CAAI Transactions on Intelligence Technology

WILEY

The Institution of Engineering and Technology

ORIGINAL RESEARCH OPEN ACCESS

TriCrackNet: Trilateral Segmentation Network for Real-Time Crack Segmentation

Haixin Jia[1] | Han Wang[2] | Xiaoqiang Shi[3] | Guoying Zhang[1] | Hanqi Jiang[4] | Jianbo Peng[1]

[1]School of Artificial Intelligence, China University of Mining and Technology-Beijing, Beijing, China | [2]School of Computer Science and Engineering, Chongqing University of Technology, Chongqing, China | [3]The Shenyang Institute of Computing Technology, Chinese Academy of Sciences-Shenyang, Shenyang, China | [4]School of Computing, The University of Georgia, Athens, Georgia, USA

Correspondence: Han Wang (3388314691@stu.cqut.edu.cn)

Received: 7 July 2025 | **Revised:** 10 March 2026 | **Accepted:** 12 March 2026

Keywords: attention mechanism | feature fusion | multi-scale features | real-time segmentation

ABSTRACT

To achieve high-precision real-time crack segmentation, we propose TriCrackNet, an efficient network based on a tri-branch collaborative architecture incorporating boundary constraints, semantic parsing, and spatial refinement. In the semantic branch, efficient atrous spatial pyramid pooling (EASPP) is integrated. The EASPP performs feature grouping and efficiently utilises each group's features through hierarchical residual connections. Simultaneously, it employs progressive dilated convolutions with incrementally expanding receptive fields to comprehensively capture multi-scale crack features. Meanwhile, an efficient feature enhancement module (EFEM) was inserted between the semantic branch and the other two branches. This module selectively utilises global features from the semantic branch to provide semantic guidance for the other branches, while enabling self-adaptive enhancement of local features in the target branches based on their own feature characteristics. Finally, the features of the last three branches are efficiently fused through an efficient feature fusion module that performs adaptive spatial weight weighting. Compared with other state-of-the-art methods, TriCrackNet achieves the best trade-off between inference speed and accuracy on CrackForest, Crack500, and DeepCrack datasets.

1 | Introduction

Accurate and timely detection of road surface cracks is helpful for the development of highway maintenance and maintenance strategies. Traditionally, pavement crack detection relied on visual inspection and manual assessment, which are costly, inefficient, and risky. These methods are also subject to subjective bias from the inspectors, leading to missed or false detections [1]. With the rapid advancement of image processing technology and computer vision, many researchers have adopted deep learning-based methods for pavement crack detection [2], achieving promising results. While methods [3–5] focused on accuracy can achieve precise crack segmentation, their slow inference speed may prevent cracks from being detected in real-time during inspections, thereby increasing road safety risks. Therefore, achieving crack segmentation with real-time responsiveness is crucial for efficient highway maintenance [6–8].

Real-time segmentation of a single branch mainly relies on reducing image resolution [9], decreasing the number of feature channels [10], and depth separable convolutions [11]. Although these methods significantly improve inference speed, they lose substantial detail, reducing segmentation accuracy [12]. Recently, the emergence of multi branch efficient networks [13, 14] has made it possible for solutions to balance speed and accuracy. However, due to the diverse sizes and complex textures of crack data [15], these networks are poorly suited for direct

Abbreviations: ANA, anti-nuclear antibodies; APC, antigen-presenting cells; IRF, interferon regulatory factor.

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2026 The Author(s). *CAAI Transactions on Intelligence Technology* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology and Chongqing University of Technology.

application. Although some existing modules [16–24] can enhance the extraction and expression of crack features in the model, their complex structure and redundant calculations greatly reduce the inference speed of the model.

To achieve accurate and efficient crack segmentation, we propose a trilateral segmentation network for real-time crack segmentation-TriCrackNet. TriCrackNet has three branches to extract the boundary, spatial, and semantic features of cracks separately. Among them, the boundary branch adopts the boundary loss function as a constraint to capture the edge texture information of the crack. In the semantic branch, we performed multiple downsampling, increased the number of feature channels, and connected residuals to extract the semantic features of cracks. We inserted our designed efficient atrous spatial pyramid pooling (EASPP) into this branch. It is based on the idea of partial convolution [25], grouping feature maps and inputting features from different groups in a hierarchical manner. It performs receptive field increasing convolution step by step to effectively extract multi-scale features of cracks while avoiding redundant calculations. We maintain the dimensionality of the feature map unchanged in the spatial branch and use simple skip connections during upsampling to efficiently capture the spatial details of cracks.

In addition, to further promote the effective extraction of boundary and spatial features, we propose an efficient feature enhancement module (EFEM). This module only uses global features from the semantic branch as weights to supplement semantic gaps in the other two branches, avoiding redundant calculations. Meanwhile, the other two branches can enhance key spatial details and dilute noise based on feature selection strategies. Finally, we propose an efficient feature fusion module (EFFM) that broadcasts key features from boundary branches to other branches to further enrich their texture details. Simultaneously, the fusion of spatial and semantic information is achieved through simple unified feature learning and matching of different weights.

As shown in Figure 1, compared with other state-of-the-art real-time segmentation schemes, TriCrackNet achieves the best balance between accuracy and speed on the CrackForest [32], Crack500 [33], and DeepCrack [34] datasets. In addition, we conducted sufficient ablation experiments to verify the effectiveness of each module. The main contributions of this paper are summarised as follows:

- We propose a new end-to-end three branch real-time crack segmentation network TriCrackNet, which aims to efficiently and accurately extract crack boundaries, spatial, and semantic features.
- We propose an efficient Spatial Pyramid Pooling (EASPP) that can achieve multi-level context awareness through feature grouping and cascaded dilated convolution strategies, significantly improving feature extraction efficiency while utilising feature information.
- We propose an efficient feature enhancement module (EFEM) that utilises the global features of semantic branch to supplement semantic information for spatial and boundary branches, and design an adaptive feature filtering mechanism to enhance spatial detail features while suppressing noise.
- We propose an efficient feature fusion module (EFFM) to effectively achieve the fusion of boundary, spatial, and semantic features.

2 | Related Work

In this section, we discussed four key areas closely related to our work: real-time semantic segmentation, multi-scale feature extraction, feature enhancement, and feature fusion.

2.1 | Real-Time Semantic Segmentation

To meet the demands of real-time performance and accuracy in segmentation tasks, many researchers have focused on the design of real-time semantic segmentation. Existing real-time semantic segmentation networks are mainly categorised into two types:

Efficient Single Branch Network: Networks like SwiftNet [35] utilise a Res18 [36] encoder and a lightweight decoder architecture to enhance the speed of semantic segmentation. DFANet [37] is based on Xception [38] and uses both sub-networks and

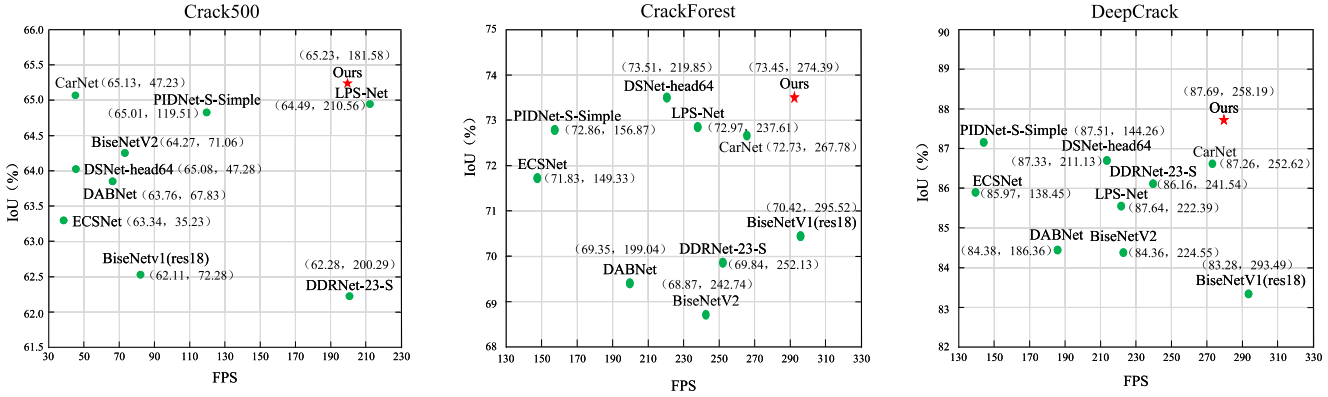


FIGURE 1 | Comparison between TriCrackNet and existing SOTA methods [13, 14, 16, 26–30, 60] on CrackForest, Crack500, and DeepCrack datasets.

sub-stage cascades to discriminate features. ShuffleSeg [39], based on ShuffleNet [40], leverages grouped convolution and channel shuffling strategies to enhance the reliability of feature extraction in lightweight encoders. RHACrackNet [41] utilises a hybrid attention mechanism and residual blocks to construct an extremely lightweight encoder decoder architecture, and achieves efficient real-time segmentation of road cracks while significantly reducing the number of parameters through depthwise separable convolution. However, most of these methods are based on depthwise separable convolutions. Although they have improved some efficiency, their performance on GPU is not as good as traditional convolution [16]. SCTNet [42] achieves high-precision real-time segmentation through dual branch training and single branch prediction. However, the lack of detail feature enhancement mechanisms limits the ability to capture complex texture details. RAFNet [43] achieves efficient segmentation through lightweight encoding and decoding architecture and reparameterised cross resolution fusion, but lacks accuracy in small target segmentation.

Efficient Multi-Branch Network: BiseNet [13] uses a dual-branch architecture to extract spatial and context features of images, achieving feature fusion through channel attention. DDRNet [26] also uses a dual branch architecture, but features are concatenated between the upper and lower branches to achieve feature fusion. RTformer [44] combines a CNN and a simple Transformer block based on the External Attention [45] in a dual-branch encoder to achieve efficient semantic segmentation. Both PIDNet [16] and LPS-Net [27] are real-time segmentation networks with three branches. Although they can effectively extract boundary, spatial, and semantic features, their inference speed is limited by the complex internal modules. BulletNet [46] adopts a unique multi-branch architecture for bullet-head, and integrates cross scale global attention modules and boundary guided fusion mechanisms to achieve efficient feature interaction while effectively balancing fine-grained detail preservation and real-time inference performance of road cracks. DDSNet [47] can efficiently extract image features through a carefully designed dual branch architecture and multiple loss constraints, but lacks enhancement and fusion mechanisms, and there is still room for improvement in final accuracy.

2.2 | Multi-Scale Feature Extraction

Multi-scale feature extraction can integrate information from different receptive fields to help models accurately segment objects of different sizes. ASPP [48] constructs multi-scale receptive fields through dilated convolutions with different dilation rates, and uses multiple parallel dilated convolution branches to form a complete dilated convolution pyramid structure to enhance multi-scale context modelling capabilities. LR-ASPP [28] simplifies the structure of ASPP by reducing the number of dilated convolution branches and combining light-weight feature maps with global contextual information, thereby reducing its complexity to a certain extent. DAPPM [26] and PAPPM [16] are based on multi-scale pooling structures for context modelling. The former uses a hierarchical aggregation approach to fuse multi-scale features, while the latter improves computational efficiency through a parallel aggregation strategy. MFACB [29] stacks dilated convolutions with different dilation rates on the same feature layer to gradually expand the receptive field and capture multi-scale contextual information. Multi-scale feature extraction modules are typically executed on feature maps with the lowest resolution and the highest number of channels in the network. Therefore, extracting features from feature maps with a large number of channels, whether using mainstream parallel or cascaded serial architectures, will impose a huge burden on the network. So most of the existing multi-scale feature map extraction modules are difficult to apply to real-time tasks.

2.3 | Feature Enhancement

Attention mechanism is an effective way to enhance features. For the enhancement of low-level features, it not only requires the modification of self attention mechanisms, but also needs to obtain more guidance and supplementation from deep semantic information in high-level features through cross attention mechanisms. Corrective Self-Attention [49] utilises the correlation between local tokens to construct attention weights, thereby enhancing the spatial covariance of features and improving the consistency and discriminative ability of pixel level representations. The MSA mechanism [50] has been used to model the dependency relationships between different signal segments, thereby achieving adaptive weighting and modelling of key temporal features. Attention U-Net [51] achieves semantic interaction between low-level features and high-level features by introducing a gated attention mechanism in skip connections. Cross-Attention in Stable Diffusion [52] is not only responsible for aligning text descriptions with image regions, but its attention map itself also encodes rich semantic feature information. The combination of these two mechanisms with modules often leads to higher computational complexity, which affects the inference speed of the model.

2.4 | Feature Fusion

Feature fusion plays a crucial role in vision tasks, integrating information from various sources to extract more complete features. AFF [23] abandons traditional static linear concatenation and achieves dynamic, context aware adaptive aggregation of cross layer and cross dimensional features through a multi-scale channel attention mechanism. SFNet [53] introduces semantic flow, which achieves precise dynamic alignment between high-level semantics and low-level details in space by learning the transformation offset field between adjacent levels in the network. AlignSeg [54] has designed an alignment feature aggregation module and alignment context modelling, which explicitly solves the problem of spatial misalignment caused by multi-resolution feature fusion using learnable offsets. SCTNet [42] aligns Transformer and convolution through feature space, deeply distills global features during training, and embeds them into extremely simple single branch CNN to achieve fusion of branch features. Although they

effectively integrate different source features, they significantly hinder the inference speed of the model.

3 | Method

In this section, we provide a detailed description of our method. We first introduce the overall architecture of TriCrackNet, followed by introduction on the efficient atrous spatial pyramid pooling (EASPP), efficient feature enhancement module (EFEM), efficient feature fusion module (EFFM), and loss function.

3.1 | Overall Architecture for TriCrackNet

The key to achieving accurate crack segmentation lies in the network's ability to effectively extract semantic, spatial, and boundary features, while the key to high inference speed depends on a sufficiently efficient network architecture. Therefore, we propose a new three branch crack segmentation network-TriCrackNet aimed at effectively extracting the boundary, spatial, and semantic features of cracks while achieving efficient inference speed. The specific architecture of TriCrackNet is illustrated in Figure 2. Due to the fact that the model does not have a backpropagation process during inference, the dotted line in the model is only retained during the training phase and removed during the testing phase.

As shown in Figure 2. After the stem layer, the network is divided into three branches, and feature aggregation and fusion are achieved in the final EFFM. Then perform 8x upsampling and input it into the segmentation head to output the final crack segmentation mask.

Semantic Branch: Using the features output by Stem as input, downsampling and residual blocks [36] are performed to extract deep semantic features of cracks. And after the last downsampling, EASPP is used to extract the deep multi-scale features of the cracks. In upsampling, we construct a simple U-shaped architecture in the form of residual connections to improve the model while mitigating the decrease in inference speed. We place two segmentation heads at the last two residual connections, and further enhance the sensitivity of this branch to features of different scales through the constraint of multi-scale loss.

Spatial Branch: Using the features output by Stem as input, perform multiple convolutions while keeping the feature dimension unchanged. After the second convolutional block, EFEM is inserted to utilise the deep information of the semantic branch to guide the feature extraction of the spatial branch, while enhancing local features. We place a segmentation head at the EFEM to calculate spatial loss. In the process of upsampling the features of spatial branches, simple skip connections (additions) are made between the features in Stem and spatial features to supplement spatial information.

Border Branch: Using the features output by Stem as input, perform multiple convolutions while keeping the feature dimension unchanged. And insert EFEM to enhance feature expression. Place a boundary head at the EFEM and output an edge map $\in R^{1\times H\times W}$ to calculate boundary loss and enhance crack details.

3.2 | Efficient Atrous Spatial Pyramid Pooling (EASPP)

Due to the varying shapes and sizes of cracks, multi-scale feature extraction is essential for crack segmentation. The

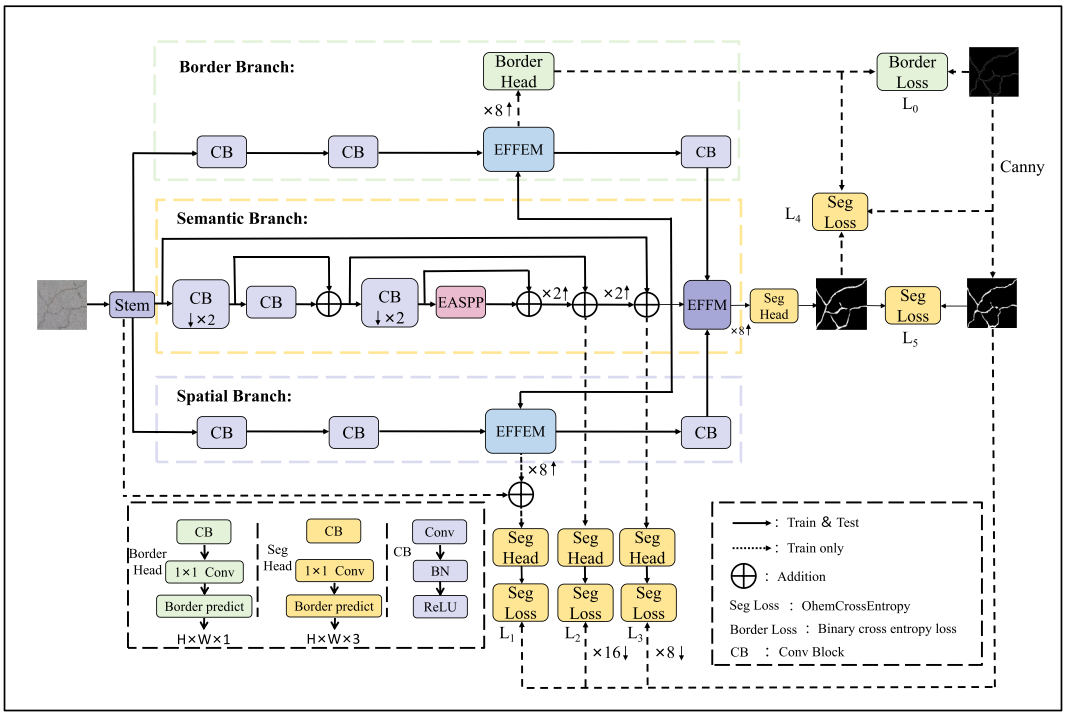


FIGURE 2 | Architecture of TriCrackNet.

multi-scale feature extraction module of traditional methods requires the entire feature map as input for serial or parallel feature extraction. Finally, all features are concatenated and convolved to preserve multi-scale features. This approach significantly reduces the inference speed of the model [16].

Therefore, we propose a plug and play efficient multi-scale feature extraction module, namely the Efficient Spatial Pyramid Pooling (EASPP) module. Inspired by PConv [25], EASPP groups features to avoid the traditional redundant calculation of using the entire set of features. And, a hierarchical residual connection structure is adopted instead of traditional concatenation and convolution, so that the output of each group not only depends on the features of the current layer, but also efficiently integrates the information of the previous group. Therefore, compared to traditional multi-scale feature extraction modules, EASPP achieves accurate multi-scale feature extraction while reducing the decrease in model inference speed. The specific architecture of the EASPP is illustrated in Figure 3.

Given feature $X \in R^{C\times H\times W}$, we split it along its channel dimension and divide it equally into four groups $\{X_i\}_{i=1}^{4} \in R^{C/4\times H\times W}$. Perform progressive dilation convolution and global average pooling on it. Among them, after performing feature extraction on each set of features, residual connections are made with the next set of features and used as input for the next feature extraction to fully and efficiently utilise feature X. Finally, we concatenate the features output from each convolution with the features upsampled by global average pooling, perform 1x1 convolution and residual concatenation to output a multi-scale feature map X_m. The specific calculation is shown in the following equation:

$$X_m = X + \mathrm{Conv}_{1\times1}\left(\mathrm{Concat}\left[\begin{array}{l}\underset{d=6}{\mathrm{Conv}_{3\times3}}(X_1),\\ \underset{d=12}{\mathrm{Conv}_{3\times3}}\left(\underset{d=6}{\mathrm{Conv}_{3\times3}}(X_1)+X_2\right),\\ \underset{d=18}{\mathrm{Conv}_{3\times3}}\left(\underset{d=12}{\mathrm{Conv}_{3\times3}}\left(\underset{d=6}{\mathrm{Conv}_{3\times3}}(X_1)+X_2\right)+X_3\right),\\ \mathrm{Upsample}\left(\mathrm{GAP}\left(\underset{d=18}{\mathrm{Conv}_{3\times3}}\left(\underset{d=12}{\mathrm{Conv}_{3\times3}}\right.\right.\right.\\ \left.\left.\left.\left(\underset{d=6}{\mathrm{Conv}_{3\times3}}(X_1)+X_2\right)+X_3\right)+X_4\right)\right)\end{array}\right]\right) \tag{1}$$

3.3 | Efficient Feature Enhancement Module (EFEM)

For the enhancement of low-level features, both self attention mechanism reinforcement and cross attention mechanism guidance based on high-level features are required. However, combining the existing two mechanisms will significantly reduce the inference speed of the model. Based on this, we have constructed an efficient feature enhancement module. It can replace traditional complex cross attention mechanisms by using only multiplication operations of global information between high-level and low-level features to achieve feature guided enhancement. It only replaces traditional self attention mechanisms with an activation function to achieve efficient spatial detail feature enhancement and noise removal. The specific architecture of this module is shown in Figure 4.

For low-level feature $X_{low} \in R^{C\times H\times W}$ and high-level feature $X_{high} \in R^{C\times H\times W}$, we first use global average pooling to efficiently obtain their global features F_1 and F_2. Then, subtract F_1 from X_{low} to obtain the local features of low-level features, convolve and sigmoid activate the global feature F_2 to obtain the semantic weight W_1, and multiply it with the global feature F_1 in low-level features to supplement semantic information for low-level features. Then, perform residual connections to output the enhanced global feature $F_1{}'$. Performing convolution on local features of low-level features and then activating them using Relu [2, 55] can further enhance spatial detail features that are more relevant to cracks, while removing noise points unrelated to foreground tasks. Finally, add it to $F_1{}'$ to obtain the enhanced feature $X_{low}{}'$. The specific calculation is shown in the following equation:

$$F_1 = GAP(X_{low}), F_2 = GAP(X_{high}) \tag{2}$$

$$W_1 = Sigmoid(Conv_{1\times1}(F_2)) \tag{3}$$

$$F_1{}' = F_1 + F_1 \times W_1 \tag{4}$$

$$X_{low}{}' = F_1{}' + ReLU^2(Conv_{1\times1}(X_{low} - F_1)) \tag{5}$$

3.4 | Efficient Feature Fusion Module (EFFM)

Feature fusion is crucial for combining information from different layers. However, traditional feature fusion methods often use multi-layer convolution or complex attention mechanisms, resulting in a significant decrease in the inference speed of the model. To address this, we propose an efficient, plug-and-play feature fusion module—efficient feature fusion module (EFFM). EFFM achieves efficient feature fusion through more direct

pooling, broadcasting, and weighting operations, with only a few convolution operations. The specific architecture of this module is illustrated in Figure 5.

Set spatial feature $X_S \in R^{C\times H\times W}$, boundary feature $X_B \in R^{C\times H\times W}$, and semantic feature $X_C \in R^{C\times H\times W}$. Firstly, perform a 3 × 3 convolution on X_B to output $X_{B1} \in R^{1\times H\times W}$, which we broadcast as boundary weights to both spatial and semantic branches to enhance the edge texture details of cracks in local and global features, and output X_{S0}, X_{C0}. Then, perform average pooling and maximum pooling operations on X_{S0} and X_{C0} along the channel dimension, and concatenate them together to obtain the representative features X_{S1} and X_{C1} of the two branches. Then perform matrix multiplication on X_{S1}, X_{C1}, and X_{B1}, followed by 3 × 3 convolution and sigmoid activation to output the three branch feature relationship matrix W_2. Finally, W_2 and $1 - W_2$ are multiplied with X_{S0} and X_{C0} as weights respectively, allowing the model to adaptively select the proportion of spatial and semantic features to output the final fused feature map X_f. The specific calculation is shown in the following equation:

$$X_{B1} = Conv_{3\times3}(X_B) \tag{6}$$

$$X_{S0}, X_{C0} = X_S + X_{B1}, X_C + X_{B1} \tag{7}$$

$$\begin{aligned} &X_{S1}, X_{C1} = \\ &Contact(meanpool(X_{S0}), maxpool(X_{S0})), \\ &Contact(meanpool(X_{C0}), maxpool(X_{C0})) \end{aligned} \tag{8}$$

$$W_2 = sigmoid(Conv_{3\times3}(X_{B1} \times X_{S1} \times X_{C1})) \tag{9}$$

$$X_f = X_{C0} \times W_2 + X_{S0} \times (1 - W_2) \tag{10}$$

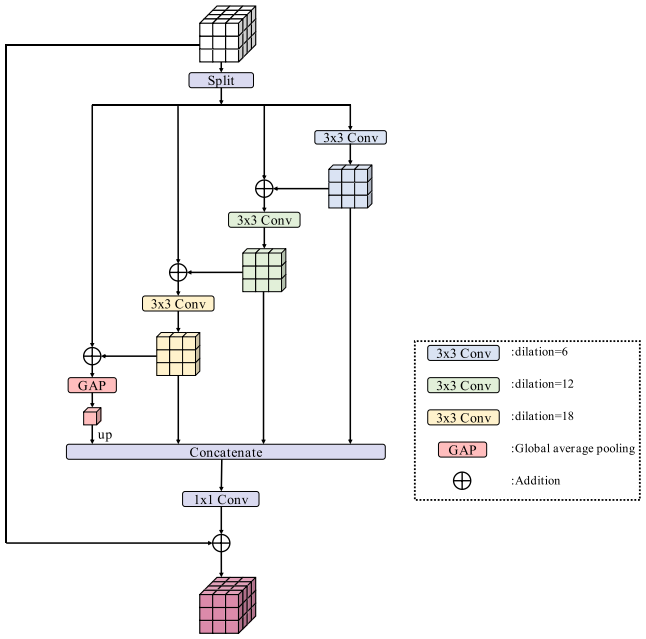


FIGURE 3 | Efficient atrous spatial pyramid pooling.

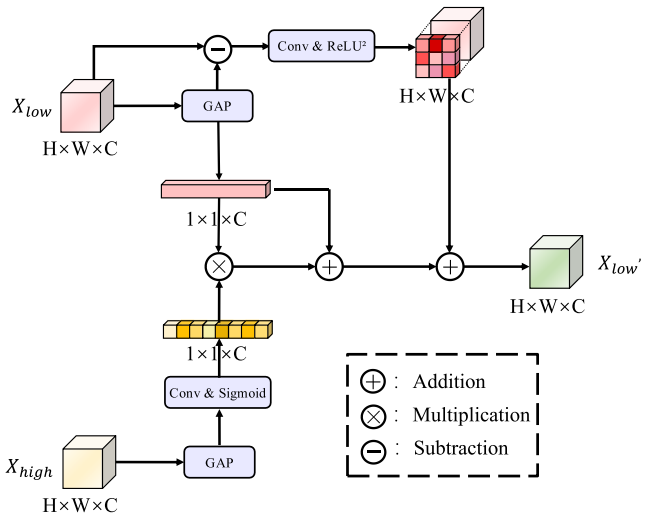


FIGURE 4 | Efficient feature enhancement module.

3.5 | Loss Function

There are a total of 6 loss functions used in TriCrackNet. We set the loss of the boundary branch to l_0, the loss of the spatial branch to l_1, the two multi-scale losses of the semantic branch to l_2 and l_3, and the losses calculated using the final crack segmentation mask to l_4 and l_5, respectively.l_0 is a binary cross entropy loss.l_1, l_2, l_3, l_4, l_5 represent CE loss. In loss l_4, we multiply the boundary map output by the boundary branch as a weight with the segmentation mask to further enhance the boundary information of the crack, and then perform loss calculation on the enhanced mask and GT. Therefore, the final loss l_{total} is as follows:

$$l_{total} = \lambda_0 l_0 + \lambda_1 l_1 + \lambda_2 l_2 + \lambda_3 l_3 + \lambda_4 l_4 + \lambda_5 l_5 \tag{11}$$

where $\lambda_0 = 1, \lambda_1 = 0.5, \lambda_2 = 0.5, \lambda_3 = 0.5, \lambda_4 = 5, \lambda_5 = 2$.

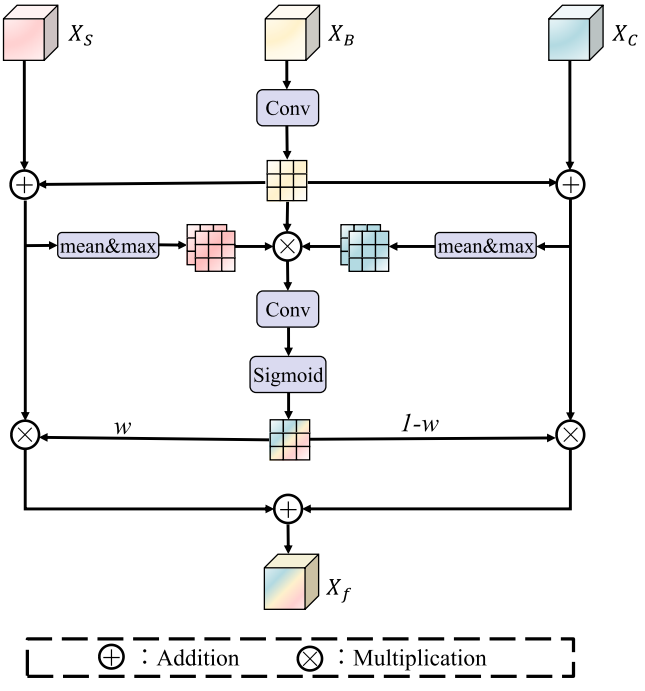


FIGURE 5 | Efficient feature fusion module.

4 | Experiments and Results

In this section, we first introduce the datasets used in this study and the data augmentation strategies, as well as the evaluation metrics. In the Implementation Details, we provide the settings for training and inference. Then, we conduct ablation experiments on the Crack500 dataset to verify the effectiveness of the modules in this paper. Finally, we compare our algorithm with other existing real-time semantic segmentation methods on the CrackForest, DeepCrack, Crack500 datasets, achieving state-of-the-art (SOTA) results on balance metrics, thus demonstrating that TriCrackNet effectively balances speed and accuracy.

4.1 | Datasets

4.1.1 | *Dataset Introduction*

CrackForest Dataset [32] is a road crack dataset consisting of 118 images, each with a resolution of 480 × 320 and corresponding labels.

Crack500 Dataset [33] is a dataset containing a large number of road cracks, consisting of 494 images, each with a resolution of 2560 × 1440 and corresponding labels.

DeepCrack Dataset [34] is a dataset containing a large number of road and concrete cracks, with a total of 537 images, each with a resolution of 544 × 384 and corresponding labels.

4.1.2 | *Data Augmentation*

To enhance the learning ability of the model and prevent overfitting. We performed data augmentation on three datasets separately (during the enhancement process, images with crack pixels accounting for less than 5% are removed.). We use magnification, random cropping, flipping, and rotation as data augmentation methods. The rotation angles are 45°, 60°, 90°. The enhanced results of each dataset and the configurations used for training, validation, and testing are shown in Table 1. The ratio of training set, validation set, and test set for all datasets is 6:2:2. Normalise the data to [0,1] before training. And no patching operation was applied to the images in our experiments.

4.1.3 | *Evaluation Metrics*

We selected accuracy metric (IoU_{Crack}), speed metric (FPS), balance metric (*WeightedScore*) as evaluation metrics for this experiment. The specific introduction of each metric is as follows:

Accuracy Metric: Since this segmentation task is a binary semantic segmentation, we only need to focus on the foreground IoU, denoted as IoU_{crack}. This metric is calculated based on the confusion matrix: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). For each category, the IoU is defined as the ratio of the intersection to the union of the predicted and actual values.

$$IoU = \frac{TP}{FP + TP + FN} \tag{12}$$

Speed Metric: For this task, the speed of the model is measured by the number of image frames processed per second, denoted as Frames Per Second (FPS).

Balance Metric: In real-time semantic segmentation tasks, both speed and accuracy are of equal importance. However, previous methods often evaluated performance based on only one of these metrics. This has led to a scenario where many methods achieve higher speeds at the expense of significantly reduced accuracy (or vice versa), which contradicts the initial purpose of real-time segmentation. To address this, we adopt a new evaluation metric called *WeightedScore*, used in studies [17, 56], which simultaneously integrates both speed and accuracy to effectively assess the overall performance of the model. Please refer to the 'Supporting Information S1' file for the pseudocode of *WeightedScore*.

$$\hat{IoU}_{crack_i} = \frac{IoU_{crack_i} - IoU_{crack_{\min}}}{IoU_{crack_{\max}} - IoU_{crack_{\min}}} \tag{13}$$

$$\hat{speed}_i = \frac{speed_i - speed_{\min}}{speed_{\max} - speed_{\min}} \tag{14}$$

$$WeightedScore = \alpha \times \hat{IoU}_{crack_i} + \beta \times \hat{speed}_i \tag{15}$$

Among them, both α and β are 0.5.

Speed and Accuracy Benchmarking Protocol: In the comparative experiments of all methods, we used input images of uniform resolution to ensure fairness in evaluation. The inference time of the model only includes the forward propagation time of the model, and does not include data loading, preprocessing, or post-processing time. All models are preheated 10 times before testing. The FPS value of each model is calculated based on its average inference speed on 1000 test images. All models are trained on a unified software and hardware benchmark platform to eliminate the potential impact of environmental differences on model performance and ensure the objective comparability of evaluation results.

TABLE 1 | Enhanced resolution and number of images for the CrackForest, Crack500 and DeepCrack datasets.

Dataset	Pixel size	Total quantity	Train quantity	Val quantity	Test quantity
CrackForest	480 × 320	18,629	11,177	3725	3727
Crack500	2560 × 1440	1986	1191	397	398
DeepCrack	544 × 384	15,887	9532	3177	3178

4.2 | Implementation Details

All experiments were tested on two NVIDIA GeForce RTX 4090 GPUs, each with 24 GB VRAM and 8.9 CUDA compute capability. The CPU used was an Intel Xeon Gold 6326 with a base frequency of 2.9 GHz, 32 logical cores, 64 GB of RAM, and support for the AVX512 instruction set. The experimental environment includes Ubuntu 18.04 and CUDA 11.6.

4.2.1 | *Training and Inference Settings*

Training Settings: We use Stochastic Gradient Descent (SGD) as the optimiser with a momentum of 0.9. The initial learning rate is set to 0.01, and the batch size is set to 8. We employ cosine annealing as the learning rate decay strategy (max learning rate: 0.01, min learning rate: 0.005), and set the maximum number of epochs to 300 to ensure the model fully converges.

Inference Settings: During inference, we remove all auxiliary heads from the models used for comparison, while the rest of the architecture remains unchanged to test the inference speed. The batch size for inference is set to 1.

4.3 | Ablation Experiments

In this paper, we designed a three branch architecture network that can simultaneously capture crack semantic, spatial, and boundary features, along with three modules: EASPP, EFEM, and EFFM. We conducted sufficient ablation experiments on the Crack500 dataset to demonstrate the effectiveness of each module and branch. After each ablation experiment, we use it as the baseline for the next ablation until all components of TriCrackNet are validated.

4.3.1 | *Three Branch Architecture Ablation*

The semantic, spatial, and edge features of foreground targets are crucial for semantic segmentation. So, first of all, to demonstrate the effectiveness of the TriCrackNet network architecture, we use the semantic branch as the baseline and sequentially add boundary and spatial branches. The three branch structure ablation experiments on both the Crack500 and DeepCrack datasets (Tables 2 and 3) showed that as the number of branches increased, the model's feature expression ability continued to enhance and its performance steadily improved. When using three branches simultaneously, the model can achieve the best results. On the Crack500 and DeepCrack datasets, there were improvements of 1.01% IoU_{crack} and 2.05% IoU_{crack}, respectively, compared to using the semantic branch alone.

4.3.2 | *Effect of the Efficient Atrous Spatial Pyramid Pooling*

EASPP adopts a strategy of feature grouping, hierarchical input, and progressively increasing receptive field, which avoids high redundant calculations, achieves efficient feature utilisation, and significantly improves the model's ability to extract multi-scale features of cracks. In this section, we use the three branch architecture constructed in this article as the baseline, and compare it with ASPP [48], DAPPM [26], PAPPM [16], and MFACB [29] to verify the effectiveness of the EASPP proposed in this paper. Tables 4 and 5 present the ablation experimental results of EASPP

TABLE 2 | The results of the three branch ablation experiment.

Branch			Crack500
Semantic	**Spatial**	**Border**	IoU_{crack}(%)
√			60.82
√	√		61.51
√		√	61.18
	√		60.71
	√	√	60.89
√	√	√	**61.83**

Note: Bold font represents the optimal result in the table.

TABLE 3 | The results of the three branch ablation experiment.

Branch			DeepCrack
Semantic	**Spatial**	**Border**	IoU_{crack}(%)
√			82.33
√	√		83.93
√		√	83.16
	√		82.17
	√	√	82.69
√	√	√	**84.38**

Note: Bold font represents the optimal result in the table.

TABLE 4 | Ablation study of EASPP on Crack500 dataset.

		Crack500	
Module	**FPS**	IoU_{crack}(%)	*WeightedScore*(%)
—	208.12	61.83	50.00
ASPP [48]	161.29	63.44	45.22
DAPPM [26]	175.83	63.52	63.00
PAPPM [16]	183.61	63.27	64.28
MFACB [29]	168.05	63.61	57.22
EASPP	199.49	63.53	**88.54**

Note: Bold font represents the optimal result in the table.

TABLE 5 | Ablation study of EASPP on DeepCrack dataset.

		DeepCrack	
Module	**FPS**	IoU_{crack}(%)	*WeightedScore*(%)
—	397.98	84.38	50.00
ASPP [48]	263.32	85.46	31.95
DAPPM [26]	274.71	85.54	38.55
PAPPM [16]	286.47	85.77	49.72
MFACB [29]	271.68	86.07	53.10
EASPP	332.76	86.02	**74.30**

Note: Bold font represents the optimal result in the table.

on the Crack500 and DeepCrack datasets. On the Crack500 and DeepCrack datasets, the addition of EASPP achieved the least FPS decay, with values of −8.63 and −65.22, respectively. At the same time, it ranks second in accuracy, with improvements of 1.7% IoU_{crack} and 1.64% IoU_{crack} compared to the baseline, and the gap with the top ranked MFACB is extremely small. And achieved the highest balance of 88.54% and 74.30% respectively. It can be seen that EASPP has a more balanced speed and accuracy compared to other multi-scale feature extraction modules. Please refer to the 'Supporting Information S1' file for the visualisation results of this experiment.

4.3.3 | *Effect of Efficient Feature Enhancement Module*

EFEM allows features to reinforce spatial information based on their own decisions, while allowing advanced features to provide semantic supplementation. So it further optimised the model's extraction of crack spatial features and boundary branch features to improve accuracy. Moreover, at the level of cross attention mechanism, we only used global features rich in semantic information to optimise low-level features, avoiding traditional redundant calculations. Tables 6 and 7 present the ablation experimental results of EFEM on the Crack500 and DeepCrack datasets. On both the Crack500 and DeepCrack datasets, EFEM ranked first in terms of accuracy improvement, with 0.96% IoU_{crack} and 0.86% IoU_{crack}, respectively. Although its speed is only second to SENET, its accuracy significantly surpasses all other competing methods. It also achieves the highest balance metrics of 75.61% and 81.21%, respectively. This demonstrates that EFEM offers a better balance between speed and accuracy compared to other feature enhancement modules. Please refer to the 'Supporting Information S1' file for the visualisation results of this experiment.

4.3.4 | *Effect of Efficient Feature Fusion Module*

An efficient feature fusion module is necessary for multi branch real-time segmentation networks. EFFM extracts key information from features of different branches for weighted fusion, and then assigns it as a weight matrix to response features to achieve efficient. Tables 8 and 9 present the ablation experimental results of EFFM on the Crack500 and DeepCrack datasets. On the Crack500 and DeepCrack datasets, EFFM achieved significant accuracy improvement with minimal FPS loss (−8.09 and −46.78, respectively), at 0.74% IoU_{crack} and 0.81% IoU_{crack}, respectively. Although its accuracy is not the highest, the addition of other modules will result in excessive speed loss, which does not meet the requirements of real-time segmentation. Moreover, EFFM achieved the best ranking in terms of balance metric. Therefore, EFFM has a good balance between speed and accuracy. Please refer to the 'Supporting Information S1' file for the visualisation results of this experiment.

4.4 | Comparative Experiments

To validate the effectiveness of TriCrackNet, we conducted thorough comparisons with existing state-of-the-art real-time segmentation methods on CrackForest, Crack500, and DeepCrack.

TABLE 6 | Ablation study of EFEM on Crack500 dataset.

		Crack500	
—	FPS	IoU_{crack}(%)	*WeightedScore*(%)
—	199.49	63.53	50.00
SENet [57]	193.91	63.82	51.24
Pag [16]	182.18	64.21	42.42
LUAFM [17]	181.72	64.19	40.24
ELGCA [58]	179.36	64.27	38.54
EFEM	189.67	64.49	**75.61**

Note: Bold font represents the optimal result in the table.

TABLE 7 | Ablation study of EFEM on DeepCrack dataset.

		DeepCrack	
—	FPS	IoU_{crack}(%)	*WeightedScore*(%)
—	332.76	86.02	50.00
SENet [57]	311.62	86.46	61.29
Pag [16]	279.44	86.54	44.18
LUAFM [17]	267.52	86.64	41.94
ELGCA [58]	258.81	86.77	43.60
EFEM	304.97	86.88	**81.21**

Note: Bold font represents the optimal result in the table.

TABLE 8 | Ablation study of EFFM on Crack500 dataset.

		Crack500	
Module	FPS	IoU_{crack}(%)	*WeightedScore*(%)
Add	189.67	64.49	50.00
SKFF [59]	174.46	64.97	44.04
AFF [23]	171.85	65.27	56.80
Bag [16]	175.11	65.19	59.52
KSFA [60]	168.74	65.28	50.00
EFFM	181.58	65.23	**77.51**

Note: Bold font represents the optimal result in the table.

TABLE 9 | Ablation study of EFFM on DeepCrack dataset.

		DeepCrack	
Module	FPS	IoU_{crack}(%)	*WeightedScore*(%)
ADD	304.97	86.88	50.00
SKFF [59]	231.69	87.36	34.12
AFF [23]	229.93	87.46	29.41
Bag [16]	246.56	87.57	51.67
KSFA [60]	232.87	87.73	51.96
EFFM	258.19	87.69	**66.48**

Note: Bold font represents the optimal result in the table.

4.4.1 | *Results on Crack500 Dataset*

As shown in Table 10, the FPS of TriCrackNet is 181.58 and IoU_{crack} is 65.23% on Crack500 dataset. From a single indicator

TABLE 10 | The results of comparative experiments on the Crack500 dataset.

Model	GPU	Resolution	FPS	GFLOPs (G)	Params (M)	Crack500	
						IoU_{crack}(%)	*WeightedScore*
BiseNetV1 (res18) [13]	RTX4090	2560 × 1440	72.28	429.17	13.39	62.11	10.57
DABNet [30]	RTX4090	2560 × 1440	67.83	147.34	0.73	63.76	9.30
BiseNetV2 [14]	RTX4090	2560 × 1440	71.06	500.84	5.19	64.27	44.83
DDRNet-23-S [26]	RTX4090	2560 × 1440	200.29	127.57	5.69	62.28	49.80
ECSNet [28]	RTX4090	2560 × 1440	35.23	89.57	0.25	63.34	19.71
LPS-Net [27]	RTX4090	2560 × 1440	**210.56**	37.14	3.37	64.49	88.14
PIDNet-S-Simple [16]	RTX4090	2560 × 1440	119.51	167.64	7.62	65.01	70.51
DSNet-head64 [29]	RTX4090	2560 × 1440	47.28	368.26	6.22	65.08	51.03
CarNet [31]	RTX4090	2560 × 1440	47.23	270.40	4.89	65.13	51.82
TriCrackNet	RTX4090	2560 × 1440	181.58	135.35	6.44	**65.23**	**91.74**

Note: Bold font represents the optimal result in the table.

perspective, TriCrackNet ranks first in accuracy and third in speed. Compared to LPSNet, which ranks first in speed, it is 28.98 FPS slower but 0.74% IoU_{crack} more accurate. Since the difference between speed and accuracy is not significant, we need to pay more attention to *WeightedScore* at this point. The *WeightedScore* of TriCrackNet is 91.74%, while that of LPSNet is 88.14%. Therefore, it can be concluded that compared to existing SOTA methods, TriCrackNet achieves the best balance between speed and accuracy on the CrackForest dataset. And from Figure 6, it can be seen that the segmentation generated by TriCrackNet is more similar to GT compared to other methods.

4.4.2 | *Results on DeepCrack Dataset*

As shown in Table 11, the FPS of TriCrackNet on DeepCrack is 258.19 FPS and IoU_{crack} is 87.69%. From a single indicator perspective, we rank first in accuracy and second in speed. But compared to BiseNetV1, which ranks first in speed, TriCrackNet has an accuracy 4.41% IoU_{crack} higher than it. Meanwhile, it achieved a balanced score of 88.62%, far exceeding other methods. So it can be seen that compared with existing SOTA methods, TriCrackNet achieves the best balance between speed and accuracy on the DeepCrack dataset. From Figure 7, it can be seen that the segmentation mask generated by TriCrackNet is closest to GT.

4.4.3 | *Results on CrackForest Dataset*

As shown in Table 12, TriCrackNet achieved a FPS of 274.39 and an IoU_{crack} of 73.45% on CrackForest. Although in terms of single indicator, speed and accuracy rank second compared to other methods. But compared to DSNet, which ranks first in accuracy, its FPS is 54.54 higher; Compared to BiseNetV1, which ranks first in speed, its accuracy is 3.03% higher. And the *WeightedScore* is 92.13% far exceeding other methods. Therefore, we can know that compared to existing SOTA methods, TriCrackNet achieves the best balance between speed and accuracy on the CrackForest dataset. From Figure 8, we can see that the segmentation mask generated by TriCrackNet is very close to GT, and its visualisation results are almost equivalent to DSNet.

4.4.4 | *Comparison of Generalisation Experiments*

To verify the generalisation of TriCrackNet, we trained it on a mixed dataset consisting of CrackForest and Crack500, and tested it on the DeepCrack dataset. The training set of the DeepCrack dataset consists of 9532 images. To ensure fairness, we randomly selected 4766 images from the training set of the CrackForest dataset. Re enhance the Crack500 dataset using the data augmentation method described in Section 4.1.2, and randomly select 4766 images. And scale the images in the mixed dataset to the same size as the images in the DeepCrack dataset.

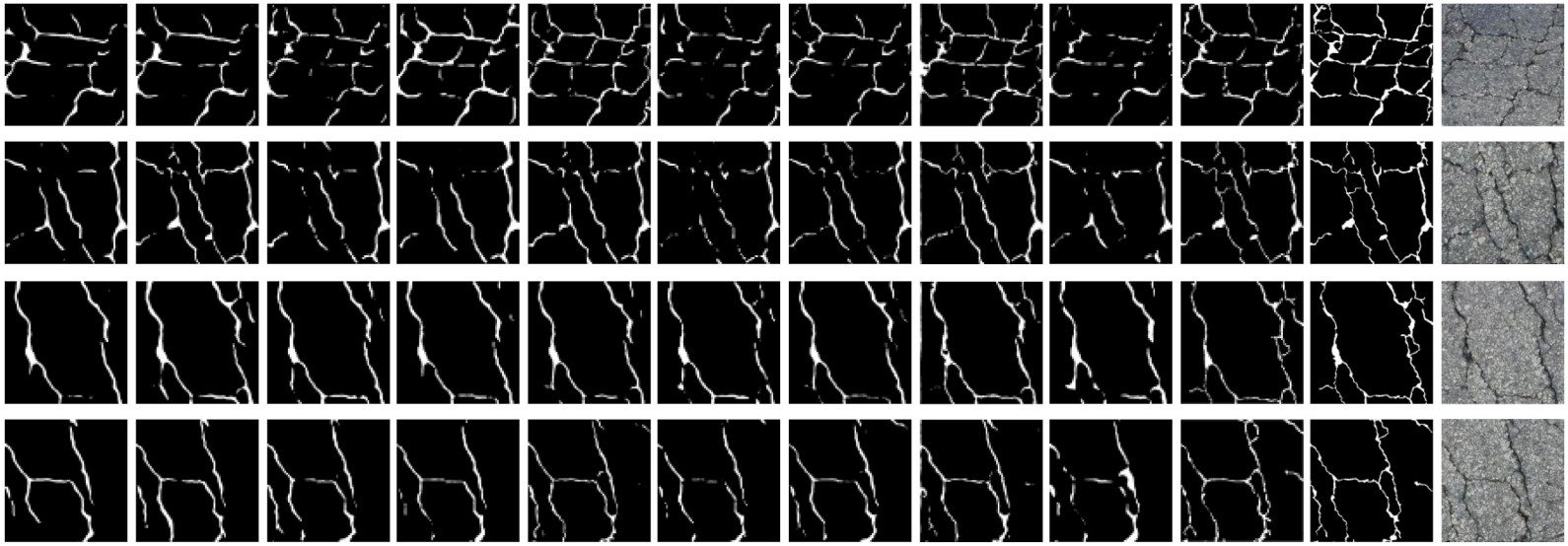


FIGURE 6 | Comparison of visualisation results between TriCrackNet and existing SOTA methods on the Crack500 dataset. From top to bottom, they are the original image, ground truth and inference results from TriCrackNet, DSNet [29], CarNet [31], PIDNet [16], LPS-Net [27], ECSNet [28], DDRNet-Slim [26], BiseNetV2 [14], DABNet [30], BiseNetV1 [13].

TABLE 11 | The results of comparative experiments on the DeepCrack dataset.

Model	GPU	Resolution	FPS	GFLOPs (G)	Params (M)	DeepCrack	
						$IoU_{crack}(\%)$	*WeightedScore*
BiseNetV1 (res18) [13]	RTX4090	544 × 384	**293.49**	24.32	13.39	83.28	50.00
DABNet [30]	RTX4090	544 × 384	186.36	8.35	0.73	84.38	27.92
BiseNetV2 [14]	RTX4090	544 × 384	224.55	28.38	5.19	84.36	40.01
DDRNet-23-S [26]	RTX4090	544 × 384	241.54	7.24	5.69	86.16	65.90
ECSNet [28]	RTX4090	544 × 384	138.45	5.08	0.25	85.97	30.50
LPS-Net [27]	RTX4090	544 × 384	222.39	2.13	3.37	87.64	76.50
PIDNet-S-Simple [16]	RTX4090	544 × 384	144.26	9.51	7.62	87.51	49.83
DSNet-head64 [29]	RTX4090	544 × 384	211.13	20.87	6.22	87.33	69.36
CarNet [31]	RTX4090	544 × 384	252.62	15.32	4.89	87.26	81.94
TriCrackNet	RTX4090	544 × 384	258.19	7.67	6.44	**87.69**	**88.62**

Note: Bold font represents the optimal result in the table.

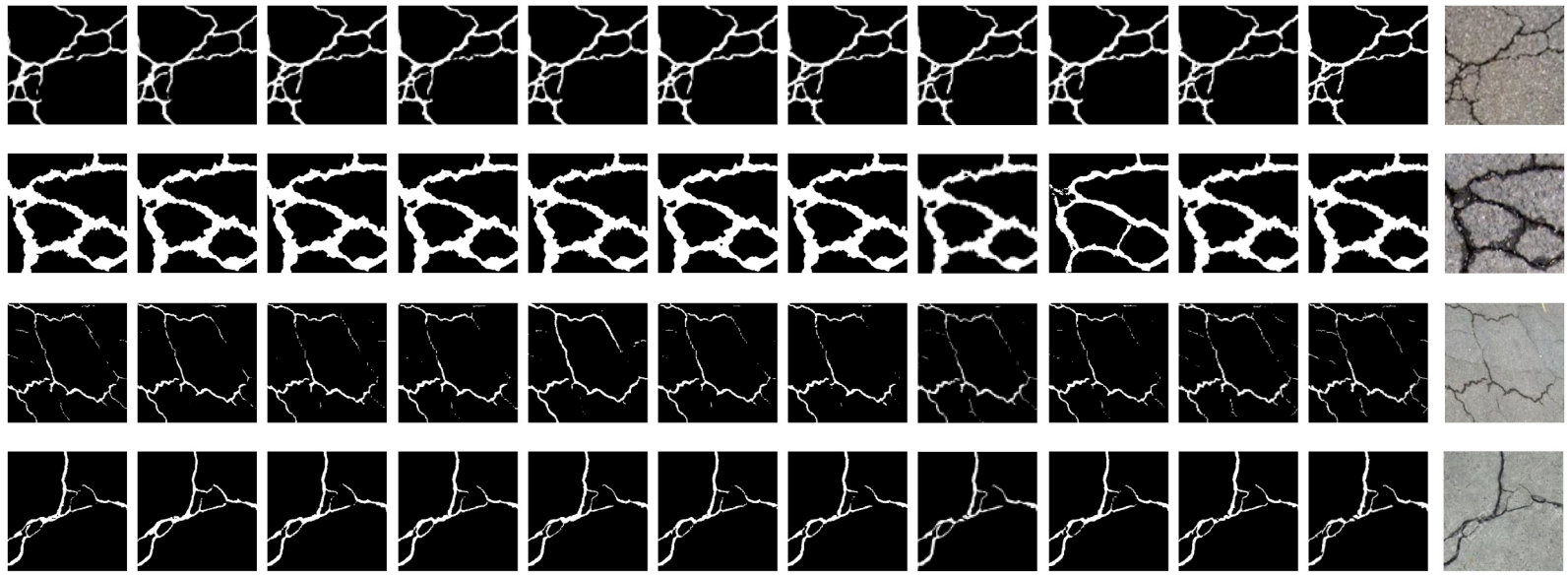


FIGURE 7 | Comparison of visualisation results between TriCrackNet and existing SOTA methods on the DeepCrack dataset. From top to bottom, they are the original image, ground truth and inference results from TriCrackNet, DSNet [29], CarNet [31], PIDNet [16], LPS-Net [27], ECSNet [28], DDRNet-Slim [26], BiseNetV2 [14], DABNet [30], BiseNetV1 [13].

TABLE 12 | The results of comparative experiments on the CrackForest dataset.

Model	GPU	Resolution	FPS	GFLOPs (G)	Params (M)	CrackForest	
						$IoU_{crack}(\%)$	*WeightedScore*
BiseNetV1 (res18) [13]	RTX4090	480 × 320	**295.52**	17.88	13.39	70.42	66.70
DABNet [30]	RTX4090	480 × 320	199.04	6.14	0.73	69.35	22.17
BiseNetV2 [14]	RTX4090	480 × 320	242.74	20.87	5.19	68.87	31.95
DDRNet-23-S [26]	RTX4090	480 × 320	252.13	5.32	5.69	69.84	45.61
ECSNet [28]	RTX4090	480 × 320	149.33	3.73	0.25	71.83	31.90
LPS-Net [27]	RTX4090	480 × 320	237.61	1.57	3.37	72.97	74.37
PIDNet-S-Simple [16]	RTX4090	480 × 320	156.87	6.99	7.62	72.86	45.57
DSNet-head64 [29]	RTX4090	480 × 320	219.85	15.35	6.22	**73.51**	74.12
CarNet [31]	RTX4090	480 × 320	267.78	11.27	4.89	72.73	82.11
TriCrackNet	RTX4090	480 × 320	274.39	5.64	6.44	73.45	**92.13**

Note: Bold font represents the optimal result in the table.

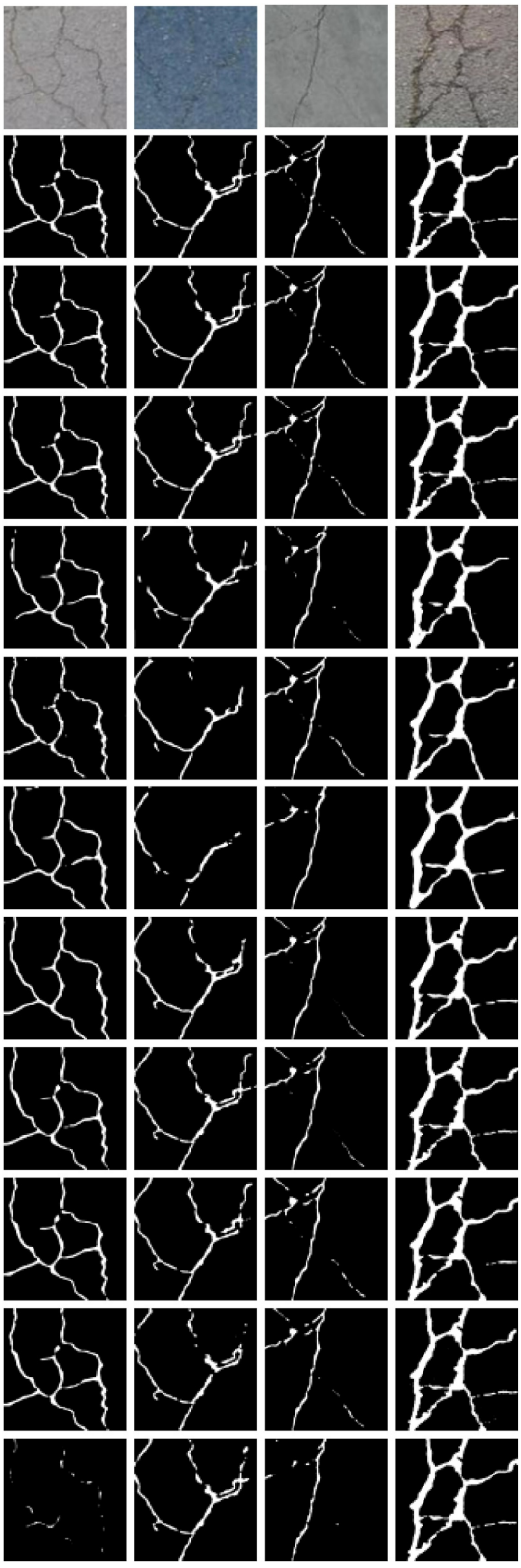


FIGURE 8 | Comparison of visualisation results between TriCrackNet and existing SOTA methods on the CrackForest dataset. From top to bottom, they are the original image, ground truth and inference results from TriCrackNet, DSNet [29], CarNet [31], PIDNet [16], LPS-Net [27], ECSNet [28], DDRNet-Slim [26], BiseNetV2 [14], DABNet [30], BiseNetV1 [13].

As shown in Table 13, the FPS of TriCrackNet on DeepCrack is 258.19 FPS and IoU_{crack} is 56.97%. From a single indicator perspective, we rank first in accuracy and second in speed. But compared to BiseNetV1, which ranks first in speed, TriCrackNet has an accuracy 5.18% IoU_{crack} higher than it. Meanwhile, it achieved a balanced score of 88.62%, far exceeding other methods. Therefore, it can be seen that compared with existing SOTA methods, TriCrackNet has better generalisation performance. From Figure 9, it can be seen that the segmentation mask generated by TriCrackNet is closest to GT.

4.5 | Fault Case Analysis

Although the TriCrackNet proposed in this paper can efficiently and accurately achieve road crack segmentation, there are still certain limitations. As shown in the red box in Figure 10, false positives may occur when the cracks are similar to the background texture. As shown in the yellow box in Figure 10, when there is noise such as stains in the image, cracks are difficult to be fully segmented.

4.6 | Limitations and Future Work

Limitations: From the experimental content in our previous section, it can be seen that although our method achieves a good balance between accuracy and speed, compared to some methods, its GFLOPS and Params are not lightweight enough. Although we achieved the best generalisation results compared to other approaches, there is still a significant gap compared to the results obtained from training and testing on the same dataset. Meanwhile, our method may produce some false positives when dealing with images of cracks and backgrounds that are similar.

Future work: In future work, firstly, we need to further optimise the lightweight performance of the model. Secondly, we will further enhance the generalisation performance of the model. Finally, enhance the robustness of the model to maintain high accuracy even in complex environments.

5 | Conclusion

This paper constructs a new three branch segmentation network TracRackNet, which efficiently and accurately captures the semantic, spatial, and boundary features of cracks. Meanwhile, the efficient atrous spatial pyramid pooling (EASPP), efficient feature enhancement module (EFEM), and efficient feature fusion module (EFFM) proposed in this paper are integrated into the network to further effectively improve the feature extraction capability of the improved model. We conducted comprehensive comparative experiments with other SOTA methods on multiple crack datasets to demonstrate that TriCrackNet achieves the optimal balance between speed and accuracy.

TABLE 13 | Generalisation experiment results on DeepCrack dataset.

Model	GPU	Resolution	FPS	GFLOPs (G)	Params (M)	DeepCrack	
						$IoU_{crack}(\%)$	*WeightedScore*
BiseNetV1 (res18) [13]	RTX4090	544 × 384	**293.49**	24.32	13.39	51.79	57.47
DABNet [30]	RTX4090	544 × 384	186.36	8.35	0.73	50.88	15.45
BiseNetV2 [14]	RTX4090	544 × 384	224.55	28.38	5.19	53.97	53.14
DDRNet-23-S [26]	RTX4090	544 × 384	241.54	7.24	5.69	55.61	72.08
ECSNet [28]	RTX4090	544 × 384	138.45	5.08	0.25	51.65	6.32
LPS-Net [27]	RTX4090	544 × 384	222.39	2.13	3.37	56.08	69.76
PIDNet-S-Simple [16]	RTX4090	544 × 384	144.26	9.51	7.62	55.26	37.83
DSNet-head64 [29]	RTX4090	544 × 384	211.13	20.87	6.22	53.70	46.59
CarNet [31]	RTX4090	544 × 384	252.62	15.32	4.89	56.12	79.84
TriCrackNet	RTX4090	544 × 384	258.19	7.67	6.44	**56.97**	**88.62**

Note: Bold font represents the optimal result in the table.

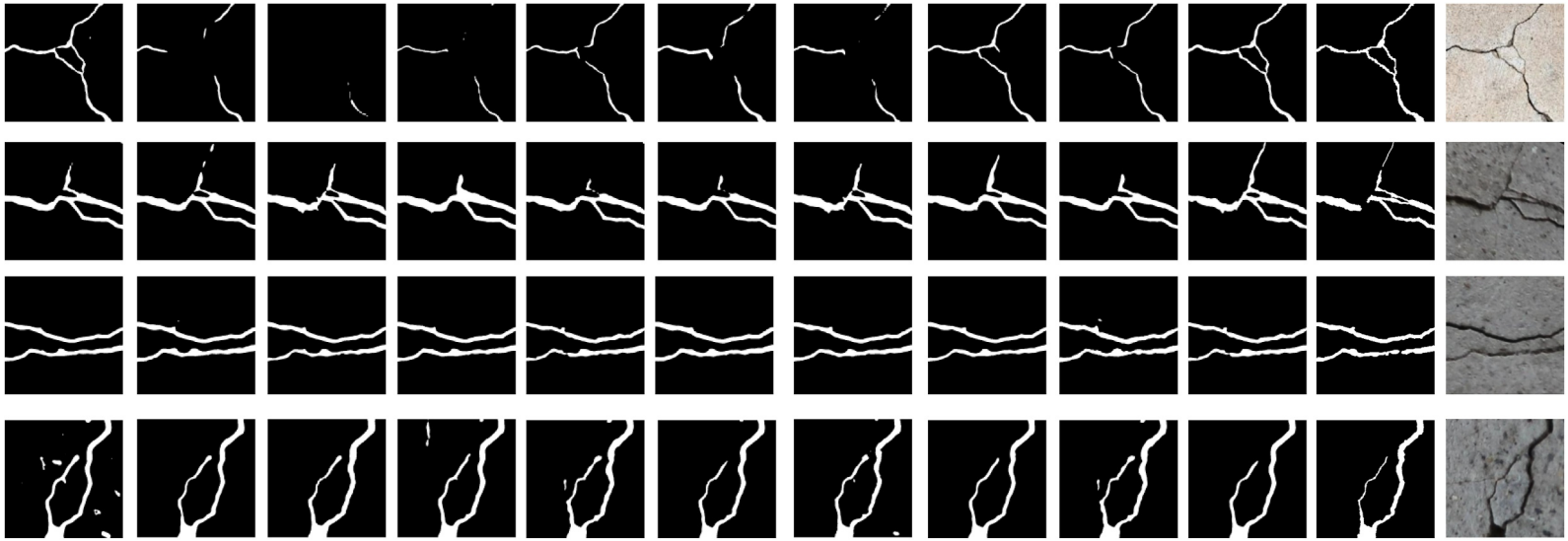


FIGURE 9 | Comparison of visualisation results of generalisation experiments on the DeepCrack dataset. From top to bottom, they are the original image, ground truth and inference results from TriCrackNet, DSNet [29], CarNet [31], PIDNet [16], LPS-Net [27], ECSNet [28], DDRNet-Slim [26], BiseNetV2 [14], DABNet [30], BiseNetV1 [13].

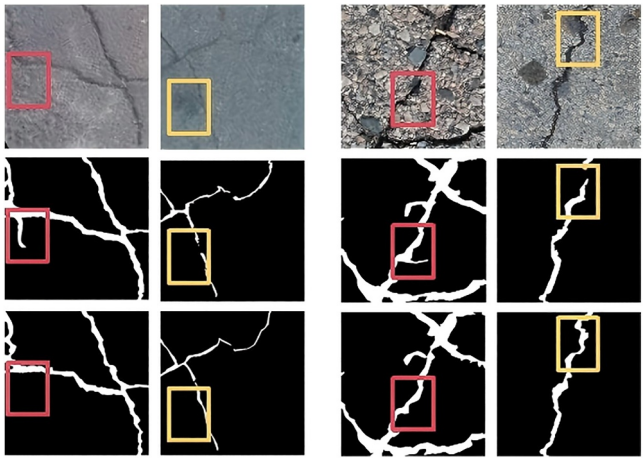


FIGURE 10 | The Failed Cases of TriCrackNet. From top to bottom, they are the original images, inference results from TriCrackNet and ground truth.

Acknowledgements

The authors would like to thank BGRIMM Technology Group for providing the data for this study.

Funding

The authors have nothing to report.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

All datasets can be accessed online.

References

1. Z. Chen, Z. Lai, J. Chen, and J. Li, "Mind Marginal Non-Crack Regions: Clustering-Inspired Representation Learning for Crack Segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2024), 12698–12708.

2. C. Han, H. Yang, and Y. Yang, "Enhancing Pixel-Level Crack Segmentation With Visual Mamba and Convolutional Networks," *Automation in Construction* 168 (2024): 105770, https://doi.org/10.1016/j.autcon.2024.105770.

3. C. Zhu, L. Li, Y. Wu, and Z. Sun, "Saswot: Real-Time Semantic Segmentation Architecture Search Without Training," *Proceedings of the AAAI Conference on Artificial Intelligence* 38, no. 7 (2024): 7722–7730, https://doi.org/10.1609/aaai.v38i7.28606.

4. Y. Wu, S. Li, J. Zhang, Y. Li, Y. Li, and Y. Zhang, "Dual Attention Transformer Network for Pixel-Level Concrete Crack Segmentation Considering Camera Placement," *Automation in Construction* 157 (2024): 105166, https://doi.org/10.1016/j.autcon.2023.105166.

5. M. Ren, Y. Li, T. Hussain, Y. Wu, and J. Li, "Pixel-Level Concrete Crack Quantification Through Super Resolution Reconstruction and Multi-Modality Fusion," *Advanced Engineering Informatics* 69 (2026): 103807, https://doi.org/10.1016/j.aei.2025.103807.

6. Y. Wang, S. Li, Y. Zhang, and Y. Li, "Lightweight Concrete Crack Segmentation Network for Drone Image With Complex Backgrounds Using Multi-Scale Feature Fusion and Optimized Architecture," *Construction and Building Materials* 495 (2025): 143667, https://doi.org/10.1016/j.conbuildmat.2025.143667.

7. X. Ma, Y. Li, Z. Yang, S. Li, and Y. Li, "Lightweight Network for Millimeter-Level Concrete Crack Detection With Dense Feature Connection and Dual Attention," *Journal of Building Engineering* 94 (2024): 109821, https://doi.org/10.1016/j.jobe.2024.109821.

8. G. Su, Y. Qin, H. Xu, and J. Liang, "Automatic Real-Time Crack Detection Using Lightweight Deep Learning Models," *Engineering Applications of Artificial Intelligence* 138 (2024): 109340, https://doi.org/10.1016/j.engappai.2024.109340.

9. H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for Real-Time Semantic Segmentation on High-Resolution Images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, (2018), 405–420.

10. Y. Li, R. Ma, H. Liu, and G. Cheng, "Real-Time High-Resolution Neural Network With Semantic Guidance for Crack Segmentation," *Automation in Construction* 156 (2023): 105112, https://doi.org/10.1016/j.autcon.2023.105112.

11. A. G. Howard, M. Zhu, B. Chen, et al., "Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861* (2017).

12. T. H. Tsai and Y. W. Tseng, "BiSeNet V3: Bilateral Segmentation Network With Coordinate Attention for Real-Time Semantic Segmentation," *Neurocomputing* 532 (2023): 33–42, https://doi.org/10.1016/j.neucom.2023.02.025.

13. C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral Segmentation Network for Real-Time Semantic Segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, (2018), 325–341.

14. C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "Bisenet v2: Bilateral Network With Guided Aggregation for Real-Time Semantic Segmentation," *International Journal of Computer Vision* 129, no. 11 (2021): 3051–3068, https://doi.org/10.1007/s11263-021-01515-2.

15. Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang, "Deepcrack: Learning Hierarchical Convolutional Features for Crack Detection," *IEEE Transactions on Image Processing* 28, no. 3 (2018): 1498–1512, https://doi.org/10.1109/tip.2018.2878966.

16. J. Xu, Z. Xiong, and S. P. Bhattacharyya, "PIDNet: A Real-Time Semantic Segmentation Network Inspired by PID Controllers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2023), 19529–19539.

17. X. Shi, Z. Yin, G. Han, et al., "BSSNet: A Real-Time Semantic Segmentation Network for Road Scenes Inspired From AutoEncoder," *IEEE Transactions on Circuits and Systems for Video Technology* (2023), https://doi.org/10.1109/TCSVT.2023.3325360.

18. R. Xian, X. Xiong, H. Peng, J. Wang, A. R. de Arellano Marrero, and Q. Yang, "Feature Fusion Method Based on Spiking Neural Convolutional Network for Edge Detection," *Pattern Recognition* 147 (2024): 110112, https://doi.org/10.1016/j.patcog.2023.110112.

19. J. Shen, Y. Chen, Y. Liu, X. Zuo, H. Fan, and W. Yang, "ICAFusion: Iterative Cross-Attention Guided Feature Fusion for Multispectral Object Detection," *Pattern Recognition* 145 (2024): 109913, https://doi.org/10.1016/j.patcog.2023.109913.

20. W. Liu, Y. Gong, G. Zhang, J. Lu, Y. Zhou, and J. Liao, "GLMDriveNet: Global–Local Multimodal Fusion Driving Behavior Classification Network," *Engineering Applications of Artificial Intelligence* 129, no. 107575 (2024): 20232024, https://doi.org/10.1016/j.engappai.2023.107575.

21. X. Liu, Z. Wu, A. Li, et al., "NTIRE 2024 Challenge on Low Light Image Enhancement: Methods and Results," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2024), 6571–6594.

22. Z. Chen, Z. He, and Z. M. Lu, "DEA-Net: Single Image Dehazing Based on Detail-Enhanced Convolution and Content-Guided Attention," *IEEE Transactions on Image Processing* 33 (2024): 1002–1015, https://doi.org/10.1109/tip.2024.3354108.

23. Y. Dai, F. Gieseke, S. Oehmcke, Y. Wu, and K. Barnard, "Attentional Feature Fusion," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, (2021), 3560–3569.

24. W. Liu, J. Lu, J. Liao, et al., "Feature-Attention-Embedding-Based Multimodal-Fusion Driving-Behavior-Classification Network," *IEEE Transactions on Computational Social Systems* (2024), https://doi.org/10.1109/TCSS.2024.3411486.

25. J. Chen, S. H. Kao, H. He, et al., "Run, Don't Walk: Chasing Higher FLOPS for Faster Neural Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2023), 12021–12031.

26. H. Pan, Y. Hong, W. Sun, and Y. Jia, "Deep Dual-Resolution Networks for Real-Time and Accurate Semantic Segmentation of Traffic Scenes," *IEEE Transactions on Intelligent Transportation Systems* 24, no. 3 (2022): 3448–3460, https://doi.org/10.1109/tits.2022.3228042.

27. Y. Zhang, T. Yao, Z. Qiu, and T. Mei, "Lightweight and Progressively-Scalable Networks for Semantic Segmentation," *International Journal of Computer Vision* 131, no. 8 (2023): 2153–2171, https://doi.org/10.1007/s11263-023-01801-1.

28. T. Zhang, D. Wang, and Y. Lu, "ECSNet: An Accelerated Real-Time Image Segmentation CNN Architecture for Pavement Crack Detection," *IEEE Transactions on Intelligent Transportation Systems* 24, no. 12 (2023): 15105–15112, https://doi.org/10.1109/tits.2023.3300312.

29. Z. Guo, L. Bian, H. Wei, J. Li, H. Ni, and X. Huang, "Dsnet: A Novel Way to Use Atrous Convolutions in Semantic Segmentation," *IEEE Transactions on Circuits and Systems for Video Technology* (2024), https://doi.org/10.1109/TCSVT.2024.3509504.

30. G. Li, I. Yun, J. Kim, and J. Kim, "Dabnet: Depth-Wise Asymmetric Bottleneck for Real-Time Semantic Segmentation," *arXiv preprint arXiv: 1907.11357* (2019).

31. K. Li, J. Yang, S. Ma, et al., "Rethinking Lightweight Convolutional Neural Networks for Efficient and High-Quality Pavement Crack Detection," *IEEE Transactions on Intelligent Transportation Systems* 25, no. 1 (2024): 237–250, https://doi.org/10.1109/tits.2023.3307286.

32. Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, "Automatic Road Crack Detection Using Random Structured Forests," *IEEE Transactions on Intelligent Transportation Systems* 17, no. 12 (2016): 3434–3445, https://doi.org/10.1109/tits.2016.2552248.

33. F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, and H. Ling, "Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection," *IEEE Transactions on Intelligent Transportation Systems* 21, no. 4 (2019): 1525–1535, https://doi.org/10.1109/tits.2019.2910595.

34. Y. Liu, J. Yao, X. Lu, R. Xie, and L. Li, "DeepCrack: A Deep Hierarchical Feature Learning Architecture for Crack Segmentation," *Neurocomputing* 338 (2019): 139–153, https://doi.org/10.1016/j.neucom.2019.01.036.

35. H. Wang, X. Jiang, H. Ren, Y. Hu, and S. Bai, "Swiftnet: Real-Time Video Object Segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2021), 1296–1305.

36. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2016), 770–778.

37. H. Li, P. Xiong, H. Fan, and J. Sun, "Dfanet: Deep Feature Aggregation for Real-Time Semantic Segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2019), 9522–9531.

38. F. Chollet, "Xception: Deep Learning With Depthwise Separable Convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2017), 1251–1258.

39. M. Gamal, M. Siam, and M. Abdel-Razek, "Shuffleseg: Real-Time Semantic Segmentation Network," *arXiv preprint arXiv:1803.03816* (2018).

40. X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2018), 6848–6856.

41. G. Zhu, J. Liu, Z. Fan, et al., "A Lightweight Encoder–Decoder Network for Automatic Pavement Crack Detection," *Computer-Aided Civil and Infrastructure Engineering* 39, no. 12 (2024): 1743–1765, https://doi.org/10.1111/mice.13103.

42. Z. Xu, D. Wu, C. Yu, X. Chu, N. Sang, and C. Gao, "Sctnet: Single-Branch Cnn With Transformer Semantic Information for Real-Time Segmentation," *Proceedings of the AAAI Conference on Artificial Intelligence* 38, no. 6 (2024): 6378–6386, https://doi.org/10.1609/aaai.v38i6.28457.

43. L. Chen, H. Dai, and Y. Zheng, "RAFNet: Reparameterizable Across-Resolution Fusion Network for Real-Time Image Semantic Segmentation," *IEEE Transactions on Circuits and Systems for Video Technology* 34, no. 2 (2023): 1212–1227, https://doi.org/10.1109/tcsvt.2023.3293166.

44. J. Wang, C. Gou, Q. Wu, et al., "Rtformer: Efficient Design for Real-Time Semantic Segmentation With Transformer," *Advances in Neural Information Processing Systems* 35 (2022): 7423–7436.

45. M. H. Guo, Z. N. Liu, T. J. Mu, and S. M. Hu, "Beyond Self-Attention: External Attention Using Two Linear Layers for Visual Tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, no. 5 (2022): 5436–5447, https://doi.org/10.1109/TPAMI.2022.3211006.

46. B. Liu, J. Kang, H. Guan, et al., "Boundary-Guided Real-Time Semantic Segmentation and Pixel-Level Quantification of Pavement Cracks," *IEEE Transactions on Intelligent Transportation Systems* (2025), https://doi.org/10.1109/TITS.2025.3642011.

47. Z. Yin, L. Qin, G. Han, et al., "DDSNet: Deep Dual-Branch Networks for Surface Defect Segmentation," *IEEE Transactions on Instrumentation and Measurement* (2024), https://doi.org/10.1109/TIM.2024.3427806.

48. L. C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," *arXiv preprint arXiv:1706.05587* (2017).

49. F. Wang, J. Mei, and A. Yuille, "Sclip: Rethinking Self-Attention for Dense Vision-Language Inference," in *European Conference on Computer Vision*, (2024), 315–332.

50. L. Cui, X. Tian, Q. Wei, and Y. Liu, "A Self-Attention Based Contrastive Learning Method for Bearing Fault Diagnosis," *Expert Systems With Applications* 238 (2024): 121645, https://doi.org/10.1016/j.eswa.2023.121645.

51. O. Oktay, J. Schlemper, L. L. Folgoc, et al., "Attention U-Net: Learning Where to Look for the Pancreas," *arXiv preprint arXiv: 1804.03999* (2018).

52. B. Liu, C. Wang, T. Cao, K. Jia, and J. Huang, "Towards Understanding Cross and Self-Attention in Stable Diffusion for Text-Guided Image Editing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2024), 7817–7826.

53. X. Li, A. You, Z. Zhu, et al., "Semantic Flow for Fast and Accurate Scene Parsing," in *Computer Vision–ECCV 2020: 16Th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16* (Springer International Publishing, 2020), 775–793.

54. Z. Huang, Y. Wei, X. Wang, W. Liu, T. S. Huang, and H. Shi, "Alignseg: Feature-Aligned Segmentation Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, no. 1 (2021): 550–557, https://doi.org/10.1109/TPAMI.2021.3062772.

55. S. Zhou, D. Chen, J. Pan, J. Shi, and J. Yang, "Adapt or Perish: Adaptive Sparse Transformer With Attentive Feature Refinement for Image Restoration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2024), 2952–2963.

56. J. Fan, F. Wang, H. Chu, X. Hu, Y. Cheng, and B. Gao, "Mlfnet: Multi-Level Fusion Network for Real-Time Semantic Segmentation of Autonomous Driving," *IEEE Transactions on Intelligent Vehicles* 8, no. 1 (2022): 756–767, https://doi.org/10.1109/tiv.2022.3176860.

57. J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2018), 7132–7141.

58. M. Noman, M. Fiaz, H. Cholakkal, S. Khan, and F. S. Khan, "ELGC-Net: Efficient Local–Global Context Aggregation for Remote Sensing Change Detection," *IEEE Transactions on Geoscience and Remote Sensing* 62 (2024): 1–11, https://doi.org/10.1109/tgrs.2024.3362914.

59. S. W. Zamir, A. Arora, S. Khan, et al., "Learning Enriched Features for Real Image Restoration and Enhancement," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV*, Vol. 16, (2020), 492–511.

60. Y. Xu, D. Wang, L. Zhang, and L. Zhang, "Dual Selective Fusion Transformer Network for Hyperspectral Image Classification," *Neural Networks* 187 (2025): 107311, https://doi.org/10.1016/j.neunet.2025.107311.

Supporting Information

Additional supporting information can be found online in the Supporting Information section.

Supporting Information S1: cit270129-sup-0001-suppl-data.pdf.