electronics

MDPI

Article

Pruning-Friendly RGB-T Semantic Segmentation for Real-Time Processing on Edge Devices

Jun Young Hwang [1,†], Youn Joo Lee [1,†], Ho Gi Jung [2] and Jae Kyu Suhr [1,*]

[1] Department of Artificial Intelligence and Robotics, Sejong University, 209 Neungdong-ro, Gwangjin-gu, Seoul 05006, Republic of Korea; hjy3451@naver.com (J.Y.H.); yjlee21@sejong.ac.kr (Y.J.L.)
[2] Department of Electronic Engineering, Korea National University of Transportation, 50 Daehak-ro, Chungju-si 27469, Republic of Korea; hogijung@ut.ac.kr
* Correspondence: jksuhr@sejong.ac.kr; Tel.: +82-2-3408-3481
† These authors contributed equally to this work.

Abstract

RGB-T semantic segmentation using thermal and RGB images simultaneously is actively being researched to robustly recognize the surrounding environment of vehicles regardless of challenging lighting and weather conditions. It is important for them to operate in real time on edge devices. As transformer-based approaches, which most recent RGB-T semantic segmentation studies belong to, are very difficult to perform on edge devices, this paper considers only CNN-based RGB-T semantic segmentation networks that can be performed on edge devices and operated in real time. Although EAEFNet shows the best performance among CNN-based networks on edge devices, its inference speed is too slow for real-time operation. Furthermore, even when channel pruning is applied, the speed improvement is minimal. The analysis of EAEFNet identifies the intermediate fusion of RGB and thermal features and the high complexity of the decoder as the main causes. To address these issues, this paper proposes a network using a ResNet encoder with an early-fused four-channel input and the U-Net decoder structure. To improve the decoder performance, bilinear upsampling is replaced with PixelShuffle. Additionally, mini Atrous Spatial Pyramid Pooling (ASPP) and Progressive Transposed Module (PTM) modules are applied. Since the Proposed Network is primarily composed of convolutional layers, channel pruning is confirmed to be effectively applicable. Consequently, channel pruning significantly improves inference speed, and enables real-time operation on the neural processing unit (NPU) of edge devices. The Proposed Network is evaluated using the MFNet dataset, one of the most widely used public datasets for RGB-T semantic segmentation. It is shown that the proposed method achieves a performance comparable to EAEFNet while operating at over 30 FPS on an embedded board equipped with the Qualcomm QCS6490 SoC.

Keywords: RGB-T semantic segmentation; edge-computing; channel pruning; embedded system

check for updates

Academic Editor: Ping-Feng Pai

Received: 24 June 2025
Revised: 24 August 2025
Accepted: 25 August 2025
Published: 27 August 2025

Citation: Hwang, J.Y.; Lee, Y.J.; Jung, H.G.; Suhr, J.K. Pruning-Friendly RGB-T Semantic Segmentation for Real-Time Processing on Edge Devices. *Electronics* **2025**, *14*, 3408. https://doi.org/10.3390/electronics14173408

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

To ensure stable autonomous driving, it is crucial to accurately perceive the surrounding environment of the vehicle. One of the most widely used approaches for camera-based perception of the vehicle's surroundings is semantic segmentation [1]. Most camera-based semantic segmentation methods rely on RGB images. However, their performance significantly degrades depending on lighting and weather conditions. To address this limitation, studies have explored the use of thermal images, which are robust to such conditions,

alongside RGB images for semantic segmentation. This approach is known as RGB-T semantic segmentation [2].

The methods of fusing images acquired from heterogeneous cameras can be largely classified into three types: early fusion, late fusion, and intermediate fusion [3–5]. In the field of RGB-T semantic segmentation, there are previous studies using early fusion and intermediate fusion. The method using early fusion is a method of inputting an image concatenating an RGB image and a thermal image into a network such as U-Net [6] and SegNet [7] showing good performance in semantic segmentation. In [2], the intermediate fusion approach was first introduced for RGB-T semantic segmentation. After that, most RGB-T semantic segmentation networks have adopted intermediate fusion. While recent RGB-T semantic segmentation networks have achieved high performance by adopting transformers [8,9], they face limitations by being hard to embed on edge devices for real-time processing. Therefore, this paper focuses on CNN-based networks that are capable of real-time performance on edge devices. Among the CNN-based RGB-T semantic segmentation networks published to date, EAEFNet [10] is a representative network showing state-of-the-art performance. EAEFNet uses a separate encoder with the same structure to independently process RGB and thermal images, respectively. Intermediate fusion is performed through a module that applies various attention techniques to the output of each encoder layer. This process is repeatedly performed for each layer, resulting in a total of five intermediate fusions. Then, the five features obtained through intermediate fusion are fed into the decoder, and a semantic segmentation result is calculated through a decoder being a complex arrangement of various modules such as Global Contextual Module (GCM) [11], mini Atrous Spatial Pyramid Pooling (ASPP) [10], and Progressive Transposed Module (PTM) [11].

Although EAEFNet [10] performs better than existing methods, it has a limitation that it operates very slowly when operating on an edge device equipped with a neural processing unit (NPU) due to its very large number of layers and complex structures. In addition, even when channel pruning [12], a representative method of network simplification, is applied, it is limited by minimal improvements to processing speed. This is because the various attention modules used by EAEFNet for intermediate fusion consume a lot of calculation time, and these modules are difficult to simplify by channel pruning. This is a major obstacle for RGB-T semantic segmentation to operate in the NPU of edge devices. Standardized and simple networks are preferred for real-time operation in NPU.

Therefore, this paper proposes an RGB-T semantic segmentation method that achieves performance comparable to EAEFNet while being more friendly for channel pruning, enabling real-time operation on edge devices. Most RGB-T semantic segmen-tation methods use a dual encoder-based intermediate fusion method that includes many layers. As a result, the effect of channel pruning on the operating speed is minimal. Consequently, they cannot be successfully embedded in an edge device or show a very slow operating speed. To solve these challenges, the proposed method adopts a single encoder-based early fusion method, which requires fewer computations to fuse the two different modalities. This reduces network complexity while enabling effective channel pruning. The proposed method uses a four-channel input concatenating RGB and thermal images for early fusion. Subsequently, a ResNet-based encoder [13] extracts features from the RGB-T four-channel input. The multi-resolution features extracted from each layer of the encoder are fed into the decoder. The decoder integrates these multi-resolution features based on PixelShuffle-based [14] upsampling, miniASPP, and PTM to produce semantic segmentation results. As such, the proposed method is designed to enable real-time semantic segmentation on edge devices, achieving optimal performance and processing speed for real-world applications.

In the experiments, the proposed method was trained and evaluated on the MFNet dataset [2], the most representative dataset for RGB-T semantic segmentation. Additionally, the method was embedded into an edge device equipped with the Qualcomm QCS6490 SoC for performance evaluation and comparison. As a result of the evaluation, it was confirmed that the proposed method showed about a 17 times faster processing speed than EAEFNet at the edge device, while showing only about 1.6%p performance degradation based on mIoU. In addition, when 90% of the channels were removed through channel pruning, EAEFNet showed 1.5 FPS at the edge device, but the proposed method showed 56.15 FPS in the same situation.

2. Related Works

Most semantic segmentation studies belong to either CNN-based or transformer-based approaches [8,9,15–19]. Among these, this literature review focuses on CNN-based methods that are supposed to operate in real time on an edge device. They are categorized into RGB image-based and RGB-T image-based methods, depending on the type of input image. Additionally, recent studies regarding computer vision on edge devices are briefly summarized.

2.1. RGB Image-Based Semantic Segmentation

Traditional semantic segmentation methods relied on handcrafted hyperparameters, such as threshold-based segmentation techniques and multi-class fuzzy support vector machines [20]. In recent years, CNN-based networks have significantly improved the performance of image semantic segmentation. Shelhamer et al. introduced the fully convolutional network (FCN) by employing deconvolution and skip connections, achieving accurate image segmentation with excellent performance [21]. Badrinarayanan et al. proposed SegNet, an encoder–decoder architecture based on an encoder framework, which enhanced positional information by utilizing max-pooling indices [7]. Romera et al. introduced a deep architecture that employed factorized convolutions and residual connections to balance accuracy and efficiency [22]. Yu et al. proposed a context module utilizing dilated convolutions to perform dense predictions with a large receptive field [23]. Chen et al. introduced ASPP to integrate multi-scale information and improve segmentation results [24]. Zhao et al. designed a pyramid pooling module to maximize the use of global contextual information through contextual aggregation across various scales [25]. Liu et al. developed a holistic boot-strap encoder to fully exploit multi-scale features enriched with semantic information, thereby enhancing the performance and efficiency of semantic segmentation [26]. Peng et al. addressed the computational overhead of traditional decoders by proposing a flexible lightweight decoder. They also introduced an attention fusion module to improve feature representation, achieving superior detection accuracy and speed [27]. While all of these methods have achieved remarkable performance in RGB semantic segmentation, they have fundamental limitations in that their performance significantly degrades under complex backgrounds and challenging lighting conditions, such as backlighting and darkness.

2.2. RGB-T Image-Based Semantic Segmentation

Infrared sensors can complement RGB images by providing thermal image information. This can significantly improve scene resolution even in environments with complex backgrounds or challenging lighting conditions, achieving better semantic segmentation performance. For this reason, RGB-T semantic segmentation, which performs semantic segmentation using both RGB and thermal images, has been actively researched. Methods fusing different modalities (in this case, RGB and thermal images) can be divided into

three categories: early fusion, late fusion, and intermediate fusion [3–5]. Early fusion combines multi-modal data using methods such as concatenation and addition, and uses them as input to a deep learning architecture. Late fusion applies separate networks to extract features from multi-modal data and fuses them just before the output. Intermediate fusion applies separate networks to multi-modal data and fuses them in the intermediate layers of the networks. Recently, the intermediate method has been actively researched. Ha et al. proposed an MFNet architecture using two symmetric encoders to extract features from multi-modal and announced the RGB-T urban scene dataset [2]. Ha et al. confirmed that using two symmetric encoders can achieve higher performance than feeding early fused data into the existing high-performance semantic segmentation through experiments. Since the MFNet architecture of Ha et al. was proposed, most RGB-T semantic segmentation networks have used intermediate fusion fusing features in the middle using two symmetric encoders. Sun et al. proposed an RTFNet architecture using a two-step fusion strategy to extract RGB-T image information and improve segmentation performance using DenseNet architecture as the backbone of the encoder [28]. Zhang et al. proposed an ABMDRNet architecture that focuses on the differences between various imaging modalities and adaptively selects features across multiple channels using a channel-weighted fusion strategy [29]. Zhou et al. proposed a GMNet architecture in which two fusion strategies (i.e., shallow feature fusion module and deep feature fusion module) are applied and the network is optimized by loss functions defined as semantic, binary, and edge features [30]. Liang et al. proposed an EAEFNet [10] architecture that uses two symmetric ResNet [13] architectures as the backbone of the encoder. It highlights the features for each layer of the two backbones using several attention modules and then fuses them as input for the next layer. The EAEFNet architecture proposed by Liang et al. currently achieves the highest performance among CNN-based RGB-T semantic segmentation methods.

2.3. Computer Vision on Edge Devices

Existing computer vision systems operate by acquiring information through various devices, transmitting it to a central server, and then performing computer vision tasks [31]. This approach introduces several issues. First, when uploading the acquired data to the central server, a large amount of network bandwidth is required, and this can result in significant costs, especially for high-resolution data. In addition, if communication becomes unstable during the transmission process, response time and communication time may be prolonged, and transmission could stop and may not work properly. Furthermore, there is a risk of personal information leakage during the transmission of acquired images to the server, which could potentially be exploited for criminal activities.

However, if computer vision is implemented on edge devices, the acquired images do not need to be transmitted to a central server and can be processed locally within the edge device itself. This can address issues such as costs, time delays, and the risk of personal information leakage. Representative fields include self-driving cars and CCTV. In the case of autonomous vehicles, they must operate in real time as they must quickly recognize environments and control their motion [32]. In the case of CCTV, they must operate in real time because they need to quickly recognize situations such as terrorism, violence, or natural disasters that require immediate action [33].

For this reason, deep learning networks are currently being developed into a form that can be implemented in edge devices by being lightweight and efficiently designed. Tools such as Google's TensorFlow Lite [34] and ONNX Runtime [35] have also been developed. In addition, various hardware for deep learning such as Qualcomm's Snapdragon [36], NVIDIA's Jetson Nano [37], and Intel's Movidius [38] have been developed,

with chipsets that balance energy efficiency and performance to enable real-time computing on edge devices.

Previous studies have implemented object detection and segmentation on edge devices and applied techniques like network slimming to achieve real-time performance on edge devices [39–42]. Lee et al. demonstrated a system that processes RGB and thermal images using DeepLabv3+ for flood monitoring, running in real time on the Qualcomm QCS610 [39]. Choi et al. implemented a system for traffic monitoring, applying YOLOv4 and DeepSORT on traffic surveillance camera footage to detect and track traffic users, also running in real time on the Qualcomm QCS605 [40,42]. Lee et al. developed a system for monitoring wildfires and floods using an unmanned aerial vehicle (UAV) with DeepLabv3+, running in real time on the Qualcomm QCS610 [41].

3. Proposed Method

3.1. Analysis of EAEFNet

This paper aims to perform real-time RGB-T semantic segmentation on edge devices. To this end, EAEFNet [10], the most accurate CNN-based network composed of operations suitable for deployment on embedded systems, is analyzed. Figure 1 illustrates the architecture of EAEFNet. The gray area on the left of Figure 1 is the encoder architecture, and the blue area on the right is the decoder architecture. EAEFNet uses separate encoders for RGB and thermal images, and utilizes the EAEF module, incorporating techniques such as channel attention and spatial attention to fuse the layer output features of each encoder. All outputs from the EAEF modules are fed into the decoder, which consists of various modules such as GCM, miniASPP, and PTM. In the GCM module, convolution operations of various kernel sizes are applied to each EAEF module, and their outputs are combined into an output feature map. Many operations are performed in the process of combining the output with the features of the previous layer, and the resultant structure becomes very complex. As a result, although the EAEFNet decoder follows a structure similar to that of U-Net [6], it becomes significantly more complex due to the inclusion of multiple additional modules and the use of more than 500 layers. In the decoder, UP $\times$ k refers to an upsampling operation that increases the spatial resolution of the feature map by k times horizontally and vertically, and this operation is simply implemented using the "torch.nn.Upsample (PyTorch v1.10.1)" module.

This paper confirms that when EAEFNet is embedded on an edge device, it cannot operate in real time. Two major causes are identified. First, to identify the parts interfering with real-time processing, the processing time of each layer is measured by operating EAEFNet on edge devices repeatedly. Next, to identify the parts unsuitable for channel pruning, channel pruning is applied to EAEFNet and its effect is measured. As a result, the EAEF module is identified as the main obstacle for channel pruning. In other words, the processing time of the EAEF module does not change before and after channel pruning. The EAEF module is analyzed and takes a lot of processing time, because it performs attention based on matrix operations such as slicing, matrix multiplication, and element-wise multiplication rather than convolution operations. Consequently, it can be confirmed that although EAEFNet is capable of performing on edge devices, it is not suitable for real-time operation.

To overcome the limitations of EAEFNet, this paper proposes a network consisting of layers suitable for edge devices and channel pruning, and with a simple structure composed of a small number of layers. The Proposed Network can operate in real time on edge devices and shows similar performance to EAEFNet. In addition, as shown in Figure 1, it is confirmed that the complex connection structure included in the decoder of EAEFNet is very similar to that of U-Net. Based on this, this paper proposes that the EAEFNet

decoder can be replaced with that of U-Net, and the experimental results confirmed that the performance difference between the two networks was small, but the inference speed increased by approximately three times.

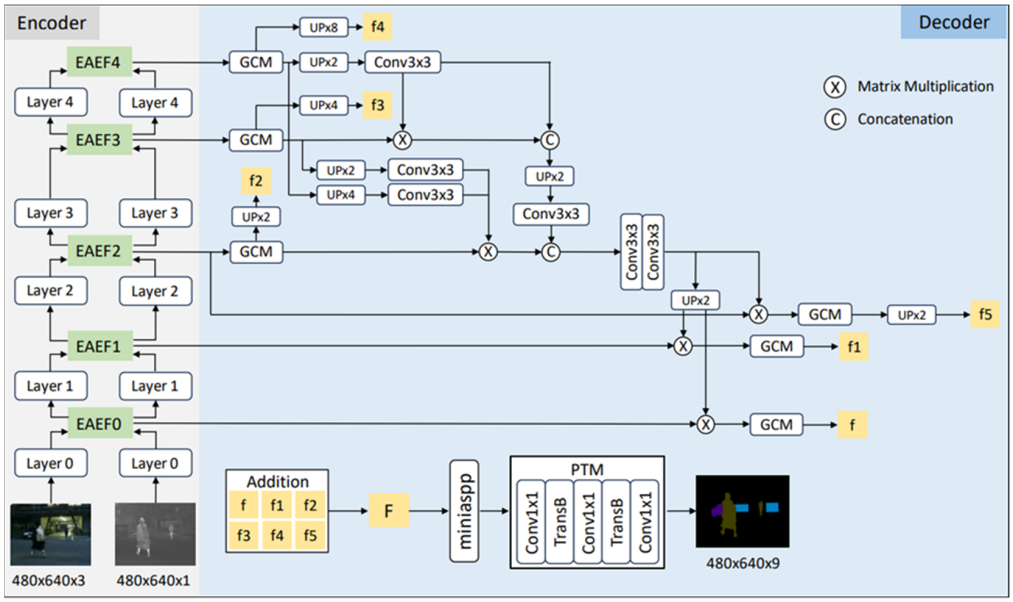


Figure 1. Architecture of EAEFNet. EAEFNet consists of a dual encoder for RGB and thermal images, EAEF modules for RGB-T feature fusion, and a decoder with multiple modules.

3.2. Proposed Network

In Section 3.1, two major causes are identified that prevent EAEFNet [10] from operating in real time on edge devices. To analyze the impact of the first cause (i.e., the EAEF modules) on inference speed, after removing each EAEF module, the changes in both performance and inference time are measured. The experimental results show that removing each EAEF module has little impact on performance. This indicates that the contribution of EAEF modules in the encoder to overall performance is minor. However, when all EAEF modules are removed, performance degradation occurs. This was because if all EAEF modules are removed, RGB and thermal images are not fused at all, and consequently, the network uses only RGB inputs (refer to Table 1 in Section 4 for details).

Based on these results, one candidate network composed of a single encoder and EAEFNet decoder using a 4-channel input obtained by early fusion of RGB and thermal images is evaluated. The results show that there is almost no difference in performance between the candidate network and the original EAEFNet. Furthermore, the inference speed on edge devices is improved by approximately four times (refer to Table 2 of Section 4 for the details).

The second cause is that the EAEFNet decoder has a large number of layers and a highly complex computation graph. To address this problem, the possibility of replacing the EAEFNet decoder with the U-Net decoder is evaluated. After changing the EAEFNet decoder to the U-Net decoder, the changes in both performance and inference time are measured. Experimental results confirmed that the performance difference between the two networks is small, but the improved inference speed is faster than the original by approximately three times (refer to Table 3 in Section 4 for details). This network is composed of a single encoder using early fused RGB and thermal images as input and a decoder based on the U-Net architecture, called as "Base Network" in this paper.

Figure 2 shows the architecture of the Base Network. The Base Network has a single encoder based on ResNet50 [13], and the encoder uses 4-channel inputs. The decoder is based on the U-Net [10] decoder and has been adapted to correspond to the ResNet50-

based encoder. Compared to EAEFNet, the Base Network shows a performance drop of approximately 2.8%p, but achieves about a 15 times faster inference speed (refer to Table 3 of Section 4 for details).

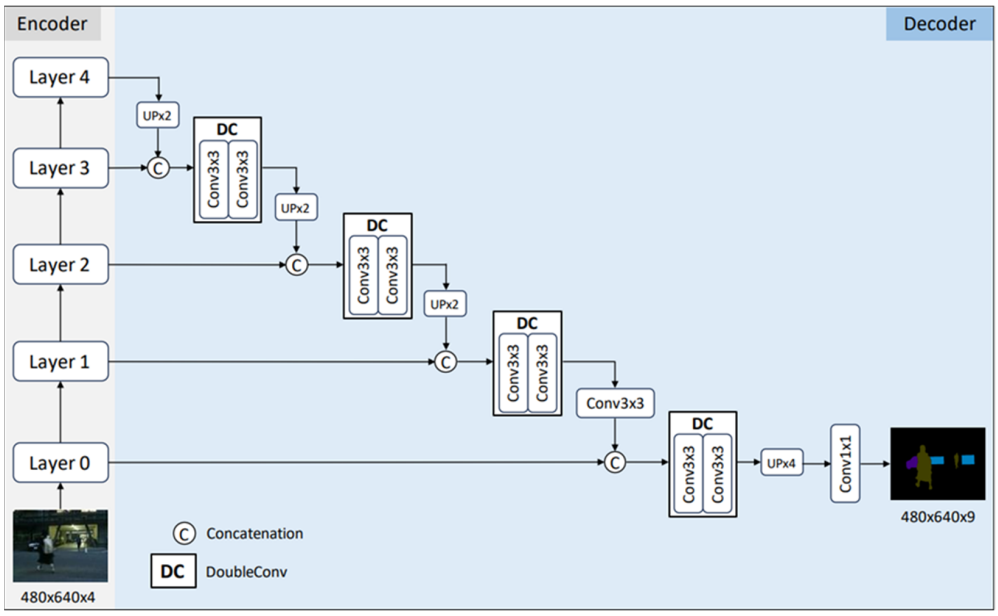


Figure 2. Architecture of Base Network. Base Network consists of a single encoder based on ResNet50 and a decoder based on U-Net.

To minimize performance degradation of the Base Network while maintaining a high inference speed, the decoder is modified by incorporating the PixelShuffle, miniASPP, and PTM modules. The UP $\times$ 2 layers of the Base Network are changed to PixelShuffle modules, and the UP $\times$ 4 layer is changed to miniASPP and PTM modules. In this way, the Proposed Network is finally completed. Figure 3 shows the architecture of the Proposed Network.

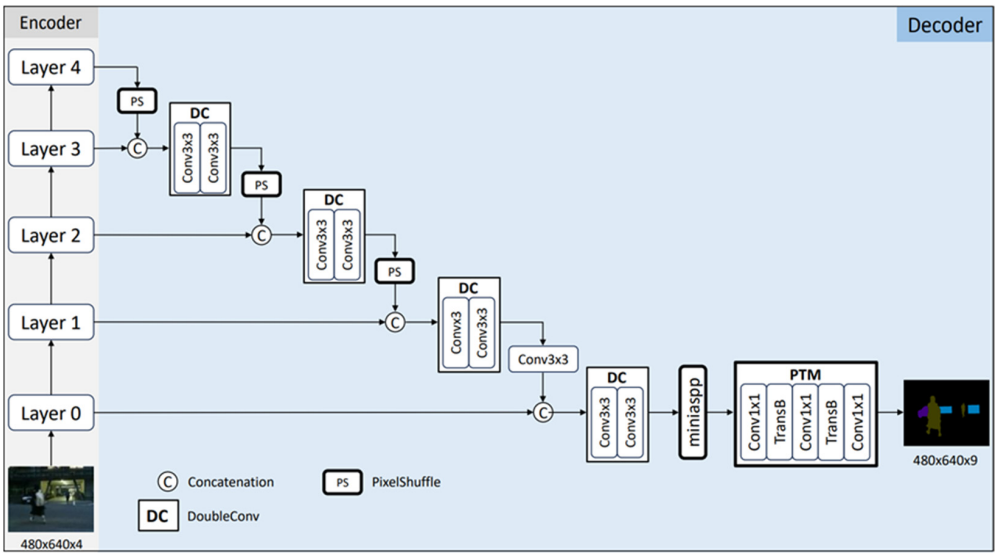


Figure 3. Architecture of Proposed Network. Proposed Network shares the same encoder as the Base Network. However, its decoder consists of PixelShuffle, miniASPP, and PTM modules, replacing the bilinear upsampling operations.

There is a possibility of information loss when bilinear upsampling operation is applied to feature maps. To minimize information loss and improve performance, the Proposed Network replaces the UP $\times$ 2 stage of the Base Network with PixelShuffle [14]. PixelShuffle

is one of the upsampling techniques mainly used in image super-resolution. Figure 4 shows the operation principle of the PixelShuffle. This method performs upsampling by rearranging pixels from the channel dimension into the spatial dimensions (width and height). This upsampling method effectively preserves spatial information, which helps improve image quality. In EAEFNet, upsampling is performed using both complex GCM and standard upsampling layers. The decoder of the Proposed Network is designed by using only the PixelShuffle module, which is more suitable for embedding.

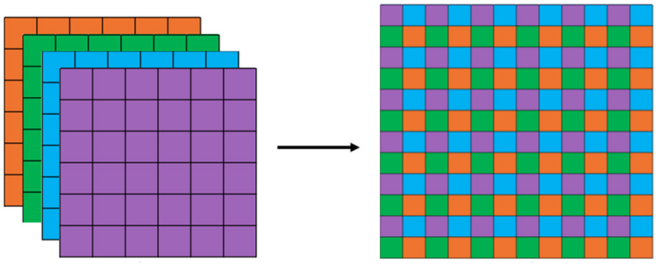


Figure 4. Result of applying PixelShuffle to 4-channel feature map.

While the Base Network uses UP $\times$ 4 in the decoder to restore the original resolution, the Proposed Network adopts the miniASPP [10] and PTM [28] modules used in the EAEFNet decoder. Figure 5 shows the structure of the miniASPP module. MiniASPP is a simplified version of the original ASPP. The original ASPP applies 1 $\times$ 1 convolution, 3 $\times$ 3 convolutions with dilation rates of 6, 12, and 18, and image pooling to the feature map, and then concatenates the resulting features. However, miniASPP applies only 3 $\times$ 3 convolutions with dilation rates of 6, 12, and 18, and combines them using element-wise addition. This makes it possible to effectively capture multi-scale information by utilizing receptive fields of various sizes, which is advantageous for reducing information loss. After the miniASPP, the PTM module, shown in Figure 6, is applied to perform upsampling. This module performs upsampling through two transposed convolutions and gradually performs upsampling by applying the convolution operation before the transposed convolution. This enables effective restoration of high-resolution features, which is advantageous for preserving fine details.

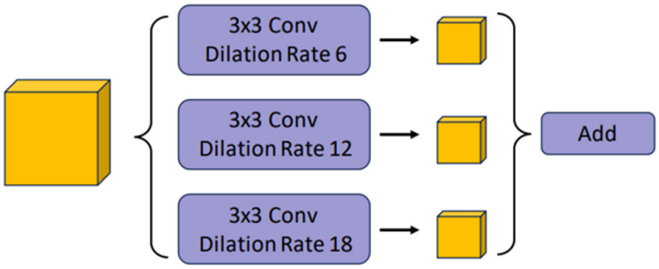


Figure 5. Structure of miniASPP module.

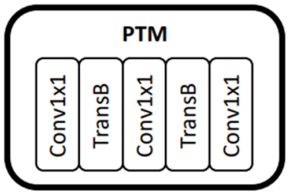


Figure 6. Structure of PTM module.

From the perspective of the number of layers, there is a substantial difference between EAEFNet and the Proposed Network. The encoder of EAEFNet consists of two encoders,

each with 125 layers, along with five EAEF modules, each comprising 11 layers. In contrast, the encoder of the Proposed Network contains a single encoder with 124 layers. Regarding the decoder, EAEFNet incorporates multiple complex modules, such as GCM, resulting in a total layer count of 813. In contrast, the decoder of the Proposed Network contains only 73 layers. Overall, by removing modules that slow down EAEFNet's operation on edge devices, the Proposed Network reduces the total layer count by 921. Furthermore, it primarily consists of convolutional layers (see Figure 3 in Section 3 for details). This structure enables faster inference on edge devices and facilitates more effective channel pruning.

3.3. Network Simplification and Embedding

For embedding on an edge device, the Proposed Network is simplified by channel pruning and quantization. Channel pruning is one of the representative network simplification techniques for improving computational efficiency. It reduces the computational cost and compresses the network by removing less important channels in convolutional neural networks. Consequently, as the inference speed becomes faster and memory usage is reduced, it makes the network more suitable for real-time processing on edge devices. In channel pruning, there are various methods for determining which channel will be removed, in other words, the importance of channels in convolutional layers. Representative methods include random selection, smaller L1 norm-based, and a smaller scaling factor of the batch normalization layer-based methods. Among these methods, random selection is employed in this study, considering previous search results [43]. Random selection is a method in which a specified percentage or number of channels is randomly selected and removed. Unlike other methods, it does not require additional processes such as sparsity training. To implement this approach, Torch Pruning [10], a general-purpose tool that supports structured pruning for various deep learning models, is utilized. After channel pruning, fine-tuning is essential to recover from performance degradation. It enables the network to adapt to the pruned architecture and optimize its parameters, thus restoring its performance.

Next, the simplified network is embedded into Qualcomm's QCS6490 chip. QCS6490 is a system on a chip (SoC) that includes CPU, GPU, and DSP, enabling high performance at low power [44]. This paper evaluates the networks only on DSP, that is, a Hexagon 770 DSP with dual Hexagon Vector eXtensions (HVX), a Hexagon Co-processor, and a Hexagon Tensor Accelerator for on-device machine learning and edge computing. Qualcomm provides a software development kit (SDK) called the Snapdragon Neural Processing Engine (SNPE) v2.14, which allows executing Deep Neural Network (DNN) models on Qualcomm hardware accelerators. SNPE provides various tools for model conversion, quantization, and runtime. The embedding process is performed by using these tools, as shown in Figure 7. In this process, the first step is a model conversion. Generally, to deploy and operate DNN networks on edge devices, the network format needs to be transformed into a format suitable for the embedded board environment. This is because trained networks are created using various deep learning frameworks such as TensorFlow, Caffe, or PyTorch, depending on the development environment. Therefore, the simplified network must be converted into a format compatible with the QCS6490 SoC. When converting the network, the network file format is transformed into a Deep Learning Container (DLC) file, which is a format compatible with Qualcomm chips [44]. In this paper, EAEFNet and the Proposed Network are trained using the PyTorch framework, and the conversion API of the SNPE converts the PyTorch network file into a DLC file. The second step is quantization. Since the DSP supports only 8-bit integer operations, it is essential to quantize a model with a floating point (e.g., FP32) into one with a low-integer format (e.g., INT8). For quantization, post-training quantization (PTQ) is utilized in SNPE. PTQ is a model

compression method applied to a trained model without a retraining process. This process often uses a calibration dataset (a small subset of the training data) to estimate optimal quantization parameters. The final step involves running a quantized DLC model on the DSP of the QCS6490 at runtime. Optimized and quantized DLC models obtained on the desktop are transferred to the board equipped with QCS6490 via an Android Debug Bridge (ADB) terminal connection, and then they are executed on the DSP.

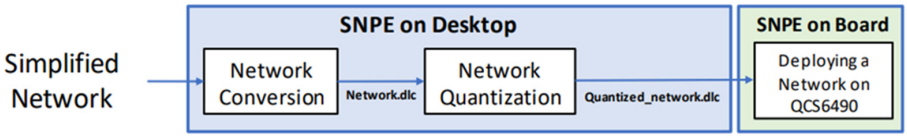


Figure 7. Process of embedding a simplified network in QCS6490.

4. Experiments

4.1. Experimental Setup

The Proposed Network is trained and evaluated based on the MFNet dataset [2], which is the most widely used public dataset for RGB-T semantic segmentation. The infrared camera used to acquire the dataset captures an image in the 8–14 μm spectral range, providing thermal images in the Long-Wavelength Infrared (LWIR) domain. Both the RGB and thermal images have a resolution of 480 × 640 pixels. The dataset consists of eight labeled classes of obstacles encountered while driving on the road: car, person, bike, curb, car stop, guardrail, color cone, and bump, and one unlabeled background class. The dataset consists of a total of 1569 images, including 820 daytime and 749 nighttime images. These are divided into training, validation, and test sets. The training data consists of 784 images in total, including 50% daytime and nighttime images, while the validation and evaluation data consist of 392 and 393 images, including 25% daytime and nighttime images, respectively.

In this paper, the networks are trained using PyTorch v1.10.1 on a system equipped with an NVIDIA-TITAN-RTX 24 GB GPU, an Intel Core i9-10900X CPU, and 64 GB RAM. For training data augmentation, random flip and random crop techniques are used. The networks are trained for a total of 100 epochs until convergence. The stochastic gradient descent (SGD) [45] optimization solver is used for training. The initial learning rate is set to 0.02, and momentum and weight decay are set to 0.9 and 0.0005, respectively. The batch size is set to 3, and ExponentialLR is applied to decrease the learning rate gradually. The four networks evaluated in the experiments are trained five times each, and the best result among the five results is reported.

Figure 8 shows the Cameleon8 board [46] from WITHROBOT, Seoul, Republic of Korea), which is used in the experiments. The board is equipped with Qualcomm's QCS6490 SoC, San Diego, CA, USA. SNPE version 2.14 is used for experiments. The inference speed on the edge device mentioned in this paper is measured using this board.



Figure 8. Chameleon8 board from WITHROBOT, equipped with Qualcomm's QCS6490 SoC.

4.2. Experiments for Analyzing EAEFNet

The performance impact of the EAEF module is analyzed. The EAEF module takes up a large amount of computation and is not efficiently simplified by channel pruning. Table 1 shows the mIoU and the inference speed on the edge device when each EAEF module is removed from EAEFNet. Through this table, it is confirmed that the performance degradation is minimal when the five EAEF modules are individually removed. When all EAEF modules are removed, performance degradation occurs by about 6%p. This suggests that removing all EAEF modules from the EAEFNet architecture would disable the fusion of RGB and thermal images, causing the network to operate based solely on RGB input.

Additional experiment compares this configuration with a single encoder network that takes an early-fused 4-channel input. Table 2 shows the mIoU and inference speed on the edge device of EAEFNet and a network combining a single encoder using an early-fused 4-channel input and the EAEFNet decoder. Through this table, it is confirmed that the performance difference between the two architectures is minimal at only 0.9%p, while the combination of the single encoder and EAEFNet decoder achieved an approximately four times faster inference speed. In summary, these findings indicate that although the EAEF module incurs considerable computational cost, its contribution to overall performance is relatively limited. This suggests that using a single encoder is advantageous for edge devices where inference speed is a priority.

Table 1. Performance evaluation when removing each EAEF module from EAEFNet individually.

Network	mIoU (%)	FPS on DSP
EAEFNet baseline	57.8	0.6
EAEFNet with EAEF module 0 removed	56.6	1.5
EAEFNet with EAEF module 1 removed	57.2	1.6
EAEFNet with EAEF module 2 removed	57.4	1.6
EAEFNet with EAEF module 3 removed	57.2	1.5
EAEFNet with EAEF module 4 removed	57.8	1.7
EAEFNet with five EAEF modules removed [1]	51.4	3.0

[1] It means that the network has no fusion module (EAEF module).

Table 2. Performance comparison between original EAEFNet and the model whose encoder part is replaced by 4-channel input single encoder.

Network	mIoU (%)	FPS on DSP
EAEFNet baseline	57.8	0.6
4-ch input single encoder + EAEFNet decoder	56.9	2.8

After the encoder is fixed as the single encoder, simplification of the decoder, consisting of numerous layers and a complex structure, is experimented with. Table 3 compares the mIoU and inference speed on the edge device of the EAEFNet decoder and Unet decoder. In this table, it is confirmed that the network using the U-Net decoder (Base Network of Figure 2) shows a 1.8%p lower mIoU compared to the network using the EAEFNet decoder, but the inference speed on the edge device is approximately 3.6 times faster.

Table 3. Performance comparison between the model with 4-channel input single encoder and EAEFNet decoder and the model with 4-channel input single encoder and U-Net decoder.

Network	mIoU (%)	FPS on DSP
4-ch input single encoder + EAEFNet decoder	56.9	2.8
4-ch input single encoder + U-Net decoder [1]	55.1	10.3

[1] This is Base Network in this paper.

4.3. Experiments for Building Proposed Network

The Base Network uses the U-Net decoder, whose performance is slightly lower than that of the EAEFNet decoder, as shown in Table 3. To improve this point, the Proposed Network is designed by modifying the decoder of the Base Network. The Proposed Network decoder includes PixelShuffle, miniASPP, and PTM modules, unlike the U-Net decoder. The Proposed Network consists of a 4-channel input single encoder based on ResNet50 and a modified U-Net decoder. Table 4 shows an ablation study on the impact of PixelShuffle, miniASPP, and PTM modules on the performance of the Proposed Network. The ablation study results indicate that the mIoU increases when each module is added individually. In addition, it is confirmed that when all three modules are applied, the mIoU is 2.5 percentage points higher than the Base Network. Based on these results, the Proposed Network is constructed by adding the corresponding modules to the Base Network. Additionally, to improve inference speed on the edge device, the Simplified Proposed Network is also constructed by reducing the convolution kernel size from 3×3 to 1×1 in the DC module of the Proposed Network decoder. The final network we propose for edge computing is the Simplified Proposed Network.

Table 4. Ablation study on the impact of PixelShuffle, miniASPP, and PTM modules on the performance of the Proposed Network.

PixelShuffle	miniASPP	PTM	mIoU (%)
			55.1
√			56.5
	√		55.9
		√	55.2
√	√	√	57.6

Table 5 presents performance comparisons among the Base Network, the Proposed Network, and the Simplified Proposed Network. In this table, mIoU and model size are measured on desktop, while inference speed (FPS) is measured on DSP of the QCS6490 chip. As shown in Table 5, the Proposed Network achieves the highest performance with an mIoU of 57.6%. In contrast, the Simplified Proposed Network attained an mIoU of 56.6%—comparable to that of the Proposed Network, while having the smallest model size and the highest inference speed.

Table 5. Performance Comparisons among Base Network, Proposed Network, and Simplified Proposed Network. (Base Network = Single encoder + U-Net decoder, Proposed Network = Single encoder + Modified U-Net decoder, Simplified Proposed Network = Single encoder + Modified U-Net decoder with 1×1 convolution layers).

Network	mIoU (%)	Model Size (MB)	FPS on DSP
Base Network	55.1 (+0.0) [1]	288.9	10.3 (+0.0)
Proposed Network	**57.6 (+2.5)**	222.8	12.5 (+2.2)
Simplified Proposed Network	56.6 (+1.5)	**112.1** [2]	**20.9 (10.6)**

[1] The values in parentheses indicate the relative performance changes of the other two networks compared to the Base Network. [2] The values in bold indicate the best results.

It should be noted that the Proposed Network, which adds modules to the Base Network, has a faster inference speed. This is because the bilinear upsampling process in the Base Network takes a relatively long time. On the other hand, the Proposed Network improved the overall inference speed by performing the upsampling operation more efficiently using the PixelShuffle and PTM modules.

Table 6 presents comparisons of mIoU and inference speed on the edge device of EAEFNet, the Base Network, and the Simplified Proposed Network. In this table, EAEFNet achieves the highest performance with an mIoU of 57.6%, but it shows a very slow inference speed of 1.2 FPS. The Base Network shows a 2.6-percentage-point decrease in performance compared to EAEFNet, but achieves a significantly faster inference speed of 10.3 FPS. The Simplified Proposed Network shows a 1.6-percentage-point decrease in performance compared to EAEFNet, but achieves a significantly faster inference speed of 20.9 FPS and an inference time of 47.6 milliseconds (ms). Memory size is also a critical factor affecting memory usage on edge devices. In terms of model size, EAEFNet is the largest at 113.6 MB, whereas the Simplified Proposed Network is the smallest at 21.8 MB. Based on these results, the Simplified Proposed Network is the most suitable for deployment on edge devices.

Table 6. Performance comparisons among Base Network, Proposed Network, Simplified Proposed Network on QCS6490. (EAEFNet =Dual encoder + EAEFNet decoder, Base Network = Single encoder + U-Net decoder, Simplified Proposed Network = Single encoder + Modified U-Net decoder with 1 × 1 convolution layers).

Network	mIoU (%)	Model Size (MB)	FPS	Inference Time (ms)
EAEFNet	**57.6 (+0.0)**	113.6	1.2	826.4
Base Network	55.0 (−2.6)	72.2	10.3	96.5
Simplified Proposed Network	56.0 (−1.6)	**21.8**	**20.9**	**47.6** [1]

[1] The values in bold indicate the best results.

Figures 9 and 10 present the qualitative evaluation results of EAEFNet, the Base Network, and the Simplified Proposed Network. Each figure is organized as follows: the first and second rows show RGB and thermal input images, respectively; the third row shows the ground truth; and the fourth to sixth rows display the segmentation results from EAEFNet, the Base Network, and the Simplified Proposed Network. The first to third columns present cases with daytime data, while the fourth to sixth columns correspond to nighttime data. Figure 9 illustrates examples of relatively accurate segmentation results from the three networks. As shown in this figure, objects are well segmented regardless of whether the images were captured during the day or at night. Large objects, such as "car," are accurately segmented by all three networks. Smaller objects, including "person," "bike," "car stop," and "curb," are also well segmented by three networks. Figure 10 shows examples where the three networks produce different results. For inputs (a), (b), and (f), EAEFNet mistakenly segments areas without bumps as "bump." In contrast, neither the Base Network nor the Simplified Proposed Network exhibit this issue for the same inputs. In fact, both networks produce segmentation results that are more consistent with the ground truth than EAEFNet. However, for input (e), the Simplified Proposed Network shows less precise segmentation of the "color cone" and "curb" compared to the other two methods. Although the Simplified Proposed Network segments small objects, such as "person," relatively well, it generally struggles to accurately segment objects like "curb," "color cone," and "car stop." As shown in the mIoU results of Table 6, it is confirmed that the Simplified Proposed Network and EAEFNet produced similar outcomes in most cases.

Finally, Table 7 compares the proposed methods with existing methods on the MFNet dataset, using benchmark results reported in the literature. We compare our methods with MFNet [2], GMNet [30], ABMDRNet [29], FEANet [28], EAEFNet [10], and the Base Network. Since model complexity varies depending on the backbone, the backbone used for each network is indicated in parentheses after its name for a fair comparison. As shown in the table, EAEFNet achieves the highest semantic segmentation accuracy, followed by the proposed method, which also demonstrates competitive performance.

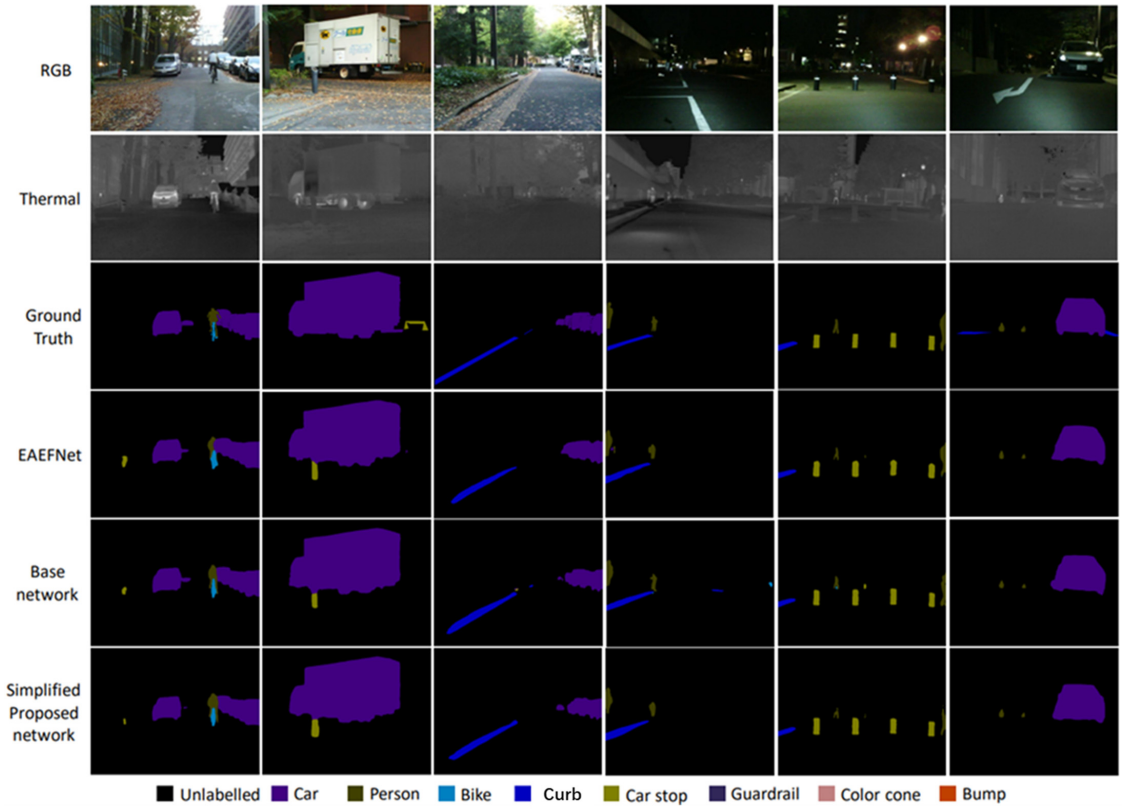


Figure 9. Examples of cases where EAEFNet, Base Network, and Simplified Proposed Network yield similarly good results.

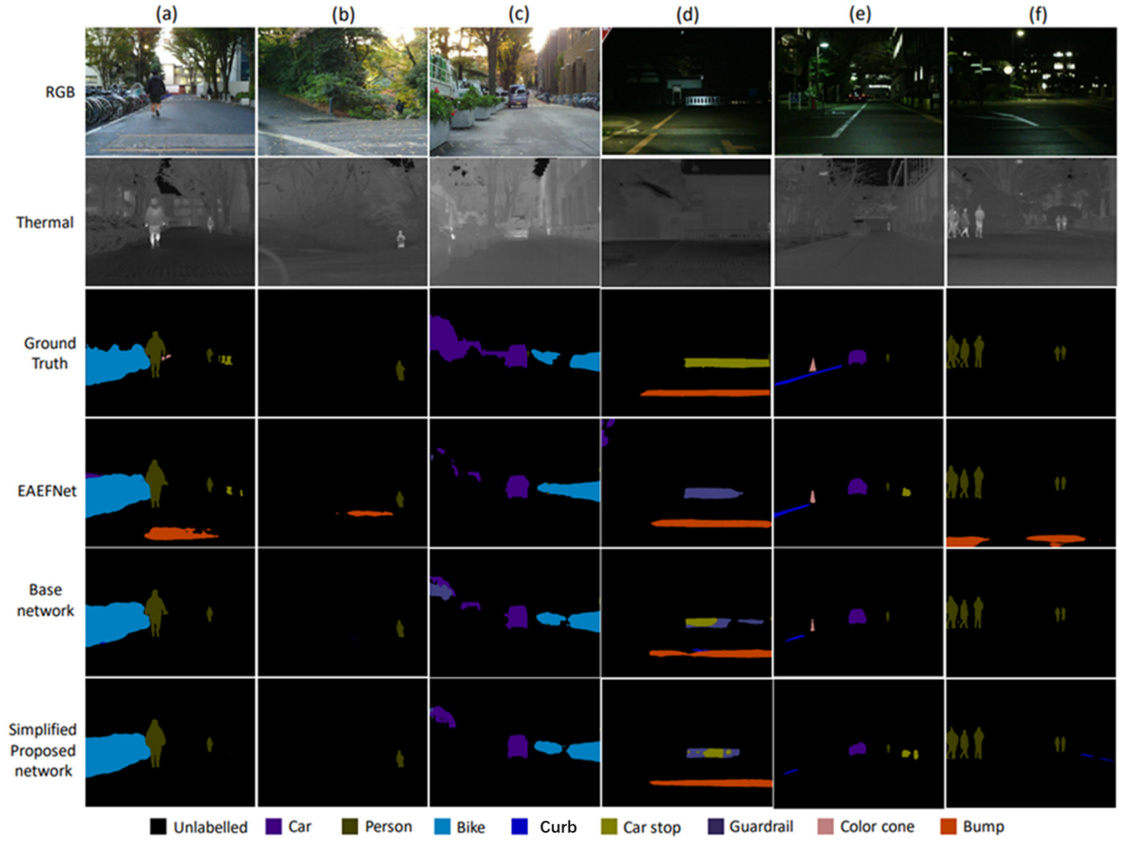


Figure 10. Examples of cases where EAEFNet, Base Network, and Simplified Proposed Network yield slightly different results: (**a**–**c**) for daytime input images; (**d**–**f**) for nighttime input images.

Table 7. Performance comparisons on MFNet dataset. The best result is shown in bold font.

Network	mIoU (%)
MFNet [2]	39.7
GMNet (ResNet50) [30]	57.3
ABMDRNet (ResNet50) [29]	54.8
FEANet (ResNet152) [28]	55.3
EAEFNet (ResNet50) [10]	**57.8**
Base Network (ResNet50)	55.1
Proposed Network (ResNet50)	57.6
Simplified Proposed Network (ResNet50)	56.6

4.4. Experiments for Channel Pruning

In this paper, channel pruning is applied to EAEFNet, the Base Network, and the Simplified Proposed Network to confirm how the mIoU, model size, and inference speed change. The pruning ratio is determined by tuning the speed-up parameter in Torch Pruning [42]. The speed-up parameter determines the target reduction ratio of the network's computational cost (FLOPs). For instance, when the speed-up parameter is set to 2, the network's FLOPs are reduced by half, and when set to 10, the FLOPs are reduced to one-tenth. In this study, channel pruning is performed while changing the speed-up values to 2, 4, 7, and 10.

Tables 8–10 show the changes in mIoU, model size, and inference speed of EAEFNet, the Base Network, and the Simplified Proposed Network, respectively, according to different speed-up parameter values. As shown in Table 8, when the speed-up value for EAEFNet increased from 1 to 10, the mIoU dropped from 57.6% to 42.5% and the model size was reduced from 113.6 MB to 15.3 MB, while the FPS showed minimal change, increasing only slightly from 1.2 to 1.5. This is because non-convolutional layers are the main cause of the reduced inference speed in EAEFNet. Therefore, even if channel pruning is applied to reduce the number of channels in the convolutional layers, the overall speed improvement is not significant.

Table 8. Performance of EAEFNet at different speed-up parameters of channel pruning.

Speed Up	mIoU (%)	Model Size (MB)	FPS	Inference Time (ms)
1× (baseline)	57.6 (−0.0)	113.6	1.2 (+0.0)	826.4
2×	54.9 (−2.7)	64.4	1.2 (+0.0)	819.7
4×	52.3 (−3.5)	42.9	1.2 (+0.0)	806.5
7×	48.8 (−8.8)	23.2	1.4 (+0.2)	694.4
10×	42.5 (−15.1)	15.3	1.5 (+0.3)	636.9

In Table 9, the Base Network exhibits a decrease in mIoU from 55.0% to 45.0% and a reduction in model size from 72.2 MB to 6.7 MB, while also showing an increase in FPS from 10.3 to 37.2 as the speed-up value changes. In Table 10, the Simplified Proposed Network shows a decrease in mIoU from 56.0% to 40.7% and a decrease in model size from 28.1 MB to 3.9 MB, while also showing an increase in FPS from 20.9 to 56.1 under the same conditions. The reason why FPS significantly increased when channel pruning was applied to the Base Network and Simplified Proposed Network is that most of the layers in the network consist of convolutional layers, maximizing the effect of channel pruning.

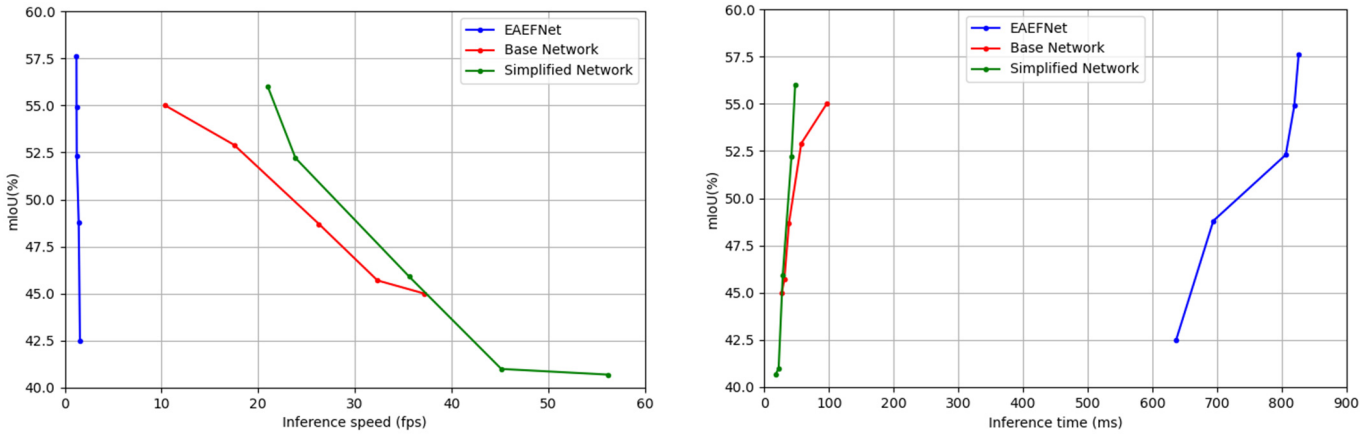
Table 9. Performance of Base Network at different speed-up parameters of channel pruning.

Speed Up	mIoU (%)	Model Size (MB)	FPS	Inference Time (ms)
1× (baseline)	55.0 (−0.0)	72.2	10.3 (+0.0)	96.5
2×	52.9 (−2.1)	35.3	17.5 (+7.2)	57.1
4×	48.7 (−6.3)	18.0	26.2 (+15.9)	38.1
7×	45.7 (−9.3)	9.3	32.3 (+22.0)	30.9
10×	45.0 (−10.0)	6.7	37.2 (+26.9)	26.9

Table 10. Performance of Simplified Proposed Network at different speed-up parameters of channel pruning.

Speed Up	mIoU (%)	Model Size (MB)	FPS	Inference Time (ms)
1× (baseline)	56.0 (−0.0)	28.1	20.9 (+0.0)	47.6
2×	52.2 (−3.8)	15.3	23.8 (+2.9)	41.9
4×	45.9 (−10.1)	8.0	35.6 (+14.7)	28.4
7×	41.0 (−15.0)	4.8	45.1 (+24.2)	22.1
10×	40.7 (−15.3)	3.9	56.1 (+35.2)	17.8

Figure 11 shows the results from Tables 8–10 in graphical form. Figure 11a compares the relationship between inference speed (FPS) and segmentation performance (mIoU) for the three networks, and Figure 11b compares the relationship between inference time and mIoU for the three networks. It is important to note that while the Simplified Proposed Network has a slightly lower mIoU than EAEFNet, it can satisfy the real-time requirements for automotive applications in terms of model size and inference time.



(**a**) (**b**)

Figure 11. Performance comparison graphs for EAEFNet, Base Network, and Simplified Proposed Network on MFNet dataset: (**a**) semantic segmentation performance (mIoU) versus inference speed (fps); (**b**) semantic segmentation performance (mIoU) versus inference time (ms).

Figure 12 shows the qualitative evaluation results of the Simplified Proposed Network at over 30 FPS, which is a considered real-time operation for automotive applications. Although the mIoU decreased slightly and some detailed differences are observed in the qualitative evaluation results, it is confirmed that these did not have a significant impact on the overall performance. In conclusion, since the Simplified Proposed Network can successfully detect the contours of various obstacles at 30 FPS on embedded systems, it is expected to be effectively utilized in real-time automotive applications.

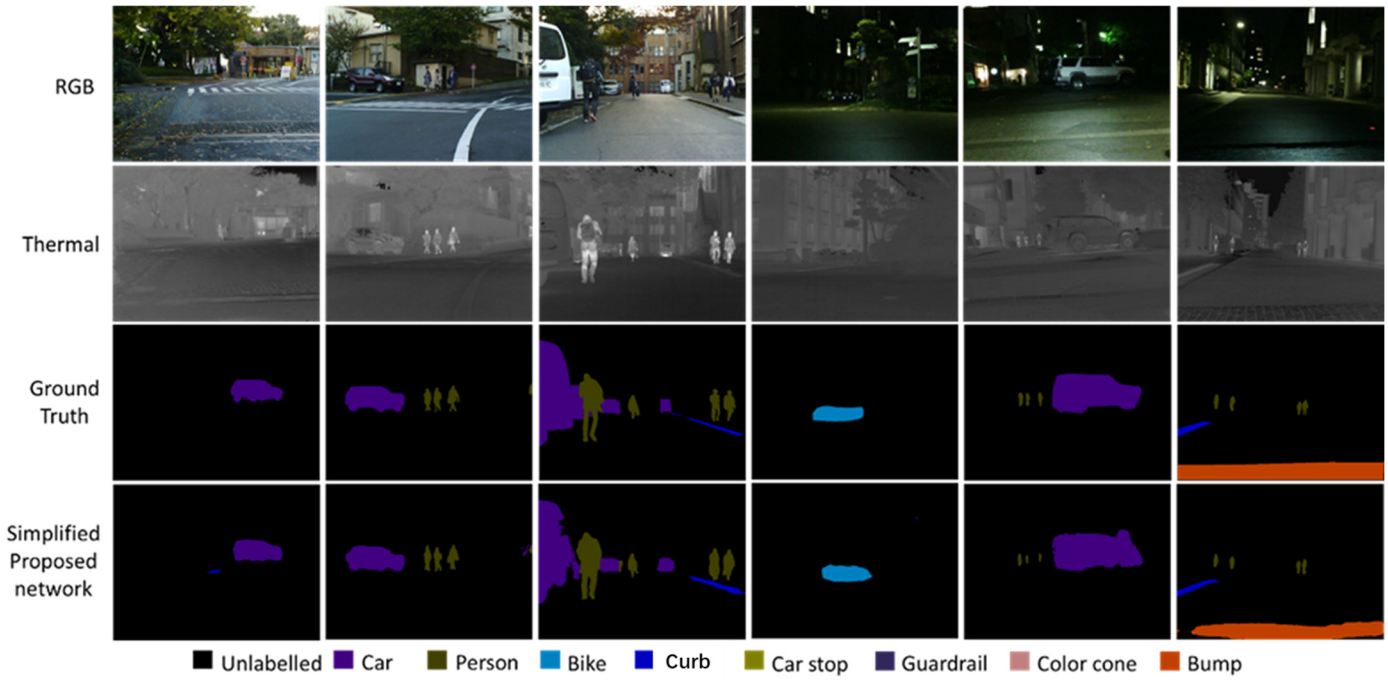


Figure 12. Segmentation results of Simplified Proposed Network after applying 4× speed pruning.

4.5. Experiments for Additional Public Dataset, PST900

This paper also conducts experiments on an additional public dataset, PST900. The PST900 dataset [47] is also a popular benchmark for RGB-T semantic segmentation. It consists of five classes of artifacts found in challenging underground environments, where there is no guarantee of environmental illumination or visibility: fire extinguisher, backpack, hand drill, survivor, and one unlabeled background. The dataset comprises a total of 894 RGB-thermal image pairs with a resolution of 720 × 1280 pixels. Of these, 597 pairs are used for training, and the remaining 297 pairs are used for testing. The hyperparameter settings for training are identical to those used in the experiments on the MFNet dataset.

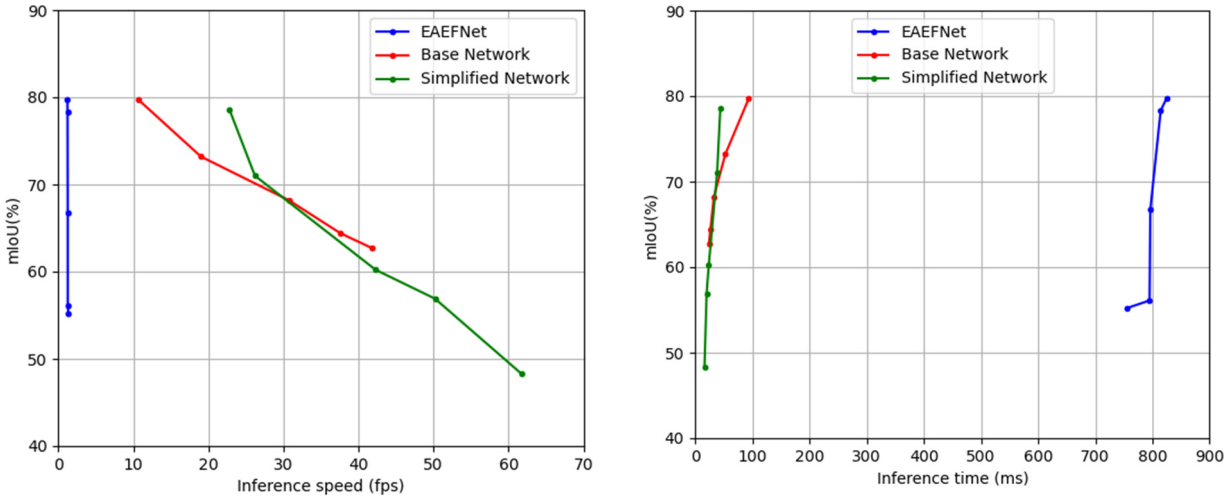
Table 11 presents the segmentation accuracy, model size, and inference speed of EAEFNet, the Base Network, the Proposed Network, and the Simplified Proposed Network on the PST900 dataset. When comparing the performance measured on a desktop, similar to the results on the MFNet dataset, EAEFNet achieves the highest mIoU of 82.9%, followed by the Proposed Network with an mIoU of 81.7%. However, when comparing the performance measured on the DSP, the Proposed Network achieves the highest mIoU of 81.2%. The Simplified Proposed Network has the smallest model size and, consequently, the fastest processing speed. In addition, the inference speeds of the other three networks are significantly faster than that of EAEFNet.

Table 11. Performance comparisons among EAEFNet, Base Network, Proposed Network, and Simplified Proposed Network on PST900 dataset. The best result is shown in bold font.

Network	mIoU (%) on Desktop	mIoU (%) on DSP	Model Size (MB)	FPS on DSP
EAEFNet	**82.9 (+0.0)**	79.8 (+0.0)	113.9	1.2 (+0.0)
Base Network	80.2 (−2.7)	79.7 (−0.1)	72.3	10.7 (+9.5)
Proposed Network	81.7 (−1.2)	**81.2 (+1.4)**	55.8	13.1 (+11.9)
Simplified Proposed Network	79.2 (−3.7)	78.6 (−1.2)	**28.1**	**22.8 (+21.6)**

Figure 13 presents the graphs obtained from the PST900 dataset using the same method as in Figure 11. After applying channel pruning to each network, graphs of

accuracy performance (mIoU) versus inference speed and inference time were plotted. As shown in Figure 13a, EAEFNet experiences a drop in performance due to channel pruning, but its inference speed and inference time remain almost unchanged. In contrast, while the performance (mIoU) of the Base Network and the Simplified Proposed Network decreases due to channel pruning, their inference speed and inference time improve significantly. If an accuracy of around 70% mIoU is acceptable, the Simplified Proposed Network is the most effective choice for deployment on edge devices. The results obtained on the PST900 dataset show the same trend as those from the MFNet dataset, demonstrating that the Proposed Networks are trained robustly and yield stable results across different datasets.



(**a**) (**b**)

Figure 13. Performance comparison graphs for EAEFNet, Base Network, and Simplified Proposed Network on PST900 Dataset: (**a**) semantic segmentation performance (mIoU) versus inference speed (fps); (**b**) semantic segmentation performance (mIoU) versus inference time (ms).

5. Conclusions

This paper proposes an RGB-T semantic segmentation network that can operate in real time on edge devices. Previous studies on RGB-T semantic segmentation have commonly employed separate encoders for each modality and adopted an intermediate fusion strategy to combine the two sources of information at a mid-level stage. However, this approach makes the network structure more complicated and includes layers that are not suitable for edge devices, which causes a very slow inference speed on edge devices. To address this issue, this paper adopts an early fusion input method and designs the network by combining components that are optimized for edge devices and have already been proven effective. Experimental results show that the Proposed Network achieves a performance comparable to state-of-the-art networks in this field, while operating at high inference speed on edge devices.

Future work includes building a dedicated dataset using a self-developed RGB-T camera module and analyzing the robustness of the proposed method under various lighting and weather conditions (e.g., early morning, evening, snow, rain), followed by investigations into potential improvements. Additional analyses will assess the method's robustness against different types of noise and vibration. RGB-T cameras are planned for use in vehicle and factory automation, with evaluations on how load conditions, such as image resolution, affect segmentation accuracy and real-time performance. The self-developed RGB-T camera module will also be applied to vehicle and factory surveillance applications. Once application scenarios and requirements are defined, acceptable mIoU

drops for each situation will be determined. Furthermore, a comparative analysis of various channel pruning methods will be conducted to identify the most suitable approach for the Proposed Network. Research on feature-level fusion and early fusion is also planned for future work.

Author Contributions: Conceptualization, J.Y.H., Y.J.L., H.G.J. and J.K.S.; methodology, J.Y.H., Y.J.L., H.G.J. and J.K.S.; software, J.Y.H. and Y.J.L.; validation, J.Y.H., H.G.J. and J.K.S.; formal analysis, J.Y.H., H.G.J. and J.K.S.; investigation, J.Y.H.; resources, J.Y.H. and Y.J.L.; data curation, J.Y.H.; writing—original draft preparation, J.Y.H.; writing—review and editing, Y.J.L., H.G.J. and J.K.S.; visualization, J.Y.H.; supervision, Y.J.L., H.G.J. and J.K.S.; project administration, J.K.S.; funding acquisition, J.K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2020R1A6A1A03038540), and in part by the Institute of Civil Military Technology Cooperation funded by the Defense Acquisition Program Administration and Ministry of Trade, Industry and Energy of Korean government under grant No. 23-SF-EL-07.

Data Availability Statement: The RGB-Thermal datasets used in this study, MFNet and PST900, are publicly available at the official project website of the University of Tokyo: https://www.mi.t.u-tokyo.ac.jp/static/projects/mil_multispectral/ (accessed on 9 June 2025) and at the website: https://github.com/ShreyasSkandanS/pst900_thermal_rgb (accessed on 11 August 2025), respectively.

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

References

1. Muhammad, K.; Hussain, T.; Ullah, H.; Del Ser, J.; Rezaei, M.; Kumar, N.; Hijji, M.; Bellavista, P.; de Albuquerque, V.H.C. Vision-Based Semantic Segmentation in Scene Understanding for Autonomous Driving: Recent Achievements, Challenges, and Outlooks. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 22694–22715. [CrossRef]
2. Ha, Q.; Watanabe, K.; Karasawa, T.; Ushiku, Y.; Harada, T. MFNet: Towards Real-Time Semantic Segmentation for Autonomous Vehicles with Multi-Spectral Scenes. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), British Columbia, Canada, 24–28 October 2017; pp. 5108–5115. [CrossRef]
3. Nie, J.; Yan, J.; Yin, H.; Ren, L.; Meng, Q. A Multimodality Fusion Deep Neural Network and Safety Test Strategy for Intelligent Vehicles. *IEEE Trans. Intell. Veh.* **2020**, *6*, 310–322. [CrossRef]
4. Muresan, M.P.; Giosan, I.; Nedevschi, S. Stabilization and Validation of 3D Object Position Using Multimodal Sensor Fusion and Semantic Segmentation. *Sensors* **2020**, *20*, 1110. [CrossRef] [PubMed]
5. Zhou, W.; Zhu, Y.; Lei, J.; Wan, J.; Yu, L. CCAFNet: Crossflow and Cross-Scale Adaptive Fusion Network for Detecting Salient Objects in RGB-D Images. *IEEE Trans. Multimed.* **2021**, *24*, 2192–2204. [CrossRef]
6. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, 5–9 October 2015, Proceedings, Part III 18*; Springer: Cham, Switzerland, 2015; pp. 234–241. [CrossRef]
7. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef]
8. Zhou, H.; Tian, C.; Zhang, Z.; Huo, Q.; Xie, Y.; Li, Z. Multispectral Fusion Transformer Network for RGB-Thermal Urban Scene Semantic Segmentation. *IEEE Geosci. Remote Sens. Lett.* **2022**, *19*, 1–5. [CrossRef]
9. Zhang, J.; Liu, H.; Yang, K.; Hu, X.; Liu, R.; Stiefelhagen, R. CMX: Cross-Modal Fusion for RGB-X Semantic Segmentation with Transformers. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 14679–14694. [CrossRef]
10. Liang, M.; Hu, J.; Bao, C.; Feng, H.; Deng, F.; Lam, T.L. Explicit Attention-Enhanced Fusion for RGB-Thermal Perception Tasks. *IEEE Robot. Autom. Lett.* **2023**, *8*, 4060–4067. [CrossRef]
11. Fan, D.P.; Zhai, Y.; Borji, A.; Yang, J.; Shao, L. BBS-Net: RGB-D Salient Object Detection with a Bifurcated Backbone Strategy Network. In Proceedings of the 16th European Conference on Computer Vision (ECCV 2020), Glasgow, UK, 23–28 August 2020; Springer: Cham, Switzerland, 2020; pp. 275–292. [CrossRef]
12. Fang, G.; Ma, X.; Song, M.; Mi, M.B.; Wang, X. Depgraph: Towards Any Structural Pruning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2023), Vancouver, BC, Canada, 19–24 June 2023; pp. 16091–16101. [CrossRef]

13. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
14. Shi, W.; Caballero, J.; Huszár, F.; Totz, J.; Aitken, A.P.; Bishop, R.; Rueckert, D.; Wang, Z. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 1874–1883. [CrossRef]
15. Strudel, R.; Garcia, R.; Laptev, I.; Schmid, C. Segmenter: Transformer for semantic segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 7242–7252.
16. Xie, E.; Wang, W.; Yu, Z.; Anandkumar, A.; Alvarez, J.M.; Luo, P. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12077–12090.
17. Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; Shao, L. Pvt v2: Improved baselines with pyramid vision transformer. *Comput. Vis. Media* **2022**, *8*, 415–424. [CrossRef]
18. Yuan, Y.; Fu, R.; Huang, L.; Lin, W.; Zhang, C.; Chen, X.; Wang, J. Hrformer: High-resolution vision transformer for dense predict. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 7281–7293.
19. Cheng, B.; Misra, I.; Schwing, A.G.; Kirillov, A.; Girdhar, R. Masked-attention mask transformer for universal image segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 1280–1289.
20. Ri, C.-Y.; Yao, M. Semantic Image Segmentation Based on Spatial Context Relations. In Proceedings of the 2012 Fourth International Symposium on Information Science and Engineering (ISISE 2012), Shanghai, China, 23–25 November 2012; IEEE: New York, NY, USA, 2012; pp. 104–108. [CrossRef]
21. Shelhamer, E.; Long, J.; Darrell, T. Fully Convolutional Networks for Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015), Boston, MA, USA, 7–12 June 2015; pp. 3431–3440. [CrossRef]
22. Romera, E.; Alvarez, J.M.; Bergasa, L.M.; Arroyo, R. ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation. *IEEE Trans. Intell. Transp. Syst.* **2017**, *19*, 263–272. [CrossRef]
23. Yu, F.; Koltun, V. Multi-Scale Context Aggregation by Dilated Convolutions. *arXiv* **2015**, arXiv:1511.07122.
24. Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 834–848. [CrossRef] [PubMed]
25. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid Scene Parsing Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), Honolulu, HI, USA, 21–26 July 2017; pp. 2881–2890. [CrossRef]
26. Liu, J.; He, J.; Zhang, J.; Ren, J.S.; Li, H. EfficientFCN: Holistically-Guided Decoding for Semantic Segmentation. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020, Proceedings, Part XXVI*; Springer: Cham, Switzerland, 2020; pp. 1–17. [CrossRef]
27. 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.
28. Sun, Y.; Zuo, W.; Liu, M. RTFNet: RGB-Thermal Fusion Network for Semantic Segmentation of Urban Scenes. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2576–2583. [CrossRef]
29. Zhang, Q.; Zhao, S.; Luo, Y.; Zhang, D.; Huang, N.; Han, J. ABMDRNet: Adaptive-weighted Bi-directional Modality Difference Reduction Network for RGB-T Semantic Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 2633–2642.
30. Zhou, W.; Liu, J.; Lei, J.; Yu, L.; Hwang, J.-N. GMNet: Graded-feature multilabel-learning network for RGB-thermal urban scene semantic segmentation. *IEEE Trans. Image Process.* **2021**, *30*, 7790–7802. [CrossRef]
31. Surianarayanan, C.; Lawrence, J.J.; Chelliah, P.R.; Prakash, E.; Hewage, C. A survey on optimization techniques for edge artificial intelligence (AI). *Sensors* **2023**, *23*, 1279. [CrossRef]
32. Lu, Y.; Ma, H.; Smart, E.; Yu, H. Real-time performance-focused localization techniques for autonomous vehicle: A review. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 6082–6100. [CrossRef]
33. Singh, V.; Singh, S.; Gupta, P. Real-time anomaly recognition through CCTV using neural networks. *Procedia Comput. Sci.* **2020**, *173*, 254–263. [CrossRef]
34. TensorFlow Lite Documentation. Available online: https://www.tensorflow.org/ (accessed on 10 June 2025).
35. ONNX Runtime Documentation. Available online: https://onnxruntime.ai/ (accessed on 10 June 2025).
36. Qualcomm Snapdragon. Available online: https://www.qualcomm.com/snapdragon/overview/ (accessed on 10 June 2025).
37. NVIDIA Jetson Nano. Available online: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (accessed on 10 June 2025).
38. Intel Movidius. Available online: https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu.html (accessed on 10 June 2025).

39. Lee, Y.J.; Hwang, J.Y.; Park, J.; Jung, H.G.; Suhr, J.K. Deep Neural Network-Based Flood Monitoring System Fusing RGB and LWIR Cameras for Embedded IoT Edge Devices. *Remote Sens.* **2024**, *16*, 2358. [CrossRef]
40. Choi, K.; Moon, J.; Jung, H.G.; Suhr, J.K. Real-Time Object Detection and Tracking Based on Embedded Edge Devices for Local Dynamic Map Generation. *Electronics* **2024**, *13*, 811. [CrossRef]
41. Lee, Y.; Jung, H.G.; Suhr, J.K. Semantic Segmentation Network Slimming and Edge Deployment for Real-Time Forest Fire or Flood Monitoring Systems Using Unmanned Aerial Vehicles. *Electronics* **2023**, *12*, 4795. [CrossRef]
42. Choi, K.; Wi, S.M.; Jung, H.G.; Suhr, J.K. Simplification of Deep Neural Network-Based Object Detector for Real-Time Edge Computing. *Sensors* **2023**, *23*, 3777. [CrossRef]
43. Kim, G.; Yoo, J.H.; Jung, H.G.; Suhr, J.K. Precise Position Estimation of Road Users by Extracting Object-Specific Key Points for Embedded Edge Cameras. *Electronics* **2025**, *14*, 1291. [CrossRef]
44. Qualcomm Neural Processing SDK. Available online: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk (accessed on 8 April 2024).
45. Bengio, Y. Neural networks: Tricks of the trade. In *Lecture Notes in Computer Science Book*, 2nd ed.; Montavon, G., Orr, G.B., Müller, K.-R., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7700.
46. WITHROBOT. Available online: http://withrobot.com/ (accessed on 10 June 2025).
47. Shivakumar, S.S.; Rodrigues, N.; Zhou, A.; Miller, I.D.; Kumar, V.; Taylor, C.J. PST900: RGB-Thermal Calibration, Dataset and Segmentation Network. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 9441–9447.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.