims to reduce computational cost and shorten the inference path through lightweight backbones, early downsampling, and efficient convolutional structures [1–3,17–21]. These methods provide an efficiency foundation for real-time segmentation, and recent hardware-aware designs further show that deployment-oriented optimization can improve practical efficiency. For example, Fast-SCNNs use learning-to-downsample to shorten the inference path, EDANets combine asymmetric convolution with dense connectivity, and ESPNetv2 and EBUNets reduce complexity through efficient convolutional or bottleneck designs. However, early downsampling and compact feature extraction may weaken high-resolution details. Subsequent lightweight decoding or compensation structures are also often insufficient to fully recover boundary cues and fine-grained shape information, which increases the risk of insufficient detail preservation during feature fusion.

2.3. Branch-Wise Feature Coordination and Boundary-Aware Fusion Methods

The third category, represented by BiSeNetV2, Deep Dual-Resolution Networks (DDRNets), PP-LiteSeg, Lightweight Attention-Guided Asymmetric Networks (LAANets), Fast Bilateral Symmetrical Networks (FBSNets), attention-guided decoding methods, and recent adaptive feature-fusion and boundary-guided real-time segmentation networks, has begun to separately model detail features and contextual semantics through dual-branch, multi-path, guided-interaction, adaptive-fusion, or boundary-guided strategies [4,5,7–10,13,22,23]. These studies indicate that feature coordination and boundary-aware representation remain active issues in real-time semantic segmentation. Although this category is closer to the core issue, its fusion processes still tend to emphasize overall information integration and regional semantic consistency. For instance, BiSeNetV2 and DDRNets focus more on the effective exchange and aggregation of features across different branches or resolutions,

whereas PP-LiteSeg and LAANets place greater emphasis on efficient context enhancement and attention-guided feature aggregation. FBSNets further use semantic information and spatial detail branches with feature aggregation to balance contextual semantics and spatial details. Similarly, guided multi-path and boundary-guided methods mainly improve global representation, adaptive feature fusion, or the efficiency of utilizing guiding information. Therefore, although existing methods have achieved significant progress in real-time segmentation, targeted modeling for selective and boundary-aware coordination during feature fusion remains insufficient.

Overall, existing studies have advanced real-time semantic segmentation from three main perspectives: semantic representation enhancement, lightweight efficient segmentation, and branch-wise feature coordination with boundary-aware fusion. These studies provide useful foundations for balancing segmentation accuracy and inference efficiency. However, under real-time constraints, how to selectively coordinate contextual semantics, local details, and boundary-related information during feature fusion remains insufficiently addressed. This motivates the design of an ERSGNet, which introduces a three-branch decoupled framework and an edge-guided residual spatial gating fusion module to improve boundary-aware and position-adaptive feature fusion.

3. Materials and Methods

3.1. Overall Architecture of the ERSGNet

To address the feature-fusion difficulty described above, the ERSGNet adopts a three-branch decoupled feature extraction framework, consisting of a detail branch D, a context branch C, and an edge branch E. These branches explicitly model local details, contextual semantics, and boundary information, thus providing a clearer feature basis for subsequent interaction and fusion. As illustrated in Figure 1, the overall network consists of a shared encoder, three task-oriented branches, and a final feature fusion module.

Among the three branches, the detail branch maintains a relatively high spatial resolution in order to preserve local details and spatial structures. However, the semantic discrimination capability of the detail branch is limited. Therefore, SGI is introduced at the intermediate stages of the detail branch to supplement semantic information from the context branch through pixel-wise on-demand injection. This design helps the detail branch retain spatial resolving capability before final fusion while reducing premature domination by strong semantics during interaction. Thus, SGI supports subsequent fusion by preserving detail fidelity.

The context branch continuously downsamples features in order to enlarge the receptive field and stabilize low-frequency semantics. At the end of the context branch, the PCP is used to aggregate contextual information from different ranges in parallel, thereby forming a stronger and more efficient global semantic representation. Rather than directly producing the final prediction, the PCP provides a more reliable semantic prior for fusion by improving contextual stability.

The edge branch is responsible for extracting high-frequency structural information near boundaries and primarily serves to guide the final fusion decision. The rationale behind the edge branch is that semantics within object interiors are usually relatively consistent, whereas semantic changes are mainly concentrated around boundary regions. Therefore, boundary locations are where local details and contextual semantics are more likely to require differentiated treatment. Based on this observation, the edge branch obtains boundary-related responses by introducing semantic differential information and further refines structural cues in subsequent layers. Accordingly, SGI and the PCP support edge-guided final fusion from the perspectives of detail fidelity and contextual stability, respectively, while ERSG-Fuse performs the final coordination between contextual seman-

tics and local details. Finally, the fused features are fed into the segmentation head to generate the prediction, and an auxiliary prediction head can be employed during training to enhance intermediate supervision.

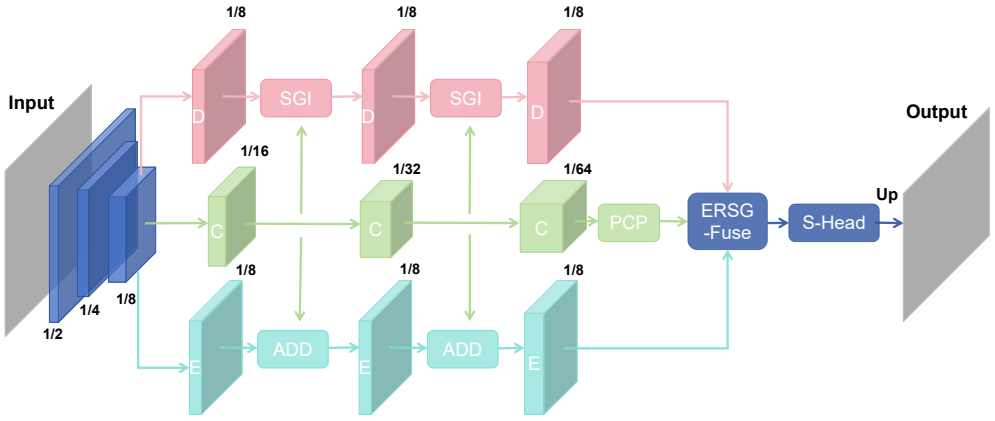

Figure 1. Overall architecture of the Edge-Guided Residual Spatial Gating Fusion Network (ERSGNet). The network consists of a shared encoder stem and three decoupled branches for detail, context, and edge modeling. Spatially Gated Interaction (SGI) is applied in the detail branch, the Parallel Context Pyramid (PCP) is used in the context branch, and Edge-Guided Residual Spatial Gating Fusion (ERSG-Fuse) performs the final fusion. ADD denotes element-wise addition, Up denotes upsampling, and S-Head denotes the segmentation head.

3.2. SGI: Spatially Gated Interaction

In multi-branch real-time segmentation, the context branch usually contains richer semantic information, but its low-resolution representation tends to lose boundary and geometric details. By contrast, the detail branch can preserve spatial structures, yet often lacks sufficient semantic discrimination capability. With naive fusion strategies such as direct addition or concatenation, strong semantic features may interfere with detail representations, causing blurred boundaries and the loss of small-target structures. To address this issue, SGI is introduced at the intermediate interaction stage. Through pixel-wise semantic injection from the context branch, the detail branch can absorb contextual semantics while preserving local detail representations. In this way, SGI prepares the detail branch for subsequent edge-guided selective fusion by improving semantic supplementation and maintaining detail fidelity.

The core of SGI lies in pixel-wise gated interaction. Figure 2 provides an overview of SGI. For the corresponding pixel vectors from the detail branch and the context branch, SGI generates a weighted output by computing a gating coefficient. The gating coefficient can be regarded as a learned scalar field over the spatial domain, and each value assigns an adaptive interpolation weight to the corresponding pixel location. Accordingly, the SGI output can be interpreted as a pixel-wise gated combination of the detail and context responses. Boundary and fine-grained structural regions can retain more detail information, whereas interior regions can absorb more contextual semantics. This interaction allows the detail branch to maintain high-resolution representations while receiving stable semantic supplementation. In ERSGNet, SGI is applied twice in the detail branch to enhance detail representations at different scales before the final fusion stage.

In implementation, before computing the gating coefficient, the context feature is first spatially aligned with the detail feature. The pixel-wise feature vectors from the detail branch and the context branch are denoted as $\mathbf{v}_d$ and $\mathbf{v}_c$, respectively. Two branch-specific projection functions, $f_d(\cdot)$ and $f_c(\cdot)$, are then used to map the two feature vectors into a common embedding space with the same dimension. Specifically, $f_d(\cdot)$ and $f_c(\cdot)$ are

implemented as independent 1×1 convolutional projections followed by normalization. The projection and gating computation are formulated as

$$\begin{aligned} f_d(\mathbf{v}_d) &= \mathrm{Norm}\Big(\mathrm{Conv}^d_{1\times1}(\mathbf{v}_d)\Big), \\ f_c(\mathbf{v}_c) &= \mathrm{Norm}\big(\mathrm{Conv}^c_{1\times1}(\mathbf{v}_c)\big), \\ \sigma &= \mathrm{Sigmoid}(f_d(\mathbf{v}_d) \cdot f_c(\mathbf{v}_c)). \end{aligned} \tag{1}$$

Here, $\mathrm{Conv}^d_{1\times1}$ and $\mathrm{Conv}^c_{1\times1}$ have independent learnable parameters. The gating coefficient σ can be interpreted as an affinity-like gating coefficient between the detail and context responses.

$$\mathrm{Out}_{\mathrm{SGI}} = \sigma \mathbf{v}_c + (1-\sigma)\mathbf{v}_d. \tag{2}$$

When σ is large, the output relies more on the semantically stronger context features; otherwise, it retains the detail-branch representation. In this way, SGI realizes selective semantic injection while helping prevent excessive detail suppression.

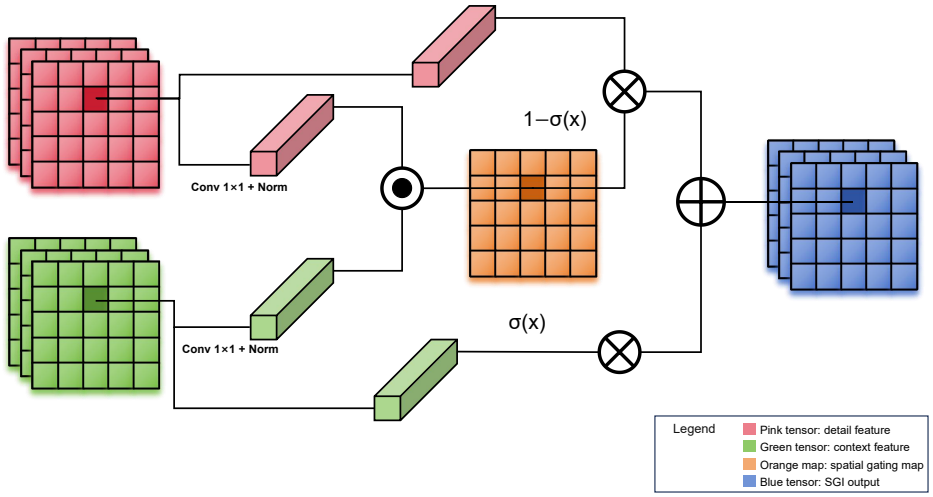

Figure 2. Structure of SGI (Spatially Gated Interaction) for pixel-wise selective semantic injection from the context branch into the detail branch. Conv denotes convolution, and Norm denotes normalization.

3.3. PCP: Parallel Context Pyramid

To construct a more stable global scene prior under real-time constraints, semantic segmentation methods usually employ pyramid pooling and its variants for contextual modeling. To balance contextual representation capability and inference efficiency, the Parallel Context Pyramid (PCP) is adopted. As shown in Figure 3, multi-scale connections are restructured in the PCP, and the channel allocation of each scale is appropriately compressed. This design improves module parallelism and enables efficient multi-scale context aggregation in lightweight models. The PCP provides a stronger and more reliable semantic prior for subsequent feature interaction from the perspective of contextual stability. This semantic prior helps the context branch produce more consistent and discriminative contextual semantics before the final fusion stage, thereby creating favorable conditions for edge-guided selective fusion.

Given an input feature map $X \in \mathbb{R}^{C_{\mathrm{in}} \times H \times W}$, the PCP constructs a multi-scale context representation through parallel pooling branches, and the resulting branch features are fused on a unified spatial grid to produce the context-enhanced feature F_c. The parallel pooling branches can be viewed as scale-dependent transformations defined on a shared spatial lattice, so that contextual responses from different receptive ranges remain directly comparable after alignment. The final aggregation preserves the base response while

incorporating multi-scale contextual increments under controlled channel compression. The computation can be abstracted as

$$\begin{aligned} B &= \mathcal{P}_0(X), \\ S_k &= \mathrm{Up}(\mathcal{P}_k(\mathrm{Pool}_k(X))) + B, \quad k = 1, \ldots, 4, \\ P &= \mathcal{G}_{3\times3}(\mathrm{Concat}(S_1, S_2, S_3, S_4)), \\ F_c &= \mathcal{P}_{\mathrm{out}}(\mathrm{Concat}(B, P)) + \mathcal{P}_{\mathrm{res}}(X). \end{aligned} \tag{3}$$

Here, $\mathrm{Pool}_k(\cdot)$ denotes context pooling at four scales, and $\mathrm{Up}(\cdot)$ indicates bilinear interpolation to align features to (H, W). The four pooling branches consist of adaptive global average pooling and average pooling with kernel/stride/padding settings of $k = 17$, $s = 8$, $p = 8$, $k = 9$, $s = 4$, $p = 4$, and $k = 5$, $s = 2$, $p = 2$. After each pooled response is projected by a 1×1 convolutional block, it is bilinearly upsampled to the spatial size (H, W) and added to the base response B. The operators $\mathcal{P}_0(\cdot)$, $\mathcal{P}_k(\cdot)$, $\mathcal{P}_{\mathrm{out}}(\cdot)$, and $\mathcal{P}_{\mathrm{res}}(\cdot)$ denote role-specific projection blocks based on 1×1 convolutions. Specifically, $\mathcal{P}_0(\cdot)$ generates the base response, $\mathcal{P}_k(\cdot)$ projects the pooled response at the k-th scale, $\mathcal{P}_{\mathrm{out}}(\cdot)$ performs output channel compression, and $\mathcal{P}_{\mathrm{res}}(\cdot)$ provides the residual shortcut. The operator $\mathcal{G}_{3\times3}(\cdot)$ denotes the grouped 3×3 convolutional refinement with four groups applied to the concatenated multi-scale features. In this formulation, B serves as the base reference, while S_k introduces aligned contextual responses from different ranges before joint compression and residual correction.

In ERSGNet, the PCP is deployed at the end of the context branch to aggregate multi-scale semantic features, providing a stronger context feature F_c for subsequent fusion.

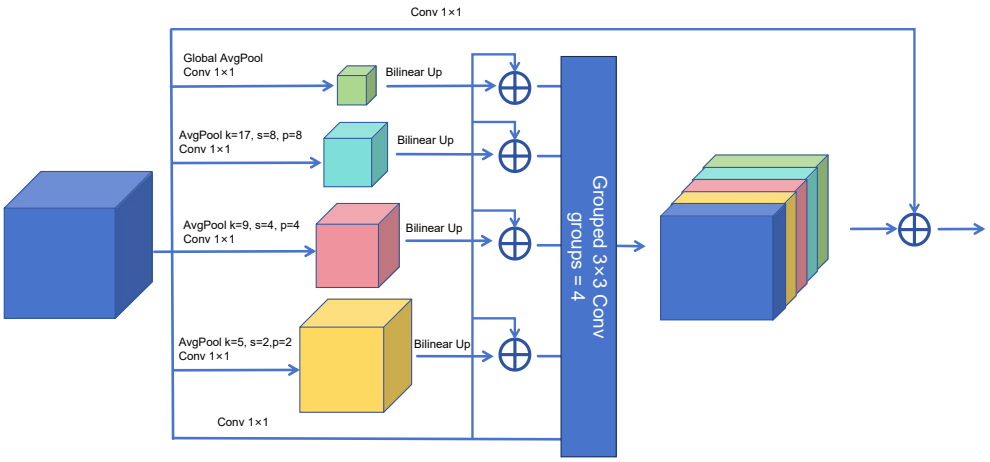

Figure 3. Structure of the PCP (Parallel Context Pyramid) for parallel multi-scale context aggregation in the context branch. AvgPool denotes average pooling.

3.4. ERSG-Fuse: Edge-Guided Residual Spatial Gating Fusion

ERSG-Fuse is proposed for the final tri-branch feature fusion under strict real-time budgets. The module helps reduce boundary blur and the loss of small-target structures, and it also provides an interpretable, low-cost, and optimization-stable interaction mechanism at the fusion stage. As shown in Figure 4, given the detail features F_d, contextual features F_c, and edge features F_e, ERSG-Fuse first aligns the three branches to the same channel dimension and then performs feature fusion.

Semantic changes are mainly concentrated around boundary regions, whereas object interiors usually require stronger regional consistency. Therefore, boundary locations require differentiated treatment during fusion. Unlike traditional fusion methods such as direct addition or concatenation, ERSG-Fuse does not assume that all spatial locations should combine detail features and contextual semantics in the same manner. Instead,

an edge-guided spatial gate is introduced to model location-dependent fusion weights. Direct addition or concatenation can be regarded as a spatially uniform fixed mixture, and this fixed mixture cannot explicitly express location-specific fusion requirements. By contrast, the edge-guided gate models the fusion coefficient as a position-dependent function conditioned on boundary information, thereby turning feature fusion from a fixed combination into a condition-adaptive combination. In this formulation, G can be viewed as a scalar field defined on the spatial domain, and each value determines the local interpolation ratio between detail and contextual responses. After spatial alignment and channel projection, the transformed detail, context, and edge features are denoted as $\tilde{F}_d$, $\tilde{F}_c$, and $\tilde{F}_e$, respectively. Based on this formulation, a single-channel spatial gate is generated from the transformed edge features:

$$G = \mathrm{Sigmoid}\big(C_g(\tilde{F}_e)\big), \qquad G \in [0,1]^{1\times H\times W}, \tag{4}$$

where $C_g(\cdot)$ denotes the 1×1 convolution used for gate generation. The single-channel spatial gate assigns a fusion weight to each pixel only in the spatial dimension, so the additional computational cost remains low and the fusion design remains suitable for real-time deployment. Using the transformed edge features $\tilde{F}_e$ as the gating condition makes the fusion weights more sensitive to boundary regions and structural variations. These regions often involve higher semantic uncertainty and stronger local structural variations, where a spatially uniform fixed fusion rule may be insufficient. By generating G from $\tilde{F}_e$, ERSG-Fuse adaptively adjusts the fusion weights according to boundary-related responses.

Subsequently, a residual gated fusion form is adopted to coordinate detail features and contextual semantics at the pixel level:

$$Y_0 = \tilde{F}_d + G \odot (\tilde{F}_c - \tilde{F}_d), \tag{5}$$

where G is broadcast along the channel dimension during the element-wise operation. This form defines a position-adaptive residual interpolation from $\tilde{F}_d$ to $\tilde{F}_c$. When G is small, the output preserves more detail features; when G becomes large, semantic information is progressively strengthened. Compared with direct replacement or simple weighted summation, this residual gated fusion does not abruptly disrupt the original representation of the detail branch, but instead regulates the intensity of semantic injection in a controlled manner. This continuous transition helps maintain optimization stability during feature fusion. On this basis, a learnable boundary residual injection is further introduced to enhance boundary representation:

$$Y_1 = Y_0 + \beta\, \tilde{F}_e. \tag{6}$$

Here, β is initialized to 0, so that the module behaves close to a stable baseline at the beginning of training. As training proceeds, the network can adaptively learn the strength of boundary enhancement, thereby achieving a better balance between stable convergence and boundary reinforcement. In this formulation, the boundary term acts as a learnable residual correction on top of Y_0 rather than as a direct replacement. This design keeps the contribution of edge information explicitly controllable. Finally, the fusion output is passed through a 3×3 convolutional refinement block $C_f(\cdot)$ to obtain the final fused feature $F_{\mathrm{fuse}} = C_f(Y_1)$. The final fused feature is then fed into the segmentation head to generate the prediction results. ERSG-Fuse achieves edge-guided, position-adaptive, and residually stable final fusion with relatively low additional cost, thereby improving the selectivity, position sensitivity, and boundary friendliness of feature fusion.

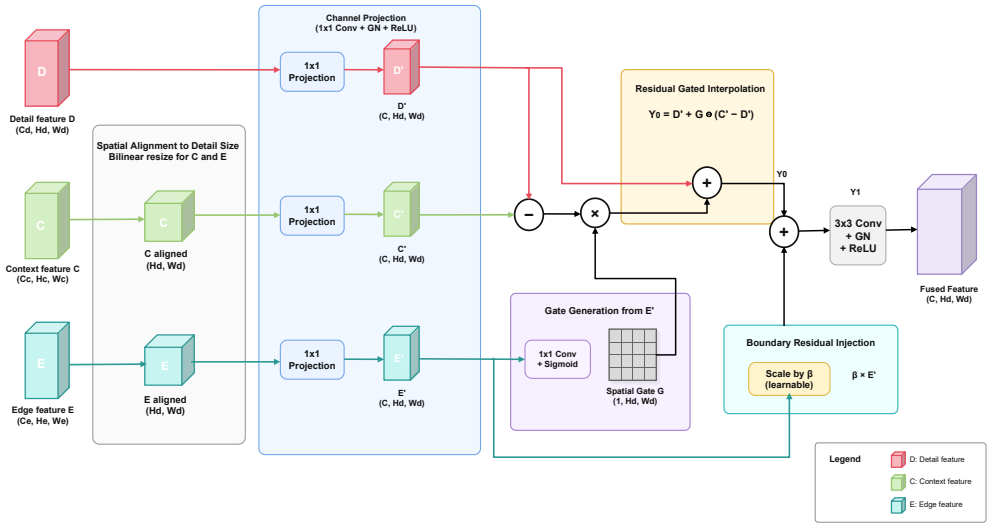

Figure 4. Structure of ERSG-Fuse (Edge-Guided Residual Spatial Gating Fusion) for edge-conditioned final tri-branch fusion. GN denotes group normalization, and ReLU denotes the rectified linear unit.

3.5. Algorithmic Summary of ERSGNet

To improve the readability of the proposed framework, Algorithm 1 summarizes the training and inference procedure of the ERSGNet. The main workflow, including feature extraction, SGI-based interaction, contextual aggregation, edge-guided fusion, prediction generation, and loss optimization, is described in the algorithm.

The training objective used in Algorithm 1 is detailed in the following subsection.

Algorithm 1 Training and inference pipeline of ERSGNet

Input: training samples $\mathcal{D} = \{(I, Y^{\mathrm{gt}}, B^{\mathrm{gt}})\}$; inference image I^{test}; loss weights $\lambda_{\mathrm{seg}}, \lambda_{\mathrm{aux}}, \lambda_{\mathrm{bnd}}, \lambda_{\mathrm{ba}}$.
Output: trained parameters Θ and final prediction $\hat{Y}$.
Notation: F_d, F_c, and F_e denote the final detail, context, and edge features before ERSG-Fuse, respectively. F_d^{aux} and F_e^{aux} denote the intermediate detail and edge features used for auxiliary semantic supervision and boundary supervision. $H_{\mathrm{seg}}(\cdot)$, $H_{\mathrm{aux}}(\cdot)$, and $H_{\mathrm{bnd}}(\cdot)$ denote the main segmentation head, auxiliary segmentation head, and boundary prediction head, respectively.

Initialize ERSGNet parameters Θ.
while training is not finished **do**
  for each mini-batch $(I, Y^{\mathrm{gt}}, B^{\mathrm{gt}}) \in \mathcal{D}$ **do**
  # Forward pipeline
    $F_0 \leftarrow \mathrm{Stem}(I)$;
    $(F_d, F_c, F_e, F_d^{\mathrm{aux}}, F_e^{\mathrm{aux}}) \leftarrow \mathrm{ThreeBranch}_{\mathrm{SGI,PCP}}(F_0)$;
    $F_{\mathrm{fuse}} \leftarrow \text{ERSG-Fuse}(F_d, F_c, F_e)$;
    $\hat{Y}_{\mathrm{seg}} \leftarrow H_{\mathrm{seg}}(F_{\mathrm{fuse}})$;
    $\hat{Y}_{\mathrm{aux}} \leftarrow H_{\mathrm{aux}}(F_d^{\mathrm{aux}})$;
    $\hat{B} \leftarrow H_{\mathrm{bnd}}(F_e^{\mathrm{aux}})$;
  # Loss mapping
    $\mathcal{L}_{\mathrm{seg}} \leftarrow \ell_{\mathrm{seg}}(\hat{Y}_{\mathrm{seg}}, Y^{\mathrm{gt}})$;
    $\mathcal{L}_{\mathrm{aux}} \leftarrow \ell_{\mathrm{aux}}(\hat{Y}_{\mathrm{aux}}, Y^{\mathrm{gt}})$;
    $\mathcal{L}_{\mathrm{bnd}} \leftarrow \ell_{\mathrm{bnd}}(\hat{B}, B^{\mathrm{gt}})$;
    $\mathcal{L}_{\mathrm{ba}} \leftarrow \ell_{\mathrm{ba}}(\hat{Y}_{\mathrm{seg}}, Y^{\mathrm{gt}}, \hat{B})$;
    $\mathcal{L} \leftarrow \lambda_{\mathrm{seg}}\mathcal{L}_{\mathrm{seg}} + \lambda_{\mathrm{aux}}\mathcal{L}_{\mathrm{aux}} + \lambda_{\mathrm{bnd}}\mathcal{L}_{\mathrm{bnd}} + \lambda_{\mathrm{ba}}\mathcal{L}_{\mathrm{ba}}$;
    $\Theta \leftarrow \mathrm{Optimizer}(\Theta, \nabla_\Theta \mathcal{L})$;
  end for
end while
 # Inference
$F_0^{\mathrm{test}} \leftarrow \mathrm{Stem}(I^{\mathrm{test}})$;
$(F_d^{\mathrm{test}}, F_c^{\mathrm{test}}, F_e^{\mathrm{test}}) \leftarrow \mathrm{ThreeBranch}_{\mathrm{SGI,PCP}}(F_0^{\mathrm{test}})$;
$F_{\mathrm{fuse}}^{\mathrm{test}} \leftarrow \text{ERSG-Fuse}(F_d^{\mathrm{test}}, F_c^{\mathrm{test}}, F_e^{\mathrm{test}})$;
$\hat{Y}_{\mathrm{seg}} \leftarrow H_{\mathrm{seg}}(F_{\mathrm{fuse}}^{\mathrm{test}})$;
$\hat{Y} \leftarrow \arg\max_c \hat{Y}_{\mathrm{seg}}^{(c)}$;
return $\hat{Y}$.

3.6. Training Objective and Loss Function

To improve both regional semantic consistency and boundary delineation quality, a joint training strategy that combines primary segmentation supervision, auxiliary semantic supervision, and boundary supervision is adopted. The primary segmentation branch focuses on learning region-level semantic consistency; the auxiliary semantic supervision stabilizes intermediate feature learning; and the boundary branch provides supervision for high-frequency structural cues, which helps suppress boundary information loss and enhances boundary delineation quality.

The overall training objective consists of four terms: the primary segmentation loss $\mathcal{L}_{\text{seg}}$, the auxiliary semantic loss $\mathcal{L}_{\text{aux}}$, the boundary supervision loss $\mathcal{L}_{\text{bnd}}$, and the boundary-aware semantic re-weighting term $\mathcal{L}_{\text{ba}}$. The total loss is formulated as a weighted sum:

$$\mathcal{L} = \sum_{i \in \{\text{seg,aux,bnd,ba}\}} \lambda_i \mathcal{L}_i. \tag{7}$$

where λ_{seg}, λ_{aux}, λ_{bnd}, and λ_{ba} are weighting coefficients. If a particular term is not used, its corresponding coefficient is set to zero.

In the implementation, the coefficients for the four loss terms are set to $\lambda_{\text{seg}} = 1.0$, $\lambda_{\text{aux}} = 0.4$, $\lambda_{\text{bnd}} = 20.0$, and $\lambda_{\text{ba}} = 1.0$. It should be emphasized that the relatively large coefficient $\lambda_{\text{bnd}} = 20.0$ is assigned to the boundary binary cross-entropy (BCE) loss $\mathcal{L}_{\text{bnd}}$, rather than to the auxiliary semantic loss $\mathcal{L}_{\text{aux}}$. The auxiliary semantic loss is weighted by $\lambda_{\text{aux}} = 0.4$, while the primary segmentation loss is weighted by $\lambda_{\text{seg}} = 1.0$.

The use of a larger coefficient for $\mathcal{L}_{\text{bnd}}$ is motivated by the sparsity of boundary pixels. Since boundary pixels occupy only a small proportion of an image, a small boundary loss coefficient may cause the boundary supervision signal to be overwhelmed by dominant non-boundary regions. Therefore, $\lambda_{\text{bnd}} = 20.0$ is used to preserve effective boundary supervision. This setting does not mean that the auxiliary semantic loss or the boundary supervision term replace the primary segmentation objective. The numerical behavior and sensitivity of this coefficient are further analyzed in the experimental section.

3.7. Datasets

Experiments are conducted on two widely used urban road-scene semantic segmentation benchmarks, CamVid and Cityscapes [24,25]. To ensure fair comparison and reproducibility, fixed class definitions, data splits, and evaluation protocols are adopted, and the training and testing procedures are kept consistent within each controlled experimental group.

3.7.1. CamVid

CamVid is a classic road-scene semantic segmentation dataset that is commonly used to validate and compare lightweight real-time segmentation models [24]. For CamVid-11 (11 classes), the split definition of 367 training images, 101 validation images, and 233 test images at a resolution of 960×720 is followed [24,26].

To avoid evaluation bias caused by inconsistent label palettes or class-index mappings, the label palette and class mapping are verified and unified, ensuring that the same 11-class definition is used in both training and evaluation. All CamVid training, validation, and testing in the experiments strictly follow the same list files and dataset root configuration, which helps ensure consistent settings for the ablation studies and the reported ERSGNet results.

3.7.2. Cityscapes

Cityscapes is a large-scale urban street-scene benchmark covering diverse cities, seasons, illumination conditions, and traffic participants, which presents high scene complexity and poses substantial generalization demands [25]. The standard 19-class training label set and the official split, including 2975 training images and 500 validation images, are adopted for training and evaluation. Cityscapes is evaluated on the official test set [25].

From an implementation perspective, Cityscapes provides two annotation encodings, labelIds and labelTrainIds. To ensure consistency between training labels and the evaluation protocol, annotations are converted to `labelTrainIds`, and the corresponding training and validation list files are generated accordingly. This processing helps avoid label inconsistency caused by duplicated or incorrect mappings, facilitates a consistent training/evaluation pipeline, and ensures compatibility with the official evaluation protocol on large-scale road scenes.

3.8. Experimental Settings

All experiments are conducted on a Windows workstation with an Anaconda environment. PyTorch 2.5.1 with Compute Unified Device Architecture (CUDA) 12.1 is used, and training and evaluation are performed on a single NVIDIA GeForce RTX 4060 Laptop graphics processing unit (GPU) with 8 GB memory. All runs are specified by configuration files together with command-line arguments; command-line overrides take precedence as the effective settings to facilitate reproducibility. Unless otherwise stated, the loss weights are set to $\lambda_{\text{seg}} = 1.0$, $\lambda_{\text{aux}} = 0.4$, $\lambda_{\text{bnd}} = 20.0$, and $\lambda_{\text{ba}} = 1.0$. Here, λ_{bnd} denotes the coefficient of the BCE loss rather than the weight of the auxiliary semantic loss. During training, the checkpoint with the best validation performance (`best`) is saved, and the results are reported on the designated evaluation split. For CamVid-11, the validation split is used for model selection, convergence monitoring, and hyperparameter sensitivity analysis, while the final performance is reported on the held-out test split. The test split is not used for hyperparameter tuning. For Cityscapes, the final segmentation accuracy is reported on the official test split.

For CamVid-11, the split definition of 367 training images, 101 validation images, and 233 test images at a resolution of 960×720 is followed [24,26]. The training split is used for model training, and the validation split is not used as the final test split. Training is conducted for 50 epochs with a batch size of 4. Both multi-scale training and random horizontal flipping are enabled. For the ablation and sensitivity analyses, the corresponding controlled training schedules are specified in their own subsections.

For Cityscapes, the 19-class trainId setting is used, following the standard split of 2975 training images and 500 validation images, with the official test split used for final evaluation. Training is conducted for 50 epochs with a batch size of 2. Multi-scale training and random horizontal flipping are disabled to keep the Cityscapes training setting simple and consistent under the single-GPU setting. To improve stability under small-batch training, Group Normalization (GroupNorm) is used in the ERSG-Fuse module. Synchronized batch normalization is not adopted because all experiments are conducted on a single GPU. For CamVid-11, the initial learning rate is set to 0.005. For Cityscapes, the initial learning rate is set to 0.01. For both datasets, stochastic gradient descent (SGD) is used as the optimizer with momentum of 0.9 and weight decay of 0.0005. Online Hard Example Mining (OHEM) is used for semantic supervision with a threshold of 0.9 and a kept-pixel number of 131,072. Mean Intersection-over-Union (mIoU) is used as the primary accuracy metric.

For frames per second (FPS) evaluation, all locally measured models are tested under the same inference protocol on the RTX 4060 Laptop GPU. The model is set to evaluation mode, and inference is performed under `torch.no_grad()` with single-precision floating-

point (FP32) input tensors and batch size 1. CUDA synchronization is used for timing. Unless otherwise specified, 50 warm-up iterations and 200 timed iterations are used, and the average FPS over repeated runs is reported. The input resolution is 960 × 720 for CamVid and 2048 × 1024 for Cityscapes. For the representative competing models re-measured in this work, the same hardware and inference protocol are used.

For computational complexity, THOP (PyTorch-OpCounter) is used to profile multiply–accumulate operations (MACs) with a single forward pass. The model is set to evaluation mode, and profiling is performed under `torch.no_grad()`, so backward propagation is not included. The dummy input is an FP32 tensor with batch size 1, i.e., `(1, 3, H, W)`. Giga floating-point operations (GFLOPs) are reported as 2× giga MACs (GMACs), following the convention that one MAC corresponds to two floating-point operations. Since no custom counting rules are used, the results follow the default THOP profiling rules; unsupported operations are not manually added. Thus, the reported GFLOPs should be interpreted as a static operation-count estimate rather than a hardware-dependent runtime measurement. For consistency, all variants of the proposed model are profiled using the same script and counting protocol. Specifically, the input resolution is 960 × 720 for CamVid and 2048 × 1024 for Cityscapes, resulting in 35.116 GFLOPs and 106.137 GFLOPs for the proposed model, respectively.

4. Results

4.1. Sensitivity Analysis of the Boundary Loss Coefficient

To further examine the influence of boundary supervision strength, a sensitivity analysis of the boundary BCE loss coefficient λ_{bnd} is conducted on the CamVid validation split. In this experiment, λ_{bnd} is varied among 5, 10, 20, 30, and 40, while all other training settings are kept unchanged. Since this experiment aims to evaluate the relative sensitivity of the boundary loss coefficient rather than to re-estimate the final benchmark performance, a 10-epoch training setting is adopted.

As shown in Figure 5, the model obtains final validation mIoU values of 80.06%, 79.84%, 80.09%, 79.15%, and 78.36% when λ_{bnd} is set to 5, 10, 20, 30, and 40, respectively. The performance remains relatively stable when λ_{bnd} is within the range of 5 to 20, with $\lambda_{\text{bnd}} = 20$ achieving the highest final validation mIoU among the tested settings. However, further increasing the coefficient to 30 or 40 leads to a noticeable performance drop, suggesting that over-emphasizing boundary supervision may disturb the balance between boundary learning and semantic segmentation. Therefore, $\lambda_{\text{bnd}} = 20$ is adopted as a balanced setting in the experiments.

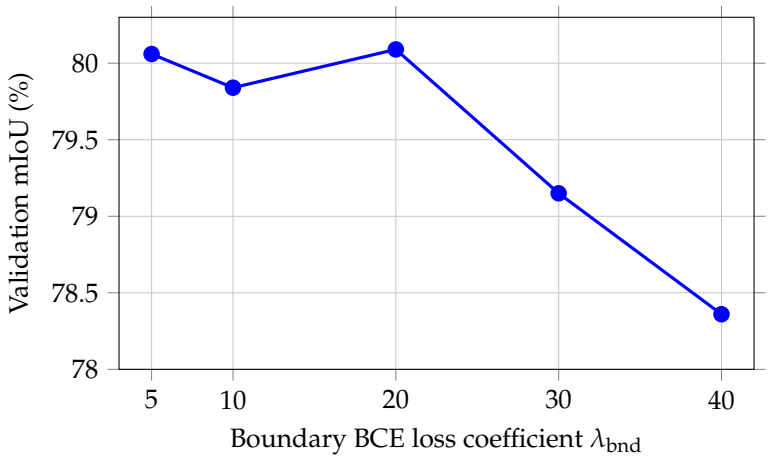

Figure 5. Sensitivity analysis of the boundary binary cross-entropy (BCE) loss coefficient λ_{bnd} on the CamVid validation split. mIoU denotes mean Intersection-over-Union.

4.2. Ablation Experiments

To validate the effectiveness of the proposed components, ablation experiments are conducted on CamVid-11, focusing on three key modules: SGI, PCP, and ERSG-Fuse. All ablations follow the same data split and training protocol. The mIoU is reported using the checkpoint with the best validation performance, and the test set is used for evaluation to support fair and reproducible comparisons. Inference speed is measured on the same hardware with the same timing script, and the reported FPS is the average of three repeated runs.

4.2.1. Ablation Setup and Variant Definitions

The full model is taken as the baseline (E0), and four controlled variants are constructed by either removing a key module or replacing ERSG-Fuse with a controlled fusion baseline. All variants follow the principle of minimal modification and keep the network interfaces unchanged.

- E0: Full model with SGI + PCP + ERSG-Fuse enabled.
- E1: w/o SGI. SGI is removed by skipping the SGI-related paths in the forward pass, while all other branches and prediction heads are kept unchanged. This variant evaluates the contribution of pixel-wise gated interaction.
- E2: w/o PCP. PCP is removed and replaced with a lightweight 1×1 projection module, while the output channel dimension and the interfaces to subsequent fusion and prediction heads are kept unchanged. This variant measures the contribution of multi-scale context aggregation.
- F0: ERSG-Fuse $\rightarrow$ Add. ERSG-Fuse is replaced with a non-learnable fusion baseline (Add): after spatial alignment and the same feature projections, element-wise addition ($F_d + F_c$) is used as the fused output, while the subsequent structure is kept identical to the ERSG-Fuse setting. This variant is designed to evaluate the importance of edge-guided fusion compared with naive summation.
- F1: ERSG-Fuse $\rightarrow$ Concat-1×1 convolution. ERSG-Fuse is replaced with a learnable concatenation-based fusion baseline: after spatial alignment and the same feature projections, the aligned features are concatenated along the channel dimension and then compressed by a 1×1 convolution to match the original fusion output dimension. The subsequent prediction head remains unchanged. This variant provides a stronger and commonly used fusion baseline than simple addition, and is used to examine whether ERSG-Fuse provides additional benefits beyond learnable channel mixing.

4.2.2. Ablation Results and Analysis

Table 1 reports the ablation results on the CamVid-11 test set. Overall, removing SGI or the PCP leads to a clear accuracy drop, while replacing ERSG-Fuse with different fusion baselines results in different accuracy–speed trade-offs. To further examine the accuracy–efficiency behavior of the proposed fusion design, an additional comparison with a learnable Concat-1×1 convolution fusion baseline is conducted.

Table 1. Ablation results on CamVid-11. FPS denotes frames per second.

Experiment ID	SGI	PCP	Fusion	mIoU (%)	FPS
E0	✓	✓	ERSG-Fuse	73.33	166.27
E1	×	✓	ERSG-Fuse	71.30	176.79
E2	✓	×	ERSG-Fuse	71.86	176.82
F0	✓	✓	Add	70.52	169.40
F1	✓	✓	Concat-1×1	73.30	148.21

Overall, the ablation results show that both SGI and the PCP contribute to the accuracy of the full model, while introducing a noticeable runtime cost of approximately 10 FPS. Removing SGI reduces mIoU from 73.33% to 71.30%, suggesting that pixel-wise guided interaction contributes to branch interaction and fine-grained representation, while the speed increases from 166.27 to 176.79 FPS. Similarly, removing the PCP decreases mIoU from 73.33% to 71.86%, suggesting that multi-scale context aggregation contributes to semantic discriminability, with FPS increasing from 166.27 to 176.82.

For the fusion strategy, replacing ERSG-Fuse with naive addition ($F_d + F_c$) reduces the mIoU to 70.52%, indicating that direct summation is insufficient to effectively integrate features from different branches. The Concat-1 $\times$ 1 baseline introduces learnable channel mixing by concatenating the aligned features and compressing them with a 1 $\times$ 1 convolution, and it achieves a competitive mIoU of 73.30%. This result shows that learnable fusion is substantially more effective than simple addition in this setting. However, the performance gap between ERSG-Fuse and Concat-1 $\times$ 1 is small in the single-run result. Therefore, we do not interpret the 0.03 percentage-point difference as a reliable accuracy advantage.

To further examine this small difference, we additionally conduct a multi-seed stability analysis between ERSG-Fuse and the Concat-1 $\times$ 1 baseline. The results are reported in Table 2. Across three random seeds, ERSG-Fuse obtains 73.61 $\pm$ 0.32% mIoU, while Concat-1 $\times$ 1 obtains 73.24 $\pm$ 0.23% mIoU. Although ERSG-Fuse shows a higher average mIoU, the result is better interpreted as comparable segmentation accuracy rather than a statistically conclusive accuracy advantage. The more consistent advantage of ERSG-Fuse lies in inference efficiency. ERSG-Fuse reaches 166.27 FPS, compared with 148.21 FPS for Concat-1 $\times$ 1.

Table 2. Multi-seed stability analysis between ERSG-Fuse and the Concat-1 $\times$ 1 baseline. Values are reported as mean $\pm$ standard deviation (SD).

Method	Seed 1	Seed 2	Seed 3	Mean $\pm$ SD
Concat-1 $\times$ 1	73.30	73.43	72.98	73.24 $\pm$ 0.23
ERSG-Fuse	73.33	73.95	73.54	73.61 $\pm$ 0.32

This accuracy–efficiency behavior can be further understood from the structural differences between the two fusion strategies. Concat-1 $\times$ 1 performs dense channel mixing after concatenating multiple branch features, which increases the channel dimension before compression. In contrast, ERSG-Fuse avoids direct high-dimensional concatenation and assigns different roles to its internal components. First, the edge-guided gate generates a single-channel spatial modulation map from the edge feature, allowing the fusion weights to vary across spatial locations without performing dense channel mixing over concatenated features. Second, the residual gated interpolation $Y_0 = \tilde{F}_d + G \odot (\tilde{F}_c - \tilde{F}_d)$ formulates fusion as a position-adaptive transition from detail features to contextual features, with the detail branch preserved as the reference representation. Third, the learnable boundary residual injection $Y_1 = Y_0 + \beta \tilde{F}_e$ introduces edge information as a controlled residual correction rather than as a full concatenated branch. In addition, edge-branch supervision provides explicit training signals for the edge representation, making the edge feature more suitable for gate generation and boundary residual correction. Therefore, these components should be interpreted as complementary structural designs for efficient boundary-aware fusion. Overall, ERSG-Fuse provides an efficient alternative to concatenation-based channel mixing, achieving comparable segmentation accuracy with higher inference speed.

To further examine the performance on detail-sensitive categories, class-wise Intersection-over-Union (IoU) results are reported for CamVid-11 categories that contain thin structures, small object regions, or complex object boundaries. Specifically, Pole, SignSymbol, Fence,

Pedestrian, and Bicyclist are selected for this analysis, since these categories are more sensitive to local detail preservation and boundary localization.

As shown in Figure 6, the full model E0 achieves favorable class-wise results on the selected detail-sensitive categories. Compared with the additive fusion baseline F0, E0 improves Pole, SignSymbol, Fence, Pedestrian, and Bicyclist by 3.06, 3.92, 2.75, 2.42, and 9.72 percentage points, respectively, indicating that ERSG-Fuse is more effective than direct summation for these detail-sensitive categories. Compared with E1 and E2, E0 achieves higher IoU on Pole, SignSymbol, Pedestrian, and Bicyclist, showing that SGI and the PCP contribute to the representation of small or structurally complex categories. Compared with F1, E0 achieves higher IoU on SignSymbol and Bicyclist, and also obtains a slightly higher overall mIoU and higher inference speed as reported in Table 1. These results suggest that ERSG-Fuse provides an effective and efficient fusion strategy for detail-sensitive semantic segmentation.

Although ERSGNet improves the representation of several detail-sensitive categories, very small objects and slim structures, such as poles and fences, remain challenging. A possible direction for future work is to further enhance the interaction between boundary cues and high-resolution detail features, so that thin structures and small object regions may receive more focused feature refinement. In addition, more targeted supervision for rare or narrow categories may also help improve their IoU.

To further provide qualitative evidence, Figure 7 presents the prediction results of different ablation variants on representative CamVid samples.

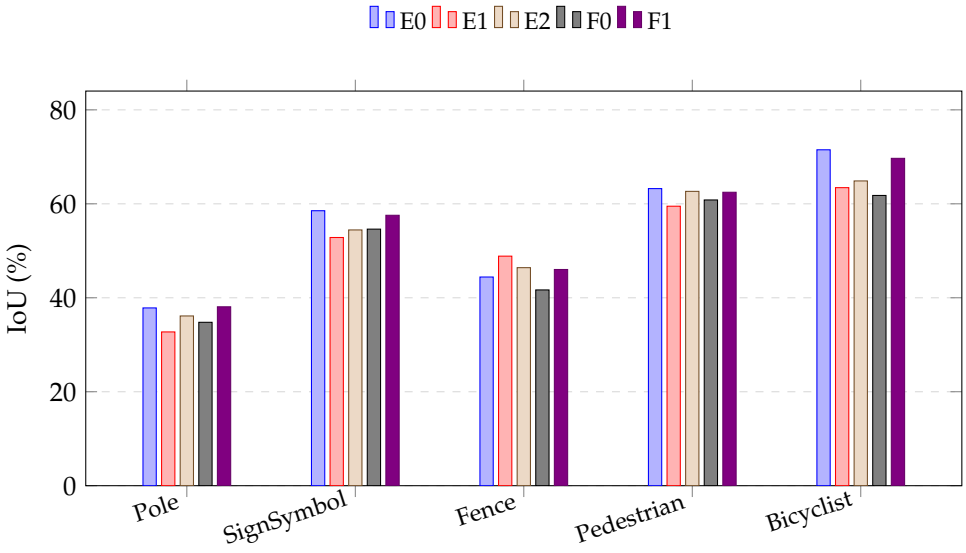

Figure 6. Class-wise Intersection-over-Union (IoU) comparison on detail-sensitive CamVid-11 categories.

As shown in Figure 7a, compared with the ablation variants, the full model produces clearer object boundaries and less merging in regions containing vehicles and traffic signs. As shown in Figure 7b, compared with the full model, the variant without SGI shows more noticeable merging between the two pedestrians, and the road intersection becomes less clearly separated. As shown in Figure 7c, all ablation variants exhibit varying degrees of degradation in the left-side sidewalk region compared with the full model, while the bicyclist also undergoes different levels of shape distortion.

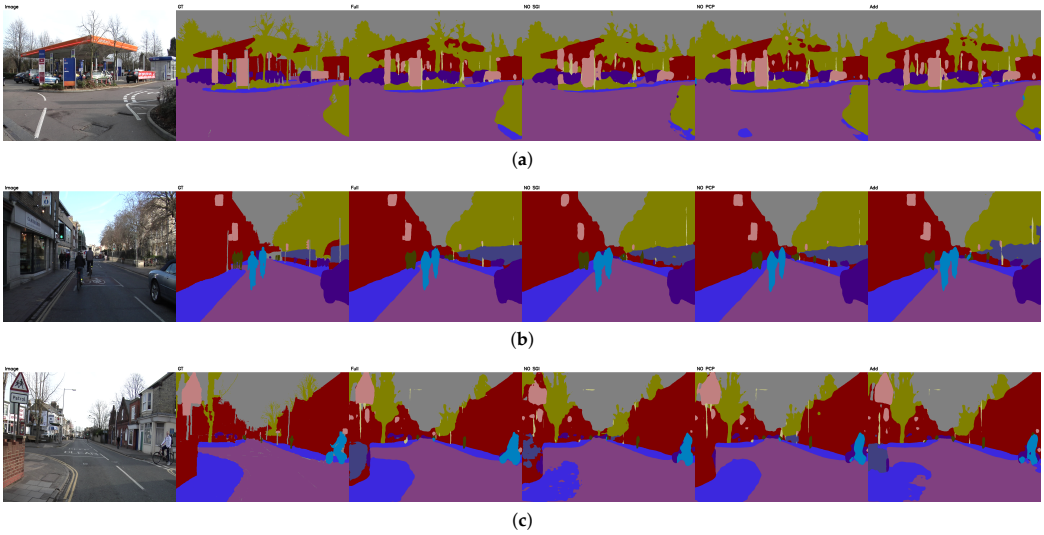

Figure 7. Qualitative comparison of ERSGNet and its ablation variants on three representative images from the CamVid test set. In each subfigure, the columns from left to right show the input image, ground truth (GT), the full ERSGNet, ERSGNet without SGI, ERSGNet without PCP, and the additive-fusion (Add) variant: (**a**) a road-intersection scene near a service station; (**b**) an urban street scene containing cyclists, pedestrians, and vehicles; (**c**) a residential street scene with roadside buildings, vehicles, and a cyclist.

4.3. Input-Resolution Robustness Analysis

To examine the robustness of ERSGNet to test-time input-resolution variation, the same trained checkpoint is evaluated on the CamVid-11 test split under different forced input resolutions without retraining. For each setting, each input image is resized to the target resolution before inference, and the prediction is resized back to the original label resolution for mIoU calculation.

As shown in Figure 8, the mIoU gradually decreases as the input resolution is reduced from 960×720 to 768×576 and 640×480. This trend is expected because lower input resolutions reduce local boundary details and small-object cues. Nevertheless, ERSGNet remains evaluable under all tested resolutions and does not exhibit abrupt performance degradation. The resolution-robustness results indicate that the proposed model is not restricted to a single fixed input size under the tested resolution settings, while the standard 960×720 resolution remains the most effective setting for CamVid-11 evaluation.

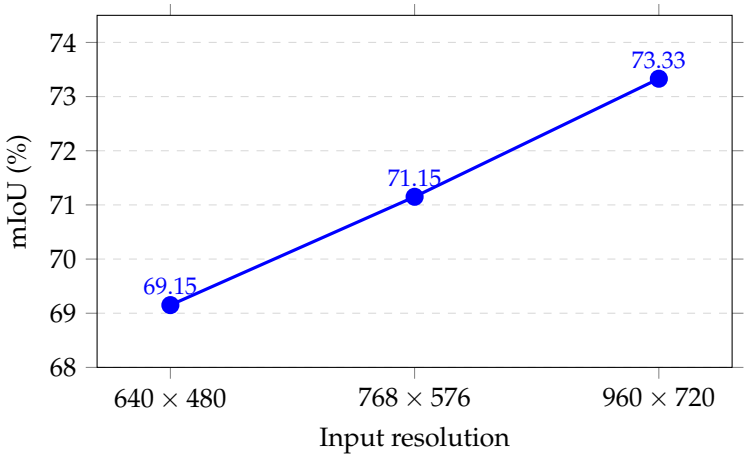

Figure 8. Input-resolution robustness analysis on CamVid-11.

Taken together, ERSG-Fuse provides a favorable accuracy–speed trade-off among the tested components, whereas SGI and the PCP further contribute complementary improvements at a moderate speed cost. The IoU results of several detail-sensitive categories further show that the performance gain of the proposed model is not limited to the overall segmentation accuracy, but is also reflected in categories that rely more heavily on fine-grained structures and local boundary cues. These results are consistent with the motivation that the proposed fusion strategy helps balance contextual semantics and local details during feature interaction.

4.4. Comparison Experiments

4.4.1. CamVid Comparison and Discussion

Table 3 summarizes the quantitative comparison on CamVid-11 in terms of mIoU, FPS, and GFLOPs. It provides an overall reference for the accuracy–efficiency trade-off among representative real-time segmentation models.

Table 3. CamVid-11 comparison with representative real-time segmentation models. GFLOPs denotes giga floating-point operations, and GPU denotes graphics processing unit.

Model	mIoU (%)	FPS	GFLOPs	GPU
ERSGNet	73.33	166.27	35.116	RTX 4060
BiSeNetV2 [8]	72.40	84.64	21.236	RTX 4060
EBUNet [21]	72.20	65.05	48.595	RTX 4060
LAANet [22]	72.19	47.32	24.955	RTX 4060
LETNet [16]	70.50	24.17	38.722	RTX 4060
FBSNet [23]	68.90	13.58	62.387	RTX 4060

On CamVid-11 [24], the ERSGNet achieves 73.33% mIoU. Specifically, the ERSGNet outperforms the BiSeNetV2, EBUNet, LAANet, LETNet, and FBSNet by 0.93, 1.13, 1.14, 2.83, and 4.43 percentage points, respectively. These results indicate that the ERSGNet achieves competitive segmentation accuracy under the CamVid-11 evaluation protocol.

The FPS and GFLOPs values in Table 3 are measured under the same input resolution of 960×720, the same RTX 4060 Laptop GPU, and the same PyTorch inference setting. Under this unified setting, the ERSGNet achieves 166.27 FPS with 35.116 GFLOPs. Its measured FPS is higher than those of all compared models in the table. In terms of computational complexity, the ERSGNet requires more GFLOPs than the BiSeNetV2 and the LAANet, but fewer GFLOPs than the EBUNet, LETNet, and FBSNet. Therefore, the ERSGNet does not achieve its performance by simply minimizing theoretical computation. Instead, it provides a favorable balance between segmentation accuracy, practical inference speed, and moderate computational complexity.

From a structural perspective, the CamVid-11 results may be associated with the branch-specialized and edge-guided design of the ERSGNet. Compared with the dual-path design of the BiSeNetV2, the ERSGNet further introduces an edge branch to provide boundary-related guidance for the final fusion stage. This additional boundary cue may help reduce the interference between contextual semantics and local details during feature fusion. Compared with lightweight efficient models such as the EBUNet and LAANet, the ERSGNet does not only rely on compact convolutional design or attention-guided aggregation. Instead, SGI selectively injects contextual semantics into the detail branch, while the PCP strengthens multi-scale contextual representation before final fusion. Compared with the LETNet and FBSNet, which strengthen semantic modeling or bilateral feature aggregation, the ERSGNet further uses ERSG-Fuse to perform edge-guided residual spatial gating fusion. This design may help preserve local structural cues during feature integration, which is consistent with the higher mIoU observed in Table 3.

The runtime result also suggests that the additional branch design does not prevent the ERSGNet from maintaining practical inference speed under the tested setting. Although the ERSGNet has more GFLOPs than the BiSeNetV2 and LAANet, it achieves higher FPS on the same hardware and input resolution. This may be related to its shared low-level feature extraction, relatively shallow edge branch, SGI applied only at two intermediate detail-branch stages, channel-compressed parallel context aggregation in the PCP, and lightweight single-channel spatial gates in ERSG-Fuse. These designs are intended to control the extra cost introduced by detail, context, and edge modeling.

Overall, the CamVid-11 comparison shows that the ERSGNet achieves the best mIoU and the highest measured FPS among the compared models under the unified evaluation setting. Since its GFLOPs are not the lowest, the advantage of the ERSGNet should be understood as an accuracy–efficiency trade-off associated with its edge-guided feature fusion design, rather than as a minimum-complexity design.

4.4.2. Cityscapes Comparison and Discussion

Table 4 reports the quantitative comparison on Cityscapes, including mIoU, FPS, and GFLOPs. It provides an overall reference for the accuracy–efficiency trade-off under the standard urban street-scene evaluation protocol.

Table 4. Cityscapes-19 comparison with representative real-time segmentation models.

Model	mIoU (%)	FPS	GFLOPs	GPU
ERSGNet	69.21	53.93	106.137	RTX 4060
CGNet [18]	64.80	23.17	57.162	RTX 4060
BiSeNet [27]	68.40	25.42	238.967	RTX 4060
Fast-SCNN [17]	68.00	85.48	14.289	RTX 4060
EDANet [19]	67.32	23.55	71.704	RTX 4060
ESPNetv2 [20]	66.15	46.60	39.624	RTX 4060

On Cityscapes-19 [25], all FPS and GFLOPs values in Table 4 are measured under the same input resolution of 2048 × 1024, the same RTX 4060 Laptop GPU, and the same PyTorch inference setting. The official test-set mIoU returned by the evaluation server is used as the primary metric. The ERSGNet achieves 69.21% mIoU on the test set. Specifically, the ERSGNet outperforms the CGNet, BiSeNet, Fast-SCNN, EDANet, and ESPNetv2 by 4.41, 0.81, 1.21, 1.89, and 3.06 percentage points, respectively [17–20,27]. These results indicate that the ERSGNet achieves competitive segmentation accuracy on the more complex Cityscapes benchmark.

Under the same 2048 × 1024 input setting, the ERSGNet achieves 53.93 FPS with 106.137 GFLOPs. Compared with the BiSeNet, the ERSGNet obtains higher mIoU, higher FPS, and lower GFLOPs, indicating that the ERSGNet achieves better accuracy and runtime performance than this dual-path baseline under the unified evaluation setting. Compared with the CGNet, EDANet, and ESPNetv2, the ERSGNet achieves higher mIoU and higher FPS, although it requires more GFLOPs. This suggests that although the ERSGNet requires more GFLOPs than these compact models, it still maintains practical inference speed under the unified runtime setting. This may be related to the relatively shallow edge branch, SGI applied only at two intermediate detail-branch stages, channel-compressed parallel context aggregation in the PCP, and the lightweight ERSG-Fuse module. Compared with Fast-SCNN, the ERSGNet improves mIoU by 1.21 percentage points, but its FPS is lower and its GFLOPs are higher. This indicates that extremely compact learning-to-downsample designs still retain advantages in raw speed and theoretical computation, while the ERSGNet focuses more on improving segmentation accuracy through richer feature coordination.

From a structural perspective, the above results are consistent with the design motivation of the ERSGNet. The CGNet, EDANet, ESPNetv2, and Fast-SCNN mainly improve efficiency through compact convolutional structures, context-guided blocks, efficient dense modules, or early downsampling. These designs reduce computation, but high-resolution detail and boundary cues may be weakened during feature extraction. The BiSeNet improves real-time segmentation through a dual-path structure that separates spatial details and semantic context. The ERSGNet further introduces an edge branch and uses SGI, the PCP, and ERSG-Fuse to coordinate detail features, contextual semantics, and boundary-related responses. The higher mIoU in Table 4 is consistent with the expected benefit of this additional edge-guided feature coordination under the full-resolution Cityscapes setting.

Overall, the Cityscapes-19 comparison shows that the ERSGNet provides a favorable accuracy–efficiency trade-off under the unified 2048×1024 evaluation setting. Its main advantage lies in achieving the highest mIoU among the compared models while maintaining practical real-time inference speed. Meanwhile, ERSGNet is not the most lightweight model in terms of GFLOPs, so its benefit should be interpreted as accuracy-oriented efficient fusion rather than a minimum-complexity design.

5. Conclusions

This work targets real-time semantic segmentation for autonomous driving and general real-time vision applications, with a particular focus on boundary blur and small-structure loss in lightweight models. The ERSGNet is proposed together with its key fusion module, ERSG-Fuse. The ERSGNet adopts a tri-branch design with dedicated detail, context, and edge branches: SGI enables on-demand semantic injection into the detail branch, while the PCP performs parallel multi-scale aggregation at the end of the context branch to enhance multi-scale contextual representation. Prior to prediction, ERSG-Fuse generates a single-channel spatial gate conditioned on edge features to realize location-sensitive fusion, and further combines residual gated fusion with learnable boundary residual injection. This design is intended to improve the selectivity of feature fusion and boundary delineation quality while keeping the additional computational overhead under control.

Experiments on CamVid and Cityscapes show that the ERSGNet achieves competitive performance in both segmentation accuracy and practical inference efficiency under the tested settings (CamVid: 73.33% mIoU at 166.27 FPS; Cityscapes: 69.21% mIoU at 53.93 FPS). For the comparison experiments, FPS and GFLOPs are measured under the same input resolution, hardware platform, and PyTorch inference setting within each dataset, which provides a direct reference for evaluating the accuracy–efficiency trade-off. Ablation studies further support the complementary roles of SGI, the PCP, and ERSG-Fuse. Class-wise IoU analysis on detail-sensitive categories, unified FPS and GFLOPs comparisons with representative baselines, loss-weight sensitivity analysis, and input-resolution robustness analysis provide additional evidence for the effectiveness of the proposed design under the tested settings. Overall, the results suggest that the ERSGNet provides an effective edge-guided fusion strategy for improving boundary-related and fine-grained representation, leading to competitive segmentation accuracy and a favorable accuracy–efficiency trade-off.

Future work will proceed in three directions. First, stronger lightweight backbones will be explored to further enhance semantic representations without significantly increasing computational cost. Second, more refined boundary supervision and structural constraints will be investigated to improve robustness under challenging conditions. Third, the inference pipeline will be further optimized for deployment, aiming to achieve a better accuracy–speed trade-off.

https://doi.org/10.3390/machines14070793

Author Contributions: Conceptualization, Z.W. and L.J.; methodology, Z.W. and L.J.; software, Z.W.; validation, Z.W.; formal analysis, Z.W.; investigation, Z.W. and T.W.; resources, L.J.; data curation, Z.W. and T.W.; writing—original draft preparation, Z.W. and L.J.; writing—review and editing, Z.W.; visualization, Z.W.; supervision, L.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under Grants 62403112 and 52527810, the China Postdoctoral Science Foundation under Grant 2025T180470, the Fundamental Research Funds for the Central Universities under Grant N2404025, and the Natural Science Foundation of Liaoning Province under Grant 2024-BSBA-19.

Data Availability Statement: The datasets used in this study are publicly available. Cityscapes is available from the official website upon registration and acceptance of the terms of use, and CamVid is publicly available for research purposes. The code and trained models are available from the authors upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ulku, I.; Akagündüz, E. A Survey on Deep Learning-based Architectures for Semantic Segmentation on 2D Images. *Appl. Artif. Intell.* **2022**, *36*, 2032924. [CrossRef]
2. Kwon, Y.; Kim, W.; Kim, H. HARD: Hardware-Aware Lightweight Real-Time Semantic Segmentation Model Deployable from Edge to GPU. In *Computer Vision—ACCV 2024*; Cho, M., Laptev, I., Tran, D., Yao, A., Zha, H., Eds.; Lecture Notes in Computer Science; Springer Nature: Singapore, 2025; Volume 15478, pp. 252–269. [CrossRef]
3. Elhassan, M.A.; Zhou, C.; Khan, A.; Benabid, A.; Adam, A.B.; Mehmood, A.; Wambugu, N. Real-time semantic segmentation for autonomous driving: A review of CNNs, Transformers, and Beyond. *J. King Saud Univ.-Comput. Inf. Sci.* **2024**, *36*, 102226. [CrossRef]
4. Zhang, Y.; Zhang, X.; Miao, D.; Yu, H. Real-Time Semantic Segmentation of Road Scenes via Hybrid Dilated Grouping Network. *IJNDI* **2025**, *4*, 100006. [CrossRef]
5. Lisauskas, B.; Maskeliunas, R. Efficient Transformer-Based Road Scene Segmentation Approach with Attention-Guided Decoding for Memory-Constrained Systems. *Machines* **2025**, *13*, 466. [CrossRef]
6. Park, M.H.; Cho, J.H.; Kim, Y.T. CNN Model with Multilayer ASPP and Two-Step Cross-Stage for Semantic Segmentation. *Machines* **2023**, *11*, 126. [CrossRef]
7. Chen, C.; Liu, G. DBBGNet: A Dual-Branch Network with Boundary Information Guidance for Real-Time Semantic Segmentation. *Vis. Comput.* **2026**, *42*, 297. [CrossRef]
8. Yu, C.; Gao, C.; Wang, J.; Yu, G.; Shen, C.; Sang, N. BiSeNet V2: Bilateral Network with Guided Aggregation for Real-Time Semantic Segmentation. *Int. J. Comput. Vis.* **2021**, *129*, 3051–3068. [CrossRef]
9. Pan, H.; Hong, Y.; Sun, W.; Jia, Y. Deep Dual-Resolution Networks for Real-Time and Accurate Semantic Segmentation of Traffic Scenes. *IEEE Trans. Intell. Transport. Syst.* **2023**, *24*, 3448–3460. [CrossRef]
10. 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* **2022**, arXiv:2204.02681. [CrossRef]
11. Elhassan, M.A.M.; Yang, C.; Huang, C.; Munea, T.L.; Hong, X.; Adam, A.B.M.; Benabid, A. S^2-FPN: Scale-aware Strip Attention Guided Feature Pyramid Network for Real-time Semantic Segmentation. *arXiv* **2022**, arXiv:2206.07298. [CrossRef]
12. Cărunta, C.; Cărunta, A.; Mizeranschi, A.E.; Popa, C.A. Real-time semantic segmentation of driving scenes via effective attention-based information fusion and hybrid encoder. *Sci. Rep.* **2026**, *16*, 2419. [CrossRef] [PubMed]
13. Zhao, S.; Fu, W.; Zhang, F.; Huo, Z.; Qiao, Y. AEAFFNet: Enhancing Real-Time Semantic Segmentation through Attention-Enhanced Adaptive Feature Fusion. *J. Supercomput.* **2026**, *82*, 53. [CrossRef]
14. Xie, E.; Wang, W.; Yu, Z.; Anandkumar, A.; Alvarez, J.M.; Luo, P. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–14 December 2021; Volume 34, pp. 12077–12090.
15. Deepak, G.D.; Hiremath, P.; Bhat, S.K. Taguchi-optimized DeepLabV3+ for semantic segmentation in autonomous driving applications. *Ain Shams Eng. J.* **2026**, *17*, 103985. [CrossRef]
16. Xu, G.; Li, J.; Gao, G.; Lu, H.; Yang, J.; Yue, D. Lightweight Real-Time Semantic Segmentation Network with Efficient Transformer and CNN. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 15897–15906. [CrossRef]
17. Poudel, R.P.K.; Liwicki, S.; Cipolla, R. Fast-SCNN: Fast Semantic Segmentation Network. In *Proceedings of the British Machine Vision Conference, Cardiff, UK, 9–12 September 2019*; BMVA Press: Durham, UK, 2019; p. 289.

18. Wu, T.; Tang, S.; Zhang, R.; Cao, J.; Zhang, Y. CGNet: A Light-Weight Context Guided Network for Semantic Segmentation. *IEEE Trans. Image Process.* **2021**, *30*, 1169–1179. [CrossRef] [PubMed]
19. Lo, S.Y.; Hang, H.M.; Chan, S.W.; Lin, J.J. Efficient Dense Modules of Asymmetric Convolution for Real-Time Semantic Segmentation. In *Proceedings of the ACM International Conference on Multimedia in Asia*; Association for Computing Machinery: New York, NY, USA, 2019. [CrossRef]
20. Mehta, S.; Rastegari, M.; Shapiro, L.G.; Hajishirzi, H. ESPNetv2: A Light-Weight, Power Efficient, and General Purpose Convolutional Neural Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*; IEEE: Piscataway, NJ, USA, 2019; pp. 9182–9192. [CrossRef]
21. Shen, S.; Zhai, Z.; Yu, G.; Yan, Y.; Dai, W. EBUNet: A Fast and Accurate Semantic Segmentation Network with Lightweight Efficient Bottleneck Unit. *Complex Intell. Syst.* **2023**, *9*, 5975–5990. [CrossRef]
22. Zhang, X.; Du, B.; Wu, Z.; Wan, T. LAANet: Lightweight Attention-Guided Asymmetric Network for Real-Time Semantic Segmentation. *Neural Comput. Appl.* **2022**, *34*, 3573–3587. [CrossRef]
23. Gao, G.; Xu, G.; Li, J.; Yu, Y.; Lu, H.; Yang, J. FBSNet: A Fast Bilateral Symmetrical Network for Real-Time Semantic Segmentation. *IEEE Trans. Multimed.* **2023**, *25*, 3273–3283. [CrossRef]
24. Brostow, G.J.; Fauqueur, J.; Cipolla, R. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognit. Lett.* **2009**, *30*, 88–97. [CrossRef]
25. 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 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 3213–3223. [CrossRef]
26. Tian, S.; Yao, G.; Chen, S. Faster SCDNet: Real-Time Semantic Segmentation Network with Split Connection and Flexible Dilated Convolution. *Sensors* **2023**, *23*, 3112. [CrossRef] [PubMed]
27. Yu, C.; Wang, J.; Peng, C.; Gao, C.; Yu, G.; Sang, N. BiSeNet: Bilateral Segmentation Network for Real-Time Semantic Segmentation. In *Computer Vision—ECCV 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2018; Volume 11217, pp. 334–349. [CrossRef]