CRFNet: Context ReFine ment Network used for semantic segmentation

Taeghyun An | Jungyu Kang | Dooseop Choi | Kyoung-Wook Min

Abstract
Recent semantic segmentation frameworks usually combine low-level and high-level context information to achieve improved performance. In addition, postlevel context information is also considered. In this study, we present a Context ReFine ment Network (CRFNet) and its training method to improve the semantic predictions of segmentation models of the encoder–decoder structure. Our study is based on postprocessing, which directly considers the relationship between spatially neighboring pixels of a label map, such as Markov and conditional random fields. CRFNet comprises two modules: a refiner and a combiner that, respectively, refine the context information from the output features of the conventional semantic segmentation network model and combine the refined features with the intermediate features from the decoding process of the segmentation model to produce the final output. To train CRFNet to refine the semantic predictions more accurately, we proposed a sequential training scheme. Using various backbone networks (ENet, ERFNet, and HyperSeg), we extensively evaluated our model on three large-scale, real-world datasets to demonstrate the effectiveness of our approach.

KEYWORDS
autonomous driving, context refinement, parameter freezing, semantic segmentation

1 INTRODUCTION

Semantic segmentation is the task of classifying every pixel in an input image into a predefined class. It can provide comprehensive information and can be used in various fields, such as autonomous driving (e.g., moving-object detection and drivable space detection), medical imaging (e.g., white blood cell segmentation, abnormal region extraction), satellite imaging, and image editing (e.g., boundary extraction and attention area extraction).
(encoder–decoder structure). In addition to the extension of the depth and width of the network architecture, various concepts, such as multiresolution [3–5], multiloss [6,7], and multitasking [8,9], have been applied.

In the dictionary sense, context is the situation in which something happens, or the group of conditions that exist at the site and time something happens. In semantic segmentation, the context implies that the information (feature, label) of a given pixel is affected by the surrounding information, and vice versa. For example, there are positional characteristics, such as “people or vehicles are on the ground,” or shape characteristics, such as “the approximate shape of the vehicle is a polygon with a bumpy bottom part.”

In semantic segmentation, the deep-learning network typically consists of an encoder–decoder structure. Recent deep-learning-based semantic segmentation uses dilated convolution [10,11] (also known as atrous convolution) or pyramidal structures [4,12,13] to present context information more effectively. These approaches represent the context of network encoder features. Although most semantic information can be contained in the encoding stage, it does not contain fine-level label information because it is processed before it passes through the decoder. Therefore, if the context of the label information is directly represented, improved results can be expected.

Random-field-based techniques are used to represent the context of the label itself [10,14,15]. They are extensively used as a postprocessing step in the classical, handcrafted, feature-based segmentation framework, and produce results that satisfy both the unary term related to the per-pixel label probability and the pairwise term considering the context between neighboring pixels. In the case of a Markov random field, it smoothens the label between neighboring pixels, and in the case of a conditional random field, there may be an improved effect associated with the use of additional information to make the boundary fit better. In some cases, CNN-based semantic segmentation and random-field techniques have been used in conjunction, or implemented as additional layers.

In this study, we propose a network refinement structure that extends existing semantic segmentation networks. As shown in Figure 1, the core component of CRFNet is a simple refiner–combiner structure. The

**FIGURE 1** Proposed Context ReFinement Network (CRFNet) framework overview. (Top) A typical fully convolutional neural network for semantic segmentation with an encoder–decoder structure. (Bottom) The CRFNet framework makes it possible to consider the context of the label itself by using the resulting label of the previous network as a feature. It considers the context of the label as it passes through the refiner module, and combines features to improve the results of the existing network in the combiner module. Additionally, parameter freezing can be used appropriately in the training process to achieve improved results.
refiner module extracts context information from the output features of the conventional semantic segmentation network. Unlike the encoder features, which implicitly contain label information, the output feature of the decoder is closer to the label. Therefore, it is possible to add context information that differs from the context handled at the encoder level. An improved result can then be produced using a combiner that combines the features from the previously refined result with those from the backbone network. In addition, we propose a sequential training scheme suitable for the proposed refinement network. Using this sequential training scheme, the proposed network can play the role of a refinement network that can address the shortcomings of the existing backbone network and not just a multistage network. We demonstrated the effectiveness of the proposed method based on extensive experiments.

A particular advantage of the proposed method is that the refinement module can be easily applied to any semantic segmentation network with structures that can extract the decoder features of the output label that immediately precedes the spatial level. Using the encoder–decoder-based conventional semantic segmentation network as the backbone network, improved results can be obtained with the proposed refinement scheme.

The proposed method was developed to improve semantic segmentation models for autonomous driving, and significant performance improvements can be observed in lightweight and fast models corresponding to semantic segmentation methods for autonomous driving.

In summary, the proposed method makes five major contributions to the literature.

- We propose a novel network extension method that considers context information from label information directly using a small number of additional parameters proportional to the square of the number of semantic classes.
- The proposed method can easily extend existing semantic segmentation networks if the decoder features of the output label and the immediately preceding spatial level can be extracted.
- We propose a sequential training method suitable for the proposed refinement network.
- Experimental results show that the proposed method stably improves the performance of existing semantic segmentation models.
- In addition to the extensively used Cityscapes [16] and CamVid datasets [17], experiments were conducted using the recently distributed KITTI-360 dataset [18] and produced meaningful results.

## 2 RELATED WORKS

### 2.1 Semantic segmentation

The development of deep learning has become a stepping-stone in the remarkable improvement of semantic segmentation. Since the completion of the pioneering works on fully convolutional networks [1] and DeConvNet [2], various research directions have been studied in accordance with the characteristics required by semantic segmentation.

Semantic segmentation is a task that requires dense prediction, and there have been methods that preserve high-resolution information using skip connections and encoder–decoder structures [19,20]. In addition, attempts have been expended to widen the receptive field using dilated convolution (atrous convolution) [11,21,22] or spatial pyramidal pooling [13]. Semantic segmentation can be useful with high accuracy; therefore, attempts have been expended to boost performance through a network of deep and complex structures [23,24] and self-attention modules that reweight feature channels [25–27]. Lightweight models have been created for real-time application purposes, such as autonomous driving [28–30].

### 2.2 Context information

Before the prevalence of deep learning, semantic segmentation algorithms used context information from graph structures, such as Markov and conditional random fields, which helped improve label prediction or identify boundaries more accurately [31,32]. They use pixel or superpixel features and can generate improved segmentation results by constructing pairwise potentials in addition to the unary potential, which is unrefined label information. Even after deep learning became popular, attempts had been expended to improve the segmentation results by combining deep networks and random fields [10,33,34].

Convolution reflects the context between adjacent pixels. CNNs, which run through multiple convolution and downsampling layers, naturally contain context information between distant pixels and are often used to solve vision problems. Additional concepts can be used to include additional contextual information. For example, some aimed to enlarge the receptive field using dilated convolution (atrous convolution) [11,20], some aimed to fuse context at different feature levels by using a multiresolution solution [13,35,36], and others suggested networks of recurrent neural network (RNN) structures to represent the long-range context [37,38].

Recently, multistage networks have been used for computer vision tasks. This concept was first applied to human pose estimation [39,40]. Since then, this concept...
has been applied to the field of semantic segmentation [27,41]. A stacked deconvolutional network (SDN) [41] stacks multiple shallow deconvolutional units with dense connections to capture more contextual information and make optimization easier. Cheng and others [27] proposed a carefully designed encoder–decoder stacking architecture with a semantic prediction guidance module.

The proposed refinement network is similar to the aforementioned multistage networks for semantic segmentation. Unlike multistage networks that require a sequential supervision scheme, unlike the simultaneous supervision scheme of existing methods, it is possible to develop a real refinement network that supplements the deficiencies in the backbone network.

3 MATERIALS AND METHODS

This study aims to improve semantic segmentation models to handle the context of the label itself. We developed a context-refinement framework for semantic segmentation that extends existing semantic segmentation models, and a pretrained model can be used as a backbone network. Various CNN-based semantic segmentation architectures can be used as backbone networks; ERFNet [29], ENet [42], HyperSeg [28], and DDRNet [43] were used in this study. The proposed refinement structure begins by feeding the output label map from the backbone network (Figure 1). The “refiner” extracts the context information of the label map entered, and the “combiner” adds the resulting context information to the backbone network. The proposed method uses a sequential supervision scheme to supplement deficiencies in the backbone network. A great advantage of the proposed method is that performance can be improved consistently for any network by using an existing pretrained network and adding the same “refiner” and “combiner” structures.

In this section, we discuss the “refiner” and “combiner” structures first. Subsequently, we describe the details of the refinement process for specific networks.

3.1 Proposed context refinement architecture

The proposed model is defined as follows.

\[ O = c(r(F_s|\theta^r),F_{s-1}|\theta^r), \] (1)

where \( F \) and \( s \), respectively, denote the feature map and spatial resolution of the feature map corresponding to the output label map. \( c(\cdot|\theta^r) \) and \( r(\cdot|\theta^r) \) are combiners with parameter \( \theta^r \) and refiners with parameter \( \theta^r \) to compute the final output \( O \).

From the input image \( I \in \mathbb{R}^{3 \times H \times W} \) (where \( H \) and \( W \) denote the height and width of the input image, respectively), the backbone network \( b \) generates feature maps \( F_i,i \in [0,1,\ldots,s] \) of different spatial resolutions \( i \) with parameters \( \theta^b \):

\[ F_0,F_1,\ldots,F_s = b(I|\theta^b). \] (2)

The feature map \( F_s \) corresponding to the label map was used as the input to the refiner. As the input of the refiner is the label itself, the function of the refiner forces the networks to generate label-contextual features for semantic segmentation. The combiner is then passed through the feature map \( F_{s-1} \) of the subsequent \( s-1 \) levels, and the final result is the output (Figure 2).

3.1.1 Refiner

The dotted red boxes in Figures 1 and 2A represent the refiner. Refiners obtain richer contextual information by combining the two types of contextual components.

In the left part of the refiner (Figure 2A), the context is acquired through the so-called encoding–decoding of label information. First, we used two downsampling modules (Figure 2B) to enlarge the receptive field. The downsampling modules consisted of \( 3 \times 3 \) convolution operations with stride two, batch normalization, and ReLU operators. In addition, the dilated convolution modules (Figure 2C) are followed by various dilation rates of the convolution kernels. In our method, we used \( 3 \times 3 \)-dilated convolution with the dilation rates of 2, 4, 8, and 16. Subsequently, our refiner produced a feature map that matched the resolution with that of the feature map (that will be combined later) using an upsampling module (Figure 2D) containing a transposed convolution and a convolution module. The convolution module is the same as the dilated convolution module with a dilation rate of 1.

Because we only want to strengthen the context information of the label through the refiner, the number of channels in the feature map is the same as the number of segment classes and does not change throughout the entire duration of the left part. Therefore, the number of parameters does not increase significantly compared with the baseline network, which will be discussed later.
On the right side of the refiner, the context of the label information is obtained through a context-pooling module (Figure 2E). To obtain the proper context, we applied two-dimensional (2D) average-pooling operations in \( H/8 \times W/8 \) regions with \( H/16 \times W/16 \) steps. Finally, we combined the left and right parts of the refiner. The atrous convolutional-based context information of the left part was refined using the polling-based context information of the right part.

As a major difference from multistage networks, the proposed refinement module adds only the context of the backbone network output feature. Therefore, the number of channels for each layer is equal to the number of semantic classes, which is the number of channels of the backbone-network output feature, without increasing or decreasing the number of channels. Existing multistage networks are stacked based on the concept of encoder–decode–repetition; therefore, the number of channels during encoding increases and decreases again during decoding. As a result, the proposed refinement network is more efficient in terms of the number of parameters and graphical processing unit (GPU) memory usage than existing multistage networks.

### 3.1.2 Combiner

The dotted green boxes in Figures 1 and 2A represent the combiner. The combiner begins with a simple concatenation of \( r(F_s|\theta') \), a feature generated in the previous refiner section, and \( F_{s-1} \), a feature from the backbone network at the \( (s-1) \) resolution level. The final resulting label map \( O \) was generated using a transposed convolution.

### 3.2 Network training with sequential supervision

To refine the backbone network, we propose herein a sequential supervision method.
The simultaneous supervision scheme (Figure 3A) used in the existing multistage methods [27,41] can be expressed by the following loss function,

$$\arg\min_{\theta^b, \theta^r, \theta^c} l(O, G) + \alpha l(F_s, G)$$  \hspace{1cm} (3)$$

where $\theta^b$, $\theta^r$, $\theta^c$ denote the parameters of the backbone network, refiner, and combiner, respectively. $G$ denotes the ground-truth label map, and $O$ and $F_s$ are the final output feature map and the output feature map from the backbone network, respectively. $l$ denotes a loss function, and $\alpha$ is a balancing coefficient between the two loss functions related to $O$ and $F_s$. Although the network form is refined, this scheme does not guarantee that the actual execution will be refined. $\theta^r$ and $\theta^c$ are trained such that the final output $O$ is equal to $G$; they are also trained under the influence of $\theta^b$, such that the intermediate result $F_s$ is also equal to $G$. Even the hyperparameter $\alpha$ requires an additional (laborious) workload to be determined experimentally.

The proposed sequential supervision begins training using the given backbone-network parameters $\theta^b$ based on the following equation.

$$\arg\min_{\theta^b} l(F_s, G).$$  \hspace{1cm} (4)$$

Existing trained networks can be imported and used, or training can be performed afresh. We then trained the parameters of the combiner and refiner, $\theta^r$ and $\theta^c$ by freezing parameter $\theta^b$.

$$\arg\min_{\theta^r, \theta^c} l(O, G; \theta^b).$$  \hspace{1cm} (5)$$

Training based in this manner makes it possible to learn $\theta^r$ and $\theta^c$ which can obtain the final results by refining the resulting features from the backbone network.

3.2.1 | Two-phase training

To optimize (5), we actually use two training phases.

In the first phase, all the parameters $\theta^b$, $\theta^r$, and $\theta^c$ are updated with the initial parameters $\theta^b$. Although well-trained backbone-network parameters $\theta^b$ are imported, they are not optimized for the entire network; therefore, they must be coordinated.

In the second phase, we freeze parameter $\theta^b$ and update the remaining parameters $\theta^r$ and $\theta^c$. In this case, the initial learning rate was reduced by half compared with the previous phase.
3.3 Network refinement with different backbones

CRFNet has a practical benefit in that it can be applied to any backbone network if the final label feature \(F_s\) and the feature at the spatial level below that level \(F_{s-1}\) can be obtained. In general, the number of feature channels is the same as the number of segmentation classes at spatial level \(s\), and at the \((s-1)\) level, the number varies slightly from one network architecture to another. Therefore, when applying the CRFNet architecture to the other models, there was no significant change, except for the feature dimension at the \((s-1)\) level.

In the experimental section, we applied CRFNet to four efficient networks: ERFNet [29], ENet [44], HyperSeg [28], and DDRNet [43]. The ERFNet is an encoder–decoder structure network that uses factorized one-dimensional convolutions \((3 \times 1\) and \(1 \times 3\)) to improve computational efficiency with comparable accuracy; ENet is an efficient network for real-time semantic segmentation; HyperSeg is a relatively recent model that uses encoder features to generate decoder parameters in encoder–decoder structures; DDRNet consists of deep dual-resolution backbones and enhanced low-resolution contextual information extractors with bilateral fusions to generate high-quality details.

In the cases of ERFNet and ENet, the proposed method (Figure 2) can be applied as is because they have an encoder–decoder structure. In the case of HyperSeg, although the decoder is slightly more complex in the case of HyperSeg, it can still extract features from the last layer and half-sized features from the preceding layer. Therefore, the proposed method is applied. In the case of DDRNet, the encoder–decoder structure may not be evident. However, it passes through a module called a segmentation head, which consists of a \(3 \times 3\) convolutional layer, followed by a \(1 \times 1\) convolutional layer, and produces the final segmentation result. This segmentation head can be considered a type of decoder, and features can be extracted both before and after the segmentation head to apply the proposed method. Because the spatial resolutions of the features before and after the segmentation head are the same, we replace the convolution module in Figure 2 with an upsampling module and the transposed convolution module with an additional segmentation head module.

3.4 Number of parameters

In the proposed CRFNet, only simple layers were added; these consisted of the same number of convolution kernels as the number of semantic classes. The number of channels of deep layers in a baseline network was a number of the order of hundreds (e.g., 128 or 256), and the number of classes was a number of the order of 10 units (19 classes for Cityscapes and 11 classes for CamVid).

In a simple convolution layer, if the number of input channels is \(C_i\), the number of output channels is \(C_o\), and the filter size is \(K\); the number of parameters \(N_p\) in that layer is as follows,

\[
N_p = N_w + N_b, \\
N_w = K^2 C_i C_o, \\
N_b = C_o,
\]

where \(N_w\) and \(N_b\) are the numbers of weights and biases, respectively. \(C_x \in \{i, o\}\) is a number that is of the order of hundred and 10 units in a baseline network and the proposed “refiner-combiner” module, respectively. Therefore, with the proposed CRFNet, the total number of parameters compared with the baseline network increased by only a few percentage units.

For example, in the case of ERFNet [29] with \(1024 \times 512\) Cityscapes dataset, the total number of parameters of the model was approximately 2.06 M. For the proposed CRF-ERFNet, the number of parameters was 2.14 M, which was only 0.08 M more than the ERFNet. Similarly, the computational complexity increased slightly from a level similar to that of the baseline network (e.g., ERFNet had 30.1 GMACs, and CRF-ERFNet had 35.1 GMACs). The number of parameters and computational complexity were measured using the THOP library, which is a tool for counting the number of parameters and computational complexity.

4 EXPERIMENTS

As mentioned previously, the proposed method was developed to improve the performance of semantic segmentation models for autonomous driving. Semantic segmentation uses suitable benchmark datasets consisting of data collected from vehicles [16-18,45,46].

To evaluate the effectiveness and robustness of the proposed method, we used three publicly available datasets for camera-based semantic segmentation: Cityscapes [16], Camvid [17] and KITTI-360 [18].

Mean intersection-over-Union (mIoU) is an extensively used evaluation metric in semantic segmentation. The definition of IoU for a particular class is,

\[
\text{IoU} = \frac{TP}{TP + FP + FN},
\]
where TP, FP, and FN are the number of true positives, false positives, and false negatives at the pixel level, respectively. The mIoU was obtained by averaging the IoU for all existing classes (19 classes in Cityscapes, 11 classes in Camvid, and 17 classes in the KITTI-360 datasets). We also report on the class IoU to observe the effects of refineNet on different classes. In this study, the corresponding metric was expressed as a percentage.

### Implementation details:

Existing semantic segmentation methods (e.g., ERFNet, ENet, and HyperSeg) and the proposed CRFNet models were all implemented using Pytorch (with CUDA 11.4) and cuDNN back-ends; experiments were performed using 4 × RTX A-6000 GPUs. In the cases of the Cityscapes and CamVid benchmarks, we applied the following image augmentation techniques: horizontal flipping with a probability of 0.5, random hue and saturation jitter, random resize with a scale range $[0.8, 1.2]$, random rotation with an angle range $[-10^\circ, 10^\circ]$ and image cropping. For KITTI-360, we used only horizontal flipping and color jitter augmentation.

In the training step, the cross-entropy loss was used to learn the networks for Cityscapes and KITTI-360, and the Lovasz loss [42] was used to learn the networks for CamVid. The Adam optimizer [47] was used, and the initial learning rate was 0.001 or 0.0005 in the first training phase, and half in the second training phase.

In the cases of ERFNet and ENet, the backbone network was trained afresh, and in the case of Hyperseg, the Hyperseg-M model was used (among the models provided by the author’s GitHub). In the DDRNet case, the DDRNet-23-slim model was used and trained from scratch.

### 4.1 Cityscapes

Cityscapes [16] is a collection of images and ground-truth labels from the driver’s perspective. Cityscapes has 5000 finely annotated image pairs and 20 000 coarsely annotated image pairs collected; however, the coarsely annotated images were not used to train the model. The finely annotated images comprised 2975 training images, 500 validation images, and 1525 test images. It contained 19 different classes, and each image had a resolution of 2048 × 1024.

Table 1 presents an evaluation of the segmentation accuracy of our method on the Cityscapes dataset. Experiments were conducted using ERFnet [29], ENet [44], HyperSeg [28], and DDRNet [43]. Overall, a significant accuracy increase was observed. Similar or slightly improved responses were observed in most classes, especially for objects such as riders, buses, and motorcycles. In particular, for ENet, which had a relatively low performance among the experimental models, the performance improved for all classes. The lower the baseline...
performance is, the greater the room for improvement in the proposed refinement scheme will be.

Figure 4 illustrates the effectiveness of the proposed refinement method using ERFNet. The proposed method improves the overall performance and has characteristics similar to those of RFs. Although the results were similar to those of the baseline network, the label information was somewhat smoothed and unified and the object boundaries tended to be represented more clearly.

4.2 CamVid

The CamVid dataset [17] is a road scene dataset comprising 701 densely annotated images. It consisted of 367 training, 101 validation, and 233 test images. It contained 11 semantic classes with a resolution of 960 × 720.

4.3 KITTI-360

KITTI-360 [18] is a recently released, large-scale dataset containing high-quality 2D and three-dimensional (3D) annotations. It consisted of 11 driving sequences, which corresponded to distinct and continuous driving

Table 2 lists the segmentation performances of the proposed method. In this dataset, we experimented with ERFnet and ENet, and in both cases, the overall performance of almost all classes was improved. In particular, objects with low IoU scores, such as signs and bicyclists, improved more than others. In the cases of natural structures or structures with large shapes, performance improvements were minor. In the case of the sky, the IoU decreased, but the change was small.

Table 2 lists the segmentation performances of the proposed method. In this dataset, we experimented with ERFnet and ENet, and in both cases, the overall performance of almost all classes was improved. In particular, objects with low IoU scores, such as signs and bicyclists, improved more than others. In the cases of natural structures or structures with large shapes, performance improvements were minor. In the case of the sky, the IoU decreased, but the change was small.

<table>
<thead>
<tr>
<th>Model</th>
<th>Sky</th>
<th>Building</th>
<th>Pole</th>
<th>Road</th>
<th>Sidewalk</th>
<th>Tree</th>
<th>Sign</th>
<th>Fence</th>
<th>Car</th>
<th>Pedestrian</th>
<th>Bicyclist</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERFNet [29]</td>
<td>93.9</td>
<td>90.3</td>
<td>48.3</td>
<td>95.9</td>
<td>83.9</td>
<td>82.1</td>
<td>53.1</td>
<td>66.2</td>
<td>89.3</td>
<td>59.8</td>
<td>59.2</td>
<td>74.7</td>
</tr>
<tr>
<td>CA-ERFNet</td>
<td>93.9</td>
<td>90.8</td>
<td>50.6</td>
<td>96.5</td>
<td>85.3</td>
<td>82.2</td>
<td>55.6</td>
<td>67.6</td>
<td>91.3</td>
<td>63.5</td>
<td>63.6</td>
<td>76.4</td>
</tr>
<tr>
<td>ENet [44]</td>
<td>93.8</td>
<td>89.1</td>
<td>46.2</td>
<td>96.3</td>
<td>85.0</td>
<td>79.9</td>
<td>48.5</td>
<td>66.1</td>
<td>89.9</td>
<td>57.2</td>
<td>57.0</td>
<td>73.5</td>
</tr>
<tr>
<td>CA-ENet</td>
<td>93.7</td>
<td>90.2</td>
<td>48.9</td>
<td>96.4</td>
<td>85.4</td>
<td>81.7</td>
<td>51.8</td>
<td>67.8</td>
<td>91.5</td>
<td>62.1</td>
<td>64.6</td>
<td>75.8</td>
</tr>
</tbody>
</table>

Note: Experiments were performed with input images with sizes equal to 960 × 720. Among the original network and the CRF-network, the class that showed higher performance was emphasized in bold.
trajectories. Each sequence provided sensor data, including a perspective stereo camera, a pair of fisheye cameras, Velodyne, and SICK laser scanning. It also provided ground-truth data for semantic segmentation, confidence scores, and 3D bounding boxes. Nine sequences with ground-truth semantic segmentation data paired with perspective camera images suitable for semantic segmentation were used as the dataset. There were 45 108 images in the training set; these corresponded to sequence numbers 0, 2, 3, 4, 5, and 6, and the validation set consisted of 16 060 images that corresponded to sequence numbers 7, 9, and 10. The KITTI-360 dataset contained many overlapping and similar data regarding the characteristics of sequence images. Therefore, for efficient training, the sequences were sampled in five frames for training and validation. Following [18], 17 semantic labels were used (the same as those used for Cityscapes, excluding the bus and train classes).

Table 3 lists the segmentation performances of the proposed method. For this dataset, we experimented with ERFnet and ENet. The KITTI-360 dataset consisted of image sequences captured while driving, and there were a few objects other than cars. This also affected the training; therefore, the overall recognition rate of objects other than cars was low. Nevertheless, refining the network using the proposed method works better on the backbone network; in particular, the margin was larger in the object family. Even in the case of bicycles, these were not detected at all with ENet but were detected after refinement. In the case of traffic lights, neither ERFNet nor ENet models can find them, even after refinement; this seems to be due to the small number of samples and small sizes of the objects. The corresponding visual results are shown in Figure 5. In the case of KITTI-360, the ground truth image is superimposed on the original image for readability because the height of an image is low compared with other datasets.

4.4 | Ablation studies

Ablation studies were conducted using the Cityscapes validation dataset.

4.4.1 | Effects of the two-phase training procedure

To evaluate the effect of the two-phase training procedure (described in Section 3.2), we compared first the proposed context-refinement network with the same network trained without a pretrained backbone network $\theta_b$ and parameter freezing. We also demonstrate the performance of the proposed two-phase training procedure by
comparing it with the results of training up to a single phase. In Table 4, CRF-ERFNet and CRF-ENet are compared. It is confirmed that the performances of both networks improve as each phase (using a pretrained backbone network) and parameter freezing progresses. Specifically, the ERFNet [29] scored 70.1 in terms of the mIoU metric. When we trained the CRF-ERFNet (without the pretrained backbone network), the score was 70.8; the improvement is minor compared with the original ERFNet. However, following the proposed two-phase training procedure, the performance improved in a step-by-step manner.

### 4.4.2 | Comparison with simultaneously supervised networks

To obtain supervised networks simultaneously, we performed optimizations (3) with different hyperparameters $\alpha$. In Table 5, CRF-ERFNet and CRF-ENet are compared. The hyperparameter $\alpha$ seems to have attained its optimal value, and even if an appropriate value is set, the proposed sequentially supervised network works better. Specifically, the ERFNet [29] and proposed CRF-ERFNet mIoU scores are 70.1 and 72.6, respectively. When we train CRFNet in a simultaneous supervision manner using (3), a maximum performance of 71.6 can be obtained with $\alpha = 0.5$, and the other values yield lower performance. However, even when the appropriate $\alpha$ value was 0.5, the performance of the simultaneous supervision method was inferior to that of the proposed sequential supervision method.

### 4.4.3 | Experiments without a combiner

The proposed CRFNet consists of a refiner and a combiner. While the context information of the label is
By adding a refiner-combiner after a conventional semantic segmentation network structure, the proposed method can achieve additional performance improvements. We demonstrated the effectiveness of the proposed refinement structure on three benchmark datasets which consisted of images captured from vehicles, and experiments were conducted using real-time semantic segmentation models suitable for autonomous driving. Although sufficient context was extracted for performance improvements, a simple refiner form was used. In the future, we plan to experiment with refiner structures that contain more contextual information.

ACKNOWLEDGMENTS
This research work was partly supported by an Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2020-0-00002, Development of standard SW platform-based autonomous driving technology to solve social problems of mobility and safety for public transport-marginalized communities, contribution rate: 50%) and by an IITP grant funded by the Korean government (MSIT) (No. 2021-0-00891, Development of AI Service Integrated Framework for Autonomous Driving, contribution rate: 50%).

CONFLICT OF INTEREST STATEMENT
The authors declare that there are no conflicts of interest.

ORCID
Taeghyun An https://orcid.org/0000-0002-7573-9059
Jungyu Kang https://orcid.org/0000-0003-3411-1932
Dooseop Choi https://orcid.org/0000-0003-0150-7017

REFERENCES
7. R. Sun, X. Zhu, C. Wu, C. Huang, J. Shi, and L. Ma, *Not all areas are equal: Transfer learning for semantic segmentation via hierarchical region selection*, (IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA), 2019. [https://doi.org/10.1109/CVPR.2019.00271](https://doi.org/10.1109/CVPR.2019.00271)


22. F. Yu, V. Koltun, and T. Funkhouser, *Dilated residual networks*, (Proc. IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA), 2017. [https://doi.org/10.1109/CVPR.2017.75](https://doi.org/10.1109/CVPR.2017.75)


30. S. Gould, R. Fulton, and D. Koller, *Decomposing a scene into geometrically and semantically consistent regions*, (IEEE 12th International Conference on Computer Vision, Kyoto, Japan), 2009. [https://doi.org/10.1109/ICCV.2009.5459211](https://doi.org/10.1109/ICCV.2009.5459211)

31. S. Yu, V. Koltun, and T. Funkhouser, *Dilated residual networks*, (Proc. IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA), 2017. [https://doi.org/10.1109/CVPR.2017.75](https://doi.org/10.1109/CVPR.2017.75)


AUTHOR BIOGRAPHIES

Taeg-Hyun An received his BS and PhD degrees in Electrical Engineering from POSTECH, Pohang, Republic of Korea, in 2007 and 2016, respectively. He is a Senior Researcher at the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea. His research interests include deep learning, autonomous driving, and semantic segmentation.

Jungyu Kang received his BS and MS degrees from the School of Electrical and Electronic Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea, in 2014. He is currently a researcher at the Electronics and Telecommunications Research Institute in Daejeon, Republic of Korea. His research interests include deep learning, autonomous driving, semantic segmentation, and simultaneous localization and mapping.

Dooseop Choi received his BS degree in Electronics Engineering from Korea University, Seoul, Republic of Korea, in 2006, and the MS and PhD degrees in Electrical Engineering and Computer Science from Seoul National University, Seoul, Republic of Korea, in 2008 and 2014, respectively. He was a Senior Engineer with Samsung Electronics, Suwon, Republic of Korea, from 2014 to 2017. Since 2017, he has been with Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea, where he is currently a Senior Researcher. His research interests include trajectory forecasting, motion planning, computer vision, and deep learning.

Kyoung-Wook Min received his BS and MS degrees in Computer Engineering from Pusan National University, Busan, Republic of Korea, in 1996 and 1998, respectively. He received his PhD degree in Computer Engineering from Chungnam National University, Daejeon, Republic of Korea, in 2012. Since 2001, he has been a Principal Researcher at Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea, and is also the director of the autonomous driving intelligence research section. His main research interest area is autonomous driving.

How to cite this article: T. An, J. Kang, D. Choi, and K.-W. Min, CRFNet: Context ReFinement Network used for semantic segmentation, ETRI Journal 45 (2023), 822–835. DOI 10.4218/etrij.2023-0017