remote sensing

MDPI

Article

Reinforcement Learning and Genetic Algorithm-Based Network Module for Camera-LiDAR Detection

Taek-Lim Kim [1] and Tae-Hyoung Park [2,*]

[1] Department of Control and Robot Engineering, Chungbuk National University, Cheongju 28644, Republic of Korea; taeglem@chungbuk.ac.kr
[2] Department of Intelligent Systems & Robotics, Chungbuk National University, Cheongju 28644, Republic of Korea
* Correspondence: taehpark@chungbuk.ac.kr

Abstract: Cameras and LiDAR sensors have been used in sensor fusion for robust object detection in autonomous driving. Object detection networks for autonomous driving are often trained again by adding or changing datasets aimed at robust performance. Repeat training is necessary to develop an efficient network module. Existing efficient network module development changes to hand design and requires much module design experience. For this, a neural architecture search was designed, but it takes much time and requires optimizing the design process. To solve this problem, we propose a two-stage optimization method for the offspring generation process in a neural architecture search based on reinforcement learning. In addition, we propose utilizing two split datasets to solve the fast convergence problem as the objective function of the genetic algorithm: source data (daytime, sunny) and target data (day/night, adversary weather). The proposed method is an efficient module generation method requiring less time than the NSGA-NET. We confirmed the performance improvement and the convergence speed reduction using the Dense dataset. Through experiments, it was proven that the proposed method generated an efficient module.

Keywords: sensor fusion; deep learning; neural architecture search

Citation: Kim, T.-L.; Park, T.-H. Reinforcement Learning and Genetic Algorithm-Based Network Module for Camera-LiDAR Detection. *Remote Sens.* **2024**, *16*, 2287. https://doi.org/10.3390/rs16132287

Academic Editor: Filiberto Chiabrando

Received: 23 April 2024
Revised: 14 June 2024
Accepted: 19 June 2024
Published: 22 June 2024

1. Introduction

Artificial neural networks have made significant advancements in autonomous driving, particularly in classification [1] and object detection [2,3]. In such environments, variations in light and humidity, particularly during adverse weather conditions or at night, can affect LiDAR data [4]. Therefore, designing networks that can adapt to various conditions in autonomous driving is crucial, and sensor fusion has proven to be an effective method for this. Recently, research on camera-LiDAR detection in autonomous driving has explored middle-level fusion techniques [5,6], which involve integrating inputs from different sensor-based networks.

Figure 1a shows the object detection network frameworks. The network takes input from a single sensor-based detection network, which refines the features and performs detection by inputting it to the middle layer or directly to the head. When fusing networks from different sensors, a multi-modal fusion method is preferred. This involves creating a backbone network for each sensor and performing fusion at a middle layer. Figure 1b,c illustrate fusion in the middle layer, and (b) is a simple fusion method carried out by stacking or concatenating features. However, in the case of multi-modal fusion, a recalibration process is required between the features from different sensors [7]. A configuration of the module is shown in Figure 1c with the feature in the middle to be corrected. The development of these modules must be efficient and effective.

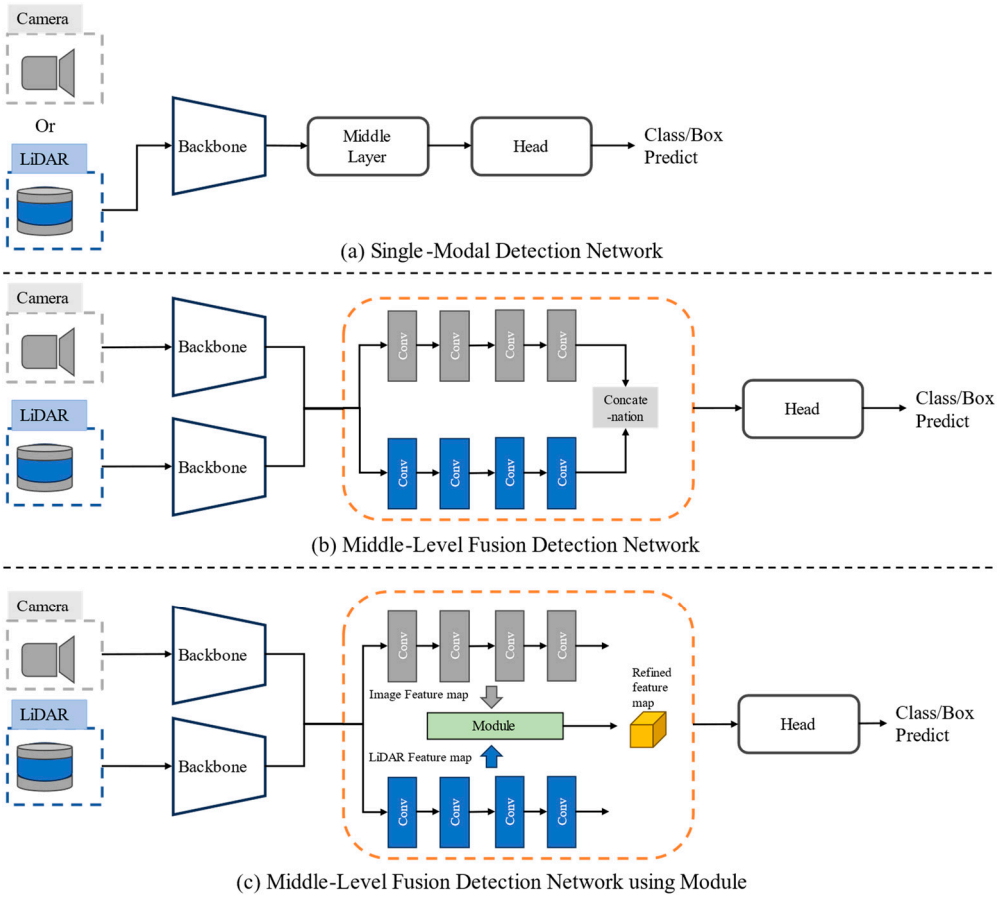

Figure 1. Comparison of the current object detection network frameworks: (**a**) Object detection with one sensor dataset. (**b**) Configure each backbone to process two or more features in the middle layer. The features of each backbone are fused at the middle layer. (**c**) Configure a module using the multi-modal fusion method to process specific features and perform refined calibration.

Efficient network metrics [8] in object detection include quality and footprint. Quality metrics, such as mAP (mean average precision) [9], precision, and recall, can evaluate and develop network effectiveness. As the data grow, the difficulty of learning increases, and it takes much time to check quality through repeated experiments. Existing studies have developed effective deep learning, and we intend to go further in existing research to resolve learning difficulties. The attention mechanism module shows visible performance improvement in various fields [7,10] with a slight increase in computation resources. We can improve performance and solve efficiency issues by utilizing the attention mechanism well.

Figure 2 shows the number of epochs needed for convergence using the same attention mechanism module as previous studies, AFAM and FSL [11,12]. We found out that the training epochs varied. AFAM, which uses contrastive learning, converges faster due to using a separate dataset for contrast learning. Our experiments suggest that separating datasets can affect the convergence speed, which can be a helpful idea in the optimization process. We propose a method to create a network module that uses neural architecture search (NAS) and utilizes separated adversary weather data. NAS methods [13] include evolutionary [14–16] and learning methods [17]. Evolutionary methods apply a genetic algorithm (GA) when solving multi-objective problems with multiple objective functions to optimize. GA creates offspring and solves the overall objective function through the population, crossover, and mutation processes. The population process in GA makes chromosomes considered for diversity. NSGA-NET [14] was proposed as a NAS method using GA. Existing NSGA [18] research has strong randomness in the population process, so

creating a learning module takes a long time. In addition, NSGA-based EEEA [16] proposed an optimization method by adding a heuristic and exiting early based on parameter values below the threshold. However, in this method, it is critical to set a standard for setting a specific threshold, and the standard can only be set through experimentation. So, in this paper, we optimized the population process to reduce randomness via a reinforcement learning network. The reinforcement learning network is divided into training and utilization, and the dataset composition varies for each stage. The contributions of this paper are as follows.

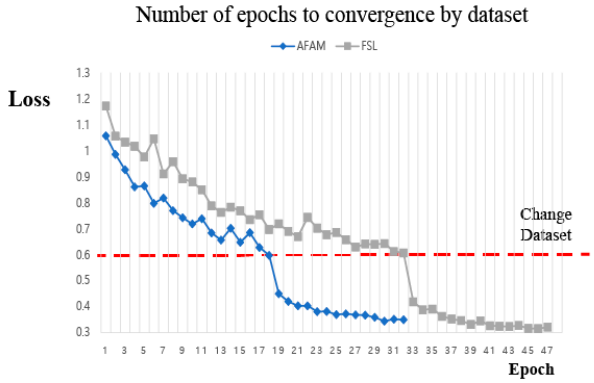

Figure 2. Number of epochs to convergence. Blue: AFAM; gray: FSL. Network's loss convergence is faster when two different networks add adversary weather data.

- The model training of the reinforcement learning network was optimized by dividing it into two stages.
- The search time to create a practical network module was shortened by reducing the search space using reinforcement learning in GA's population process.
- Performance and network convergence speed are improved based on GA assisted with reinforcement learning.

This paper applies GA to improve performance and computation resource issues and solve multi-objective problems for fast convergence. Our core contribution is to propose a method for GA's search space problem using a reinforcement learning network rather than random offspring. This method can effectively obtain a solution close to global rather than local optimal. As a comparative experiment, it was demonstrated by comparing the creation of a network module using GA-based NSGA and reinforcement learning support GA.

2. Related Works

Deep learning networks have made significant progress in object detection and classification [19] in computer vision. Recently, researchers have studied a network module that refines features using squeeze and excitation [20] or attention mechanisms. The CBAM [10] method proposes refined features by emphasizing essential features for each element in each feature map using channel and spatial attention. Wang et al. [21] improved performance by configuring efficient modules using channel and spatial attention for different domain alignments. The attention mechanism can be applied to other dimensions, and the ReFID [22] study proposed learning in the frequency domain by utilizing the attention mechanism in the frequency domain, improving generalization ability, and proving it through extensive experiments. The Transformer [23] is an attention mechanism that has been used in various fields and has achieved several state-of-the-art results [24–26]. However, its high computation requirements limit its use in embedded environments. These studies [10,20] can be applied to various fields such as speech recognition [27], audio–video speech [28], and image captioning [29]. Our research aims to improve performance with minimal computation. The SCConv [30] method generates a feature map highlighted from channel and spatial information, but it has the limitation of determining the ratio through

experimentation. We will create a module using attention research for this multi-modal network fusion.

In NAS, there were evolutionary methods [15] and reinforcement learning [17] methods, and research was actively conducted based on NASNet [31] proposed by Zoph. Among various approaches, the cell-based method by Bender [32] has been applied as a simple way to implement NAS, and in particular, DARTs [33] enable network design more efficiently than before by optimizing search space. It is possible to quickly derive optimal results by defining and learning various cases in advance, but computing power is required. Evolutionary NAS [34,35] has been efficiently utilized in multiple-objective problems aimed at operating in various environments, reducing computation, and improving performance. NSGA-NET [14], an existing study of evolutionary methods, effectively solved multiple-objective problems and showed the potential for performance and an efficient approach. We changed the search strategy to NSGA-Net and found a structure with faster convergence through the optimized population process.

Over the past decades, neural network-based object detection research [36–38] has been conducted using cameras as single sensors. Object detection networks for autonomous driving rely on multi-sensors, making camera, LiDAR, and radar fusion research necessary [4,39]. Research on sensor fusion may be a matter of considering the domain or fusing various views. The study proposed [40] learning complex correlations through graph collaboration in a multi-view problem, and at the same time, it designed the learning convergence problem as an optimization problem and proved fast convergence using the Lagrange multiplier method. A neural network based on fusion techniques [6,39,41–43] can enhance the limited data from far-off LiDAR sensors by incorporating detailed information from cameras or depth estimates. This integration can improve the accuracy of vehicle volume and heading estimation by incorporating shape and spatial data. The network fusion of multi-modals has different roles and requires alignment in the network [20]. In a study [7] that effectively fused multi-modals such as voice, video, and motion, filtering was applied to each sensor feature using attention [10] on the channel and spatial information, and then, features of the different modals were combined through a squeeze and excitation network (SENet) [20]. Since data fusion cannot simply stack different features, it can be refined using attention and SENet, as in previous studies.

In previous extensive research, the measurement deviation of LiDAR [44–46] in adversary weather situations, that is, high-humidity situations, was analyzed, and object classification through noise filtering using CNNs was proposed [47]. Camera-based adversary weather studies [48,49] analyzed clear and fog differences using noise to create training data by making insufficient data synthetically. In research on adversary weather, an important element is analyzing the characteristics of noise according to the situation of each sensor. In adversary weather, as a study, domain adaptations [50] learn only daytime and sunny weather data; evaluate learning on fog and rain, which is the target domain; and aim to learn a generalized classifier by proposing domain-invariant representations [51,52]. However, since such research requires a target domain, domain generalization research has recently been proposed [53]. So, domain generalization does not require target data, enabling more generalized network learning, but since it uses only the source domain, there is a performance discrepancy in learning. Existing studies on reducing this domain gap can be considered a very effective method in adversary weather or when there is a lack of data, but the difficulty of network learning must be considered.

3. Method Overview

We are working on designing network modules that can handle middle-level fusion and deliver reliable performance even in adverse weather conditions. However, finding a suitable network module can be time-consuming and experimental. To address this, we are focusing on improving the efficiency of both gene generation and convergence modules. First, we have improved generation efficiency by introducing chromosomes based on reinforcement learning to the gene generation of the GA algorithm. The reinforcement

learning network aims to optimize the gene generation process, and we aim to achieve this within minimal training time. However, the reinforcement learning process can take longer if data deviations increase. So, we have excluded adverse weather data from the source data used for training in reinforcement learning.

In Figure 3a, the process of creating GA involves reinforcement learning. However, the adversary weather data is not included in the source data used for training. Reinforcement learning networks are designed to optimize the gene generation process to minimize training time. However, if the variance in the data increases, the reinforcement learning process may require more training time. During Stage I, reinforcement learning learns the source dataset and predicts the loss in the next epoch for each module generated via GA. This prediction helps forecast the network's convergence speed. Reinforcement learning predicts the loss, modifies the module with actions, and receives a penalty if the loss is not improved as much as predicted.

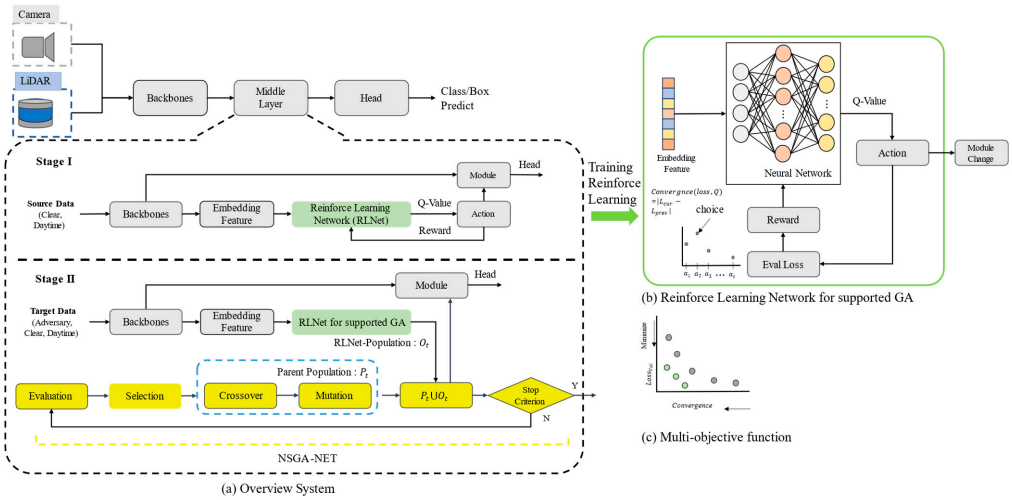

Figure 3. Description of the overall structure of the proposed method. (**a**) Modifying the middle layer in the object detection network. Stage I is the training stage of the reinforcement learning network. Stage II supports the GA's regeneration process using RL in the network module design process. Yellow is the NSGA-NET method. (**b**) The reinforcement learning training process consists of evaluating action and loss and giving a reward. (**c**) Designing multi-objective functions with improved performance and convergence.

In Stage II, a module is created using the NSGA-NET algorithm proposed in [14]. The method generates a gene population by combining the module set O_t proposed through reinforcement learning with the parent population P_t generated via NSGA. The proposed module is obtained through sampling from the rescued gene population. This approach enables simultaneous search and learning by the network's training with the head. Some deviations are expected as the data were collected during adversary weather conditions. However, the optimization process of GA can prevent the problem of falling into local optimality. Furthermore, the module estimated from the source data of reinforcement learning can reduce search time and support faster convergence speed.

4. Efficient Module Design Method

4.1. Genetic Algorithm for Neural Architecture Search

We are looking for modules that can refine and highlight the features of the camera and LiDAR backbone. The features extracted from the backbone are F_C^i and F_L^i, where i is the number of features extracted from the backbone layer. The features extracted from the backbone are denoted as $F_{C,L}^{(i)}$, where C and L refer to the camera and LiDAR. A feature $F_{C,L}^{(i)}$ is of size $W \times H \times D$, where W(width) and H(height) are the spatial dimensions, and

D(depth of feature tensor) is the depth of the backbone network; the feature $F_{C,L}^{(i)}$ will become $F_{M\_C,L}^{(i)}$ of a specific size through the input and calculation process to the module $M_{C,L}^{(i)}$. $F_{M\_C,L}^{(i)}$ is the i-th operation result of module $M_{C,L}^{(i)}$, denoted as $W^{'}, H^{'}, D^{'}$, and will be different from the input feature W, H, D. So, we applied channel attention to $F_{M\_C,L}^{(i)}$ and converted it into a feature map of a certain size. Since channel attention uses squeeze and excitation, if D' of $F_{M}^{(i)}$ is smaller than D, it is expanded, and in the opposite case, it is squeezed to create a feature map of a certain size. It is expressed as Equations (1) and (2).

$$F_{M\_C}^{(i)} = M_C^i\left(F_C^i\right), \tag{1}$$

$$F_{M\_L}^{(i)} = M_L^i\left(F_L^i\right), \tag{2}$$

The highlighted feature map is created by applying elemental-wise multiplication to the existing input feature map to the channel attended result $F_{M\_C,L}^{(i)}$, which is given in Equations (3) and (4):

$$F_C' = M_C^i\left(F_C^i\right) \bigotimes F_C^i = F_{M\_C}^{(i)} \bigotimes F_C^i, \tag{3}$$

$$F_L' = M_L^i\left(F_L^i\right) \bigotimes F_L^i = F_{M\_L}^{(i)} \bigotimes F_L^i, \tag{4}$$

where $\otimes$ denotes element-wise multiplication. By adding the recalibrated feature map $F_{C,L}'$ and the backbone features $F_{C,L}^i$, a feature map with highlighted features is created. Equations (3) and (4) are highlighted feature maps, and the i-th modules and the results for each sensor are concatenated and delivered to the layer for object detection. The output of the module created in this paper is delivered to the head or middle layer through the process of Equations (1)–(6).

$$F_C'' = F_C' + F_C^i, \tag{5}$$

$$F_L'' = F_L' + F_L^i, \tag{6}$$

The genetic algorithm of the proposed method must define encoding, selection, crossover, and mutation, and reinforcement learning is used as an aid in the gene reproduction process using these. To find an efficient and effective module, similar to the NSGA-NET [14] process, we made the pop size smaller and optimized it by changing the selection process to reduce GPU-Days.

(1) Encoding: In the proposed method, each iterative process gradually builds up a good group. The middle layer consists of blocks and nodes, and Figure 4 shows the encoding process for each block. In this study, the node of the block is expressed as $x_{C,L}^{(i)}$, where the subscript indicates the sensor and the superscript indicates the location of the block. Each block can consist of up to m or n nodes. Each $x_{C,L}^{(i)}$ node is a basic computation based on VGG [54], ResNet [1], and DenseNet [55] in the CNN literature. We chose a computation operation from among the following options, collected based on their prevalence in the CNN literature (1–8).

1. 1×1 max pooling;
2. 3×3 max pooling;
3. 1×1 avg pooling;
4. 3×3 avg pooling;
5. 3×3 local binary conv;
6. 5×5 local binary conv;
7. 3×3 dilated convolution;
8. 5×5 dilated convolution.

$x_{C,L}^{(1)}$ refers to the camera or LiDAR feature map and is the 1st conv block. A node consists of a total of 6 bits; 3 bits are the operation described above and the remaining

3 bits are the output channel size. When the GA algorithm determines only channel and operation, the kernel size may be larger than the input, and in this case, it is skipped without operation.

(2) Selection: Selection plays an important role in the process of GA finding the global optimal solution, and the tournament method was mainly used. Selection results due to fitness bias do not always provide us with a global optimal solution and can also have the disadvantage of widening the search space. In the proposed method, the best candidates were derived in terms of performance and calculation time, respectively, so that nondominant solutions could be passed on to the next generation. Then, it is to be noted once again that a certain probability is also added to the outcome proposed by the reinforcement learning network later.

(3) Crossover: Fitness is improved by stacking blocks through selection based on fitness bias and gene combination process crossover. The building block represents various node connection combinations, and the various connections optimize the conv block. Our proposed method used a one-point method, randomly selecting a node in the encoding vector and changing the designated point. If the crossover is diverse, the search space expands and a global near optimal can be found. However, to reduce the search time, a simple method is used and supplemented with reinforcement learning.

(4) Mutation: To improve diversity, there is mutation as a gene generation process that deviates from the local optimal solution during the population process, and we changed the connection node by flipping the bits of the encoding vector. The bits of the encoding vector are reversed and the operation or output channel at the connection node is changed. The proposed method only changes the form of the operation or channel, but it is difficult to find a new structure. However, in this study, a reinforcement learning network was able to solve the problem of diversity by proposing a new structure.

(5) Reproduction: The softmax method of selecting the reproduction process, like the method proposed by DARTs [33], is suitable when using only genetic algorithms. To form an optimal gene population, we must appropriately combine genes generated through reinforcement learning with previous generations during reproduction. A reasonable moment when reinforcement learning genes should be sampled is when the update size of the loss is small.

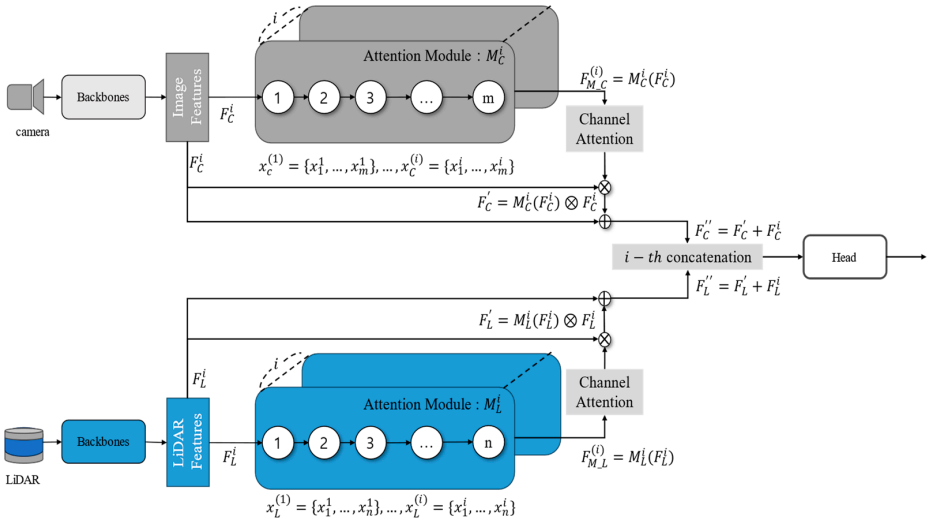

Figure 4. Encoding: illustration of a middle-layer network encoded by $x = x_s$, where is the connection in the block (gray, blue boxes, each with a possible maximum of 6 nodes). See Section 4.1 for a detailed description of the encoding schemes.

Algorithm 1 is the module creation process for GA-based NAS. Genes are organized into modules, and fitness functions for the initial genes are calculated. Dominant genes are sampled through binary tournament selection and reproduced as the size of K. Genes

are updated by adding modules generated via RLNet or crossover and mutation processes within a generation. After breeding, based on the evaluation of an objective function favoring fitness, nondominant genes are selected and passed on to the next generation, updating the generation.

Algorithm 1: Population Strategy of Genetic Algorithm using Reinforcement Learning Network

Input: Max, number of generations G, Size of crossover selection K,
Crossover probability P_c, Mutation probability P_m,
Reinforcement Learning Network (*RLNet)*
Output: Parent population PoP

Stage II)
$g \leftarrow 0$ //initialize a generation counter.
$RL(Prob) \leftarrow 1$ //initialize the control parameter for exploration.
$P \leftarrow$ initialize the parent population by uniform sampling
$f \leftarrow Evaluate(P)$
$[F_1, F_2, \ldots] \leftarrow NondominatedSort(f, \tilde{f}(P))$
while $g < G$ **do**
  k $\leftarrow 0$ // initialize an individual counter
  O $\leftarrow \emptyset$ // offspring is created in each iteration k.
  $p \leftarrow BinaryTournamentSelction$ (P, $[F_1, F_2, \ldots]$, K)
  while $k < K$ **do**
    if $rand() > RL(prob)$ **then**
      $o \leftarrow Crossover(p, P_c)$
      $o \leftarrow Mutation(o, P_m)$
    else
      $o \leftarrow$ sample an offspring from *RLNet*. // reinforcement learning network
    end
    $O \leftarrow O \cup o; k \leftarrow k + 1$
  end
  $f' \leftarrow Evaluate$(O)
  $[F_1, F_2, \ldots] \leftarrow NondominatedSort(f \cup f', \tilde{f}(P) \cup \tilde{f}(O))$
  dist $\leftarrow CrowdingDistance(F_1, F_2, \ldots)$
  $P \leftarrow Selection$ ($P \cup O$, $[F_1, F_2, \ldots]$, *dist*, K)
  $g \leftarrow g + 1$;
end
update $RL(Prob)$ according to Equation (7).
Return parent population PoP

$RL(prob)$ refers to the sampling probabilities obtained from the learned reinforcement learning model. The larger the difference between the current loss and the loss estimated by the reinforcement learning model, the more reasonable it will be to change the module. The value of w^{tabu} for the *i-th* block increases if it has been changed frequently, thus increasing the stability of the creation process by avoiding changes to the block. This reduces frequent changes during the creation process. To shorten the search time and minimize frequent changes, the $Epoch$ is also proposed. A flowchart and pseudocode outlining the overall approach are shown in Figure 3 and Algorithm 1, respectively.

$$RL(prob)_{(t)}^{(i)} = \frac{loss(Cur) - loss(Predict)}{w^{tabu} * Epoch}, \tag{7}$$

4.2. Reinforcement Learning Network for Supported Genetic Algorithm

The reinforcement learning network (RLNet)'s action is to choose changes that can significantly improve the module. Reinforcement learning changes the gene encoding to change its structure. The difference with GA is that GA does not consider improved

convergence speed, while reinforcement learning considers the convergence speed for improved performance. The proposed method can efficiently find fast and effective modules by separating roles to find structures that can achieve fast convergence.

In Algorithm 2, RLNet is used as a training algorithm for Stage I. To begin, determine the required M and capacity N for reinforcement learning. Next, initialize the Q function and network weight. The memory holds the updated information, improved loss, and extracted features as inputs. s_1 sets each state θ_1, which is a feature extracted from the input camera and LiDAR backbone. The DQN-based [56] action A_t is taken from the CNN literature mentioned above. RLNet training is stopped once the loss is lower than a predetermined threshold. The population can be utilized when the change in loss decreases, and the backbone may have different standards for this.

$$Policy^{*}(s) = arg_{a}\max\{Q(a,s)\}, \tag{8}$$

$$Q(a,s) \leftarrow Q(a,s) + \alpha \cdot \left(r_s + \gamma max_{a'} Q\left(a', s'\right)\right), \tag{9}$$

Q is a state-dependent action function, and Equation (8) was constructed to maximize Q. The reward of reinforcement learning is Equation (9), also constructed to maximize Q. There is a certain probability that it will act the opposite of r_s, and this value is minimal. The action that maximizes Q is determined at time t, and this is the CNN literature that will be changed to A_t.

The action A_t plays a crucial role in calculating rewards and penalties when it comes to reinforcement learning. This type of learning involves taking actions and updating rewards and penalties accordingly to achieve a specific goal. In the case of this paper, the goal of RLNet is to determine whether the loss has decreased more or less than expected. To make this determination, the gene module P_{t-1} and the O_t module, both of which are affected by the action, must be saved and evaluated through learning. Reinforcement learning reward Equation (10) is determined through loss prediction.

Algorithm 2: Reinforcement Learning Network Training Method

Input: Embedding feature θ_k **Output:** Action A_t

Stage I)
Initialize replay memory M to capacity N
Initialize action-value function Q with random weights
Initialize *threshold*
Initialize update-iter
for episode = 1, *max* **do**
  Initialize sequence $s_1 = \{\theta_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for t = 1, T **do**
    With probability e selection a random action a_t
    Otherwise select $A_t = max_a Q^*(\phi(s_t), a; \theta)$
    Execute action A_t in source dataset
    Change the module O_t// select CNN literature
    Evaluation of object detection networks with modified modules (P_t, O_t)
    Training object detection network
    if update-iter **then**
      Observe reward according to Eq. (11).
      Update *Reinforcement learning Network (RLNet)*
    else
      Freeze *Reinforcement Learning Network (RLNet)* Weight
    end
  end
  if *Loss* < *threshold* **then**
    Stop training *RLNet*
    Next Stage II
  end
end

The purpose of the reinforcement learning network is to improve loss.

$$Decision = Diff\big(Loss_{Avg}(P_{t-1}, O_t)\big) = Loss_{avg}(P_{t-1}) - Loss_{avg}(O_t), \tag{10}$$

$$Rewards = \begin{cases} 1\ if\ Decision > 0\ and\ Loss(P_t) < Loss(O_t) \\ 0.5\ if\ Decision > 0\ and\ Loss(P_t) > Loss(O_t)' \\ Penalty = -1\ if\ Decision < 0 \end{cases} \tag{11}$$

Rewards are determined by comparing the learning loss of the previous module with that of the currently changed module. Reinforcement learning aims to find module configurations that converge faster than genetic algorithms; therefore, if the module O_t composed of predictions has more significant improvement than the gene module P_t, a reward of 1 is given. If reinforcement learning can accurately predict loss, imposing a penalty when the current module is improved is desirable. However, in this paper, the goal is to search modules efficiently. If a penalty is given just because the prediction module O_t has improved less than the gene module P_t, the performance of the reinforcement learning-based module may decline. So, there is an improvement, but the current module converges faster, and only half the reward is given. A penalty is given when the loss is updated without improvement.

4.3. Search Strategy for Fast Convergence Module

To find a globally optimal solution to an optimization problem, designing the objective function and constraints is important. The problem with NAS is that the search space for the global optimal solution has a wide range of objective functions and constraints, so it takes a long time. In this study, GA-based NAS was selected to solve problems with fast convergence and performance, and the search space is still enormous. To assist with this, we proposed a NAS algorithm using reinforcement learning. The important point is that if the value is large based on a certain probability $RL(prob)$, the structure proposed using RLNet is added to the sample and then, the optimal solution is found using nondominated sort. Since the module proposed via reinforcement learning may not always be nondominant, all proposed outputs must be evaluated. The fitness function evaluates performance and convergence, Frames Per Second (FPS) and is given in Equation (12).

$$Fitness_{GA\_RL} = -\alpha * Loss + \beta * FPS + \gamma * Convergence(loss, Q), \tag{12}$$

Loss is an improvement of the network and includes classification and box regression for detection as learning. As learning improves, the loss should become smaller. So, we gave it a negative value so that it improves as the value becomes smaller. We think about calculating fitness for convergence like making a prediction. For a prediction, two methods were used in parallel. First, we compared the current loss and the previous loss using Equation (13), as used in reinforcement learning.

$$Convergence(loss, Q) = |L_{cur} - L_{predict}|, \tag{13}$$

Our reinforcement learning predicts loss improvement for each work of the CNN literature (1–8). Our learning strategy is to predict the loss for each configuration change and select the module with the highest change. $Convergence(loss, Q)$ included a small value as a fitness function to control uncertainty in the calculated value. Through this equation, the fitness function can contain the possibility of improvement for the next generation. Lastly, *FPS* refers to calculation speed, and as the amount of calculation increases, *FPS* decreases. Our goal is to maximize the fitness function.

5. Experiment Setup and Evaluation of the Proposed Method

In this section, we first describe our implementation details. Then, we compare our proposed NAS with NSGA-Net and other modules. NSGA-Net compares the efficiency and

performance of the module creation process. GPU-Day is used to evaluate the efficiency of the production process.

5.1. Implementation Details

The search itself is repeated five times with different initial random seeds. We use the EfficientDet [57] loss function for learning the associated weights for each architecture. All experiments are performed on Nvidia 3080Ti GPU cards. The experimental dataset used the Dense dataset [4], and the composition of the train, validation, and test sets is summarized in Table 1. Stage I uses only the T1 dataset, while Stage II utilizes all datasets. In other words, the T1 dataset refers to the source data described in the Method Overview Section, and the T1-T6 datasets refer to the target data.

Table 1. Dataset size used for training, testing, and validation.

Dataset	Environmental Condition (Light, Weather)	Training	Validation	Testing
T_1	Daytime, Clear	2183	399	1005
T_2	Daytime, Snow	1615	226	452
T_3	Daytime, Fog	525	69	140
T_4	Nighttime, Clear	1343	409	877
T_5	Nighttime, Snow	1720	240	480
T_6	Nighttime, Fog	525	69	140
	Total	8238	1531	3189

The performance is evaluated using mean average precision (mAP). We have applied the PASCAL VOC 11-point interpolation method to compute the average precision (AP) for each class. Later, the average is computed using mean average precision across all the classes. In our case, there are two object class labels, i.e., pedestrian and vehicle. So, we compute the AP for each class using Equation (14):

$$AP_{label(obj)} = \frac{1}{11}\sum_{r\in\{0,\ 0.1,\ \ldots,\ 1\}} P_{interp}(r), \tag{14}$$

where $label = \{vehicle,\ pedestrian\}$, obj is the index of class values in the $label$, and P corresponds to the precision at each interpolated recall r. The mAP is computed using Equation (15). The IoU threshold was set to 0.5.

$$mAP_{0.5} = \frac{1}{n}(AP_{Pedestrian} + AP_{Vehicle}), \tag{15}$$

Here, n is the number of class labels.

Figure 5 shows the specific structure of the network used in the experiment, using EfficientDet. EfficientDet proposed a BiFPN layer to efficiently and quickly converge learning. In this paper, a comparative experiment was conducted by adding a module to the EfficientDet structure for efficient module testing. The overall structure of the object detection network used EfficientDet, but to prove its effectiveness, experiments were conducted using ResNet [1], Efficientnet [58], and ResT [59] backbone.

In order to add a module to EfficientDet for module evaluation, the feature map size had to be changed to a specific size. We created a feature map of a specific size using channel attention used in CBAM. Figure 5a illustrates the process and (b) indicates the specific feature map size used in the experiment. k is the kernel size of 2d convolution in Figure 5b. First, from the bottom, we calculate $F_{C,L}^{(i)}$, a recalibration process for a total of three feature maps. The detailed process of $F_{C,L}^{(i)}$ is described in Section 4.1, and it was concatenated and input into the BiFPN layer to use the EfficientDet structure without change. EffcientDet

extracts five feature maps from the backbone and inputs them into the BiFPN Layer, but in this paper, modules for three feature maps were created. When applying GA to find a CNN-based module, the sizes of W and H of the fourth and fifth feature maps were tiny, so it did not help improve performance. The image used RGB data, and the LiDAR data used the Range Image proposed by Laser-Net [60], which is a projection of the size of the image. Image pixels were resized to 896 for learning and evaluation.

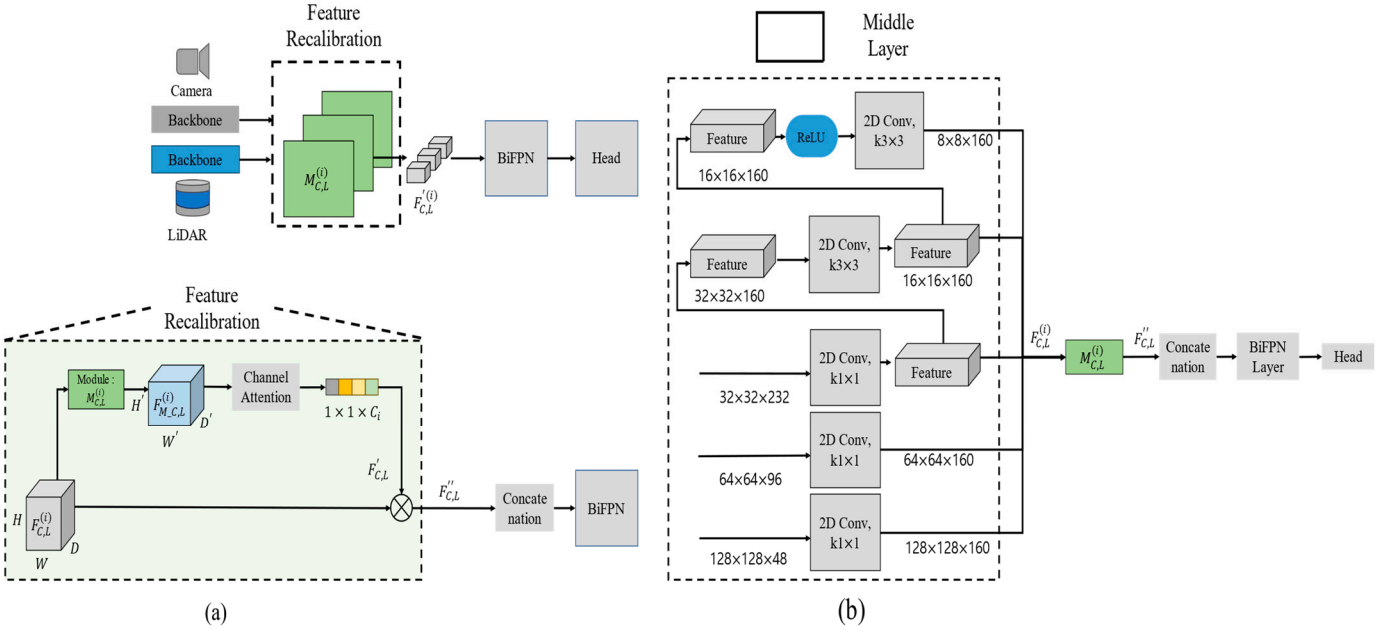

Figure 5. Describes implementation details. Backbone is displayed in the table with the EfficientDet structure. (**a**) Attention techniques are used as a process to fuse features of different sizes within the middle layer. (**b**) Size of experimental feature map.

Table 2 shows the experimental hyperparameters. Search space is the parameter of the initial GA gene block, and the block composition consists of up to six nodes. The initial channel is extracted from the backbone as described in Figure 5b and is a conv channel. Learning optimization is a dropout and update method to prevent the overfitting of learning, and when the dropout rate $(r_{dropout})$ is less than 0.5, overfitting problems often occur. Adam [61] optimization was applied to gradient descent for network learning, and the search strategy is a parameter for the regeneration process of GA. As the population size grew, there was a problem of overfitting or increased learning instability, and in this paper, experiments showed that the number 10 was the most stable and that learning and network architecture search functioned normally. Crossover and mutation are the probabilities of the combination and the change of two selected genes, and α, β, and γ are the effects of each element in the fitness evaluation and have the greatest evaluation of loss.

Lastly, as reinforcement learning parameters, alpha and beta suggest the ratio of loss and convergence of reinforcement learning predictions. A batch is not a batch of images and LiDAR data; it refers to the size of the episode. What is remarkable is that batch was used in reinforcement learning to increase learning stability. However, the batch was not used in the NAS module creation and object detection network learning. To evaluate the creation time (GPU-Days) and learning convergence time of an object detection network, the influence of batch must be reduced.

Table 2. Hyperparameter setting.

Categories	Parameters	Settings
Search space	Initial channel	160
	Maximum nodes	6
Learning optimizer	Dropout rate $\left(r_{dropout}\right)$	0.5
	Maximum epochs	500
	Learning rate schedule	ADAM [61]
Search strategy	Population size	10
	Crossover probability	0.3
	Mutation probability	0.1
	α (Loss)	0.8
	β (FPS)	0.1
	γ (Convergence)	0.1
Reinforcement Learning	Batch size	128
	Gamma	0.99
	ε^{start}	0.9
	ε^{end}	0.05
	τ	10,000
	Learning rate	1×10^{-4}

5.2. Effectiveness and Efficiency of Proposed Method

Table 3 shows the results of an experiment by changing the backbone, comparing the method proposed in the modified EfficientDet-b3 with the existing hand-crafted module. In addition, we evaluated performance, FPS, and convergence speed by comparing the existing modules CBAM and AFAM, which were manually designed by experts. By changing the backbone to ResNet and ResT, we verified whether module search using reinforcement learning efficiently generates modules even if the backbone is changed. CBAM [10] is an efficient attention mechanism that combines features emphasized in channel and spatially to improve performance in various tasks through refined calibration. The proposed module improved performance compared to the CBAM module, reduced network learning time, and improved performance.

Table 3. Result of object detection with different existing modules and backbones on Dense dataset.

Backbone	Fusion Module (C, L)	FPS (Frame/s)	Top1-mAP (%)	Number of Epochs to Convergence
ResNet	None	15	0.275	16
	CBAM	12	0.278	28(+12)
	AFAM (CBAM + MMTM)	10	0.293	29(+13)
	Propose	11	0.292	20(+4)
Efficientnet-b3	None	15	0.414	28
	CBAM	12	0.388	18(−10)
	AFAM (CBAM + MMTM)	10	0.419	27(−1)
	Propose	11	0.434	24(−4)
ResT	None	5	0.247	35
	CBAM	4	0.299	40(+5)
	AFAM (CBAM + MMTM)	4	0.319	45(+10)
	Propose	3	0.334	31(−4)

We conducted a comparative experiment with the existing research, AFAM, which requires contrastive learning and uncertainty calculation. In this paper, only the module's structure was used without calculating the uncertainty of AFAM. The module configuration of AFAM is a fusion of CBAM and MMTM and determines the camera's or LiDAR's weight according to uncertainty to construct a recalibrated feature map. Here, without calculating

uncertainty, the features $F_{C,L}$ extracted from each backbone were input to the AFAM module, and then, two recalibrated feature maps F'_C and F'_L with weights for the two sensors were created. None is a concatenation method without an intermediate module, corresponding to (b) in Figure 1.

In the case of ResNet, concatenation at the intermediate level is more efficient than configuring modules. Experimental results show performance improvement when adding modules, and overall learning time increases. Efficientnet-b3 is the best backbone for the EfficientDet structure. Therefore, the performance and convergence results were the most efficient. CBAM focuses on features for each sensor, and AFAM proposes methods to recalibrate backbone features extracted from both sensors. When extracting with the Efficientnet backbone, recalibrating the two features performed excellently. Based on the experimental results, the features of the two sensors increase by 0.3% when recalibrating. In Transformer-based backbone ResT, MLP-based CBAM and AFAM improved learning performance, but the learning time increased accordingly. Only modules found through NAS achieved faster performance and learning convergence. Unlike ResNet, we were able to confirm that the attention module achieves efficient performance improvement in the Transformer-based backbone.

Table 4 shows the experimental results to verify the effectiveness of the module. In this paper, we designed a module that uses NAS to support object detection networks in adverse weather environments. The comparative experiment used intermediate fusion methods and EA-based NSGA-NET and EEEA. In the case of EEEA, when looking for a lightweight model by setting a specific threshold in NSGA-NET, an early exit was proposed to reduce GPU-Days dramatically. However, as a result of applying it through experiments, in order to set the threshold, it was necessary to perform learning at least once and find an appropriate threshold for early exit to operate. Otherwise, there was no difference between NSGA-NET and GPU-Days. In Table 4, only the best results are written.

Table 4. Evaluating the efficiency and effectiveness of modules created using evolutionary methods.

Backbone	Module Search Method	Components	GPU-Days	FPS (Frame/s)	Top1-mAP (%)	Number of Epochs to Convergence
ResNet	None	-	-	15	0.275	16
	NSGA-NET	GA	4	11	0.288	22(+6)
	EEEA	GA + Rule	0.2	13	0.280	17(+1)
	Propose	GA + RL	3	11	0.292	20(+4)
Efficientnet-b3	None	-	-	15	0.414	28
	NSGA-NET	GA	4	11	0.428	25(−3)
	EEEA	GA + Rule	0.2	12	0.381	22(−6)
	Propose	GA + RL	3	11	0.434	24(−4)
ResT	None	-	-	5	0.247	35
	NSGA-NET	GA	8	3	0.322	33(−2)
	EEEA	GA + Rule	0.5	4	0.283	30(−5)
	Propose	GA + RL	7	3	0.334	31(−4)

In the case of ResNet, the modules found based on genes improved performance like the hand-crafted modules CBAM and AFAM, but the overall learning time increased. In the case of Efficientnet-b3, the performance of the attention module has been improved, and the convergence speed can also be confirmed to have been increased. Through experiments, it can be confirmed that the attention module operates most effectively in the EfficientDet structure. Similarly, in ResT, the attention module shows significant results in terms of learning time and performance improvement.

Overall, through all experiments, it was confirmed that ResNet's backbone attention module improves performance but increases learning time. In the case of Efficientnet and ResT, the configuration of the attention module was able to simultaneously increase

learning time and performance. In the case of CBAM, the results show deviations such as performance decreases or increases, while AFAM has fewer deviations, but the learning time increases. In the case of NAS based on a genetic algorithm, NSGA-NET can reliably find modules. However, it takes a lot of GPU-Days and to solve this problem, EEEA has been proposed. Nevertheless, in EEEA, there were many cases where no performance improvement was achieved or the extent of the improvement was minimal. For early exit, it is judged that it is not possible to find a module with fast performance and convergence speed due to limitations in parameter size. Therefore, assistance with the proposed method, reinforcement learning, can contribute to reliably finding effective and efficient modules.

5.3. Ablation Study

A limitation of NAS research is that network reproducibility is difficult. In this study, the training loss steadily decreased, but the validation loss diverged, causing frequent overfitting. The reason is that learning stability decreased as the proposed method repeated module search simultaneously with learning. As one of the stable convergence methods of learning, Dropout [62], a regulation technique, is simple but can increase learning stability. Setting a high dropout rate for convergence in the middle layer is adequate. In the case of existing AFAM and CBAM, learning was stable regardless of $r_{dropout}$, but learning was unstable in the evolutionary method.

Figure 6 shows the learning stability and performance of the proposed method for each $r_{dropout}$. When $r_{dropout} = 0.1$, performance deteriorates due to low learning stability in the NAS that learns while changing the module configuration. Overall, the stability of learning is an essential factor in module search. As the value of $r_{dropout}$ increases, the learning stability is secured, and a module with performance that meets the purpose can be found. However, at $r_{dropout} = 0.7$, the stability of learning and the convergence speed increase simultaneously, increasing GPU-Days. The problem of learning stability, a limitation of NAS research, was compensated for with dropout, but sometimes, it was impossible to find a module with fast convergence.

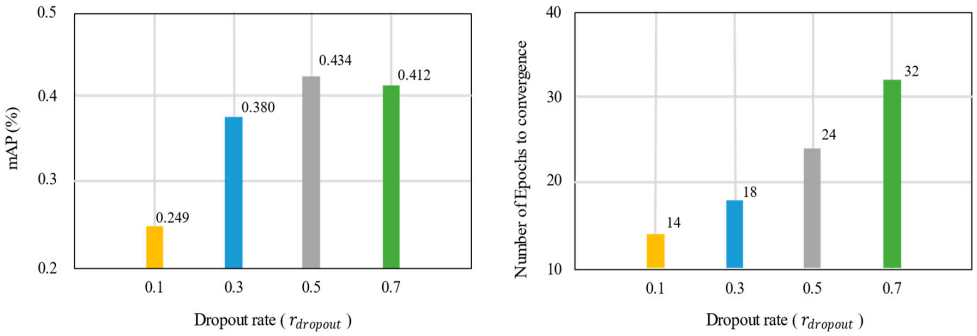

Figure 6. The proposed method achieved the best performance at $r_{dropout} = 0.5$ and experimental results by dropout rate.

Figure 7 shows the average learning time. If EfficientDet uses only cameras, it converges very quickly. However, as explained later, the performance is lower than the sensor fusion result. Our experiments were based on the simple fusion EfficientDet (C, L). Handcrafted CBAM and AFAM took a longer overall learning time than NAS. The characteristics of the two modules are that they are based on MLP, so learning takes a lot of time. The modules found through NAS comprise the CNN literature, enabling efficient learning from image-based data. For this reason, we found a module with fast learning convergence. CNN-based NAS studies such as NSGA have advantages in terms of training time.

Figure 8 illustrates the practical implications of our research, showing the results of AFAM with the slightest deviation as the average and variance values for mAP, a performance indicator from an effective perspective. CBAM and AFAM are based on MLP and may result in slight deviations. This is a significant finding as it demonstrates that

the MLP-based module can effectively reduce learning deviation. NSGA and EEEA show significant differences in performance. EEEA limits the size of parameters to find efficient modules. Therefore, performance decreases but learning time is advantageous. On the other hand, the proposed method shows similar means and deviation accuracy as NSGA. This problem is with a wide search space, such as GA-based research. In the context of object detection, where CNN-based modules may exhibit large learning deviations, this is a crucial insight. It highlights the high probability of overfitting and the need for appropriate regulation techniques such as dropout.

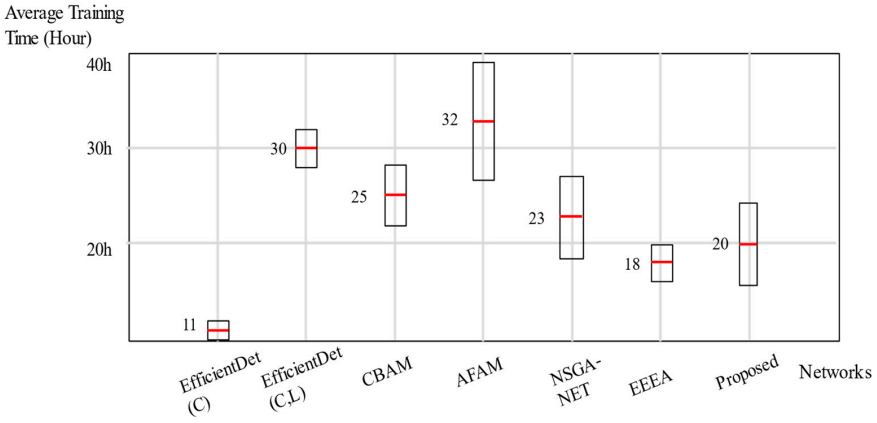

Figure 7. Networks' average training time, unit: hour. The red line represents the average, and the box represents the deviation.

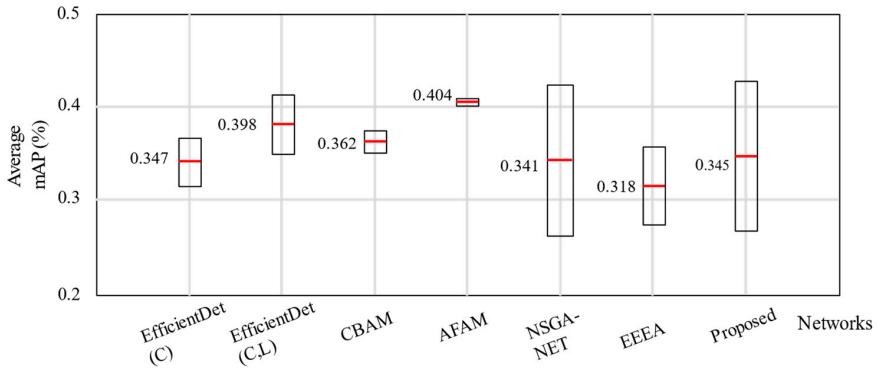

Figure 8. Network's average mAP. The red line represents the average, and the box represents the deviation.

Also, our research uncovers the complex nature of NAS studies. While they hold promise for high performance, there are instances where the performance falls short of the baseline network. This variability in learning performance presents a significant challenge. Even with the implementation of dropout, further research is necessary to identify modules with consistently good performance and to address reproducibility problems.

Next, Table 5 is an experiment to check the effect of fusion prediction on the gene generation process in reinforcement learning. Using ResNet and Efficientnet backbone in the EfficientDet structure, a comparative experiment was conducted between the cases where only loss and FPS are included in fitness and adding predicted convergence. The comparative experiments confirmed that accuracy and performance were slightly improved. Although the difference in values is not significant, it has been verified that reinforcement learning results are involved in the gene generation process by influencing the fitness function, and through this, convergence speed and slight performance improvement can be achieved. As the population grows and becomes more diverse, the impact of reinforcement learning will become much more significant.

Table 5. Result of including the prediction loss of reinforcement learning in the genetic algorithm's fitness function. w/: with.

Backbone	Fitness	Top1-Acc	Number of Epochs to Convergence
EfficientDet-b3	Loss, FPS	0.406	25
	w/Convergence (Loss, Q)	0.424	24
ResNet-EfficientDet	Loss, FPS	0.290	21
	w/Convergence (Loss, Q)	0.292	20

Figure 9a–d show the qualitative results of the EfficientDet, CBAM-EfficientDet, and AFAM-EfficientDet papers using cameras only and the proposed method. Camera-based networks and other modules may see inaccurate bounding boxes in Day, Clear situations. Since CBAM and AFAM used the MLP-based squeeze and excitation [20] module, errors frequently occurred when the vehicle was truncated. However, since the proposed method generates modules from the 2D CNN literature, it shows robust results. Object detection performs well both Day and Night in foggy situations that require fusion during adverse weather situations. In particular, we could confirm that the vehicle's bounding box was accurately found even at night. However, an interesting result is that the false detection rate increases in the 2D CNN literature-based modules like camera-based networks in Day and Fog situations.

Figure 9. Object detection performance qualitative results. (**a**) Only camera feature-based object detection result; (**b**–**d**) camera-LiDAR fusion-based result; (**b**) with CBAM module; (**c**) with AFAM module; (**d**) proposed module using reinforcement learning and genetic algorithm.

6. Conclusions

Object detection is a crucial part of autonomous navigation in dynamic environments. A lot of research has been conducted in this area, which has led to the development of object detection modules based on sensor fusion. However, these modules are challenging in handling frequent training to handle adverse weather conditions and extreme lighting changes. To address these shortcomings, a new method has been proposed which uses a genetic algorithm to ensure diversity and reinforcement learning for optimization. This method improves object detection accuracy and enables module search with fast convergence speed. Experiments conducted on a benchmark dataset show that this proposed method improves overall detection accuracy and reduces learning convergence time.

The study focuses on improving neural networks' performance and convergence speed for object detection in adverse weather conditions. The proposed method affects real-world scenarios in a learning environment with various situations and contributes to creating effective modules for object detection. However, reproducibility difficulties and deviations in learning performance are problems. In the future, this study aims to expand research to secure uniform object detection performance in a frequent data learning environment and generalize the model through the evaluation of various datasets by considering the issue of reproducibility.

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

Funding: This work was supported by the Innovative Human Resource Development for Local Intellectualization program through the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (IITP-2024-2020-0-01462).

Data Availability Statement: Data available in a publicly accessible repository [4,7,10,14,16].

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

References

1. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June 26–1 July 2016; pp. 770–778.
2. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the Computer Vision–ECCV (ECCV), Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
3. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
4. Bijelic, M.; Gruber, T.; Mannan, F.; Kraus, F.; Ritter, W.; Dietmayer, K.; Heide, F. Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–18 June 2020; pp. 11682–11692.
5. Gu, R.; Zhang, S.-X.; Xu, Y.; Chen, L.; Zou, Y.; Yu, D. Multi-modal multi-channel target speech separation. *IEEE J. Sel. Top. Signal Process.* **2020**, *14*, 530–541. [CrossRef]
6. Li, Y.; Yu, A.W.; Meng, T.; Caine, B.; Ngiam, J.; Peng, D.; Shen, J.; Lu, Y.; Zhou, D.; Le Quoc, V.; et al. DeepFusion: Lidar-Camera Deep Fusion for Multi-Modal 3D Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 17182–17191.
7. Joze, H.R.V.; Shaban, A.; Iuzzolino, M.L.; Koishida, K. MMTM: Multimodal transfer module for CNN fusion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–18 June 2020; pp. 13289–13299.
8. Menghani, G. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Comput. Surv.* **2023**, *55*, 1–37. [CrossRef]
9. Menghani, G. Microsoft coco: Common objects in context. In Proceedings of the Computer Vision–ECCV (ECCV), Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
10. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. CBAM: Convolutional block attention module. In Proceedings of the Computer Vision–ECCV (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
11. Kim, T.-L.; Arshad, S.; Park, T.-H. Adaptive Feature Attention Module for Robust Visual–LiDAR Fusion-Based Object Detection in Adverse Weather Conditions. *Remote Sens.* **2023**, *55*, 3992. [CrossRef]

12. Kim, T.-L.; Park, T.-H. Camera-lidar fusion method with feature switch layer for object detection networks. *Sensors* **2022**, *22*, 7163. [CrossRef] [PubMed]
13. Gao, S.; Li, Z.-Y.; Han, Q.; Cheng, M.-M.; Wang, L. RF-Next: Efficient receptive field search for convolutional neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 2984–3002. [CrossRef] [PubMed]
14. Lu, Z.; Whalen, I.; Dhebar, Y.; Deb, K.; Goodman, E.D.; Banzhaf, W.; Boddeti, V.N. Multiobjective evolutionary design of deep convolutional neural networks for image classification. *IEEE Trans. Evol. Comput.* **2020**, *25*, 277–291. [CrossRef]
15. Angeline, P.J.; Saunders, G.M.; Pollack, J.B. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans. Neural Netw.* **1994**, *5*, 54–65. [CrossRef] [PubMed]
16. Termritthikun, C.; Pholpramuan, J.; Arpapunya, A.; Nee, T.P. EEEA-Net: An Early Exit Evolutionary Neural Architecture Search. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104397. [CrossRef]
17. Tenorio, M.; Lee, W.-T. Self organizing neural networks for the identification problem. *Adv. Neural Inf. Process. Syst.* **1988**, *1*. Available online: https://proceedings.neurips.cc/paper/1988/hash/f2217062e9a397a1dca429e7d70bc6ca-Abstract.html (accessed on 18 June 2024).
18. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
19. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009.
20. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 7132–7141.
21. Cui, T.; Zheng, F.; Zhang, X.; Tang, Y.; Tang, C.; Liao, X. Fast One-Stage Unsupervised Domain Adaptive Person Search. *arXiv* **2024**, arXiv:2405.02832.
22. Peng, J.; Zhang, X.; Wang, Y.; Liu, H.; Wang, J.; Huang, Z. ReFID: Reciprocal Frequency-aware Generalizable Person Re-identification via Decomposition and Filtering. *ACM Trans. Multimed. Comput. Commun. Appl.* **2024**, *20*, 1–20. [CrossRef]
23. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
24. Rekesh, D.; Koluguri, N.R.; Kriman, S.; Majumdar, S.; Noroozi, V.; Huang, H.; Hrinchuk, O.; Puvvada, K.; Kumar, A.; Balam, J.; et al. Fast conformer with linearly scalable attention for efficient speech recognition. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*; IEEE: New York, NY, USA, 2023; pp. 1–8.
25. Zong, Z.; Song, G.; Liu, Y. Detrs with collaborative hybrid assignments training. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 2–6 October 2023; pp. 6748–6758.
26. Hu, J.C.; Cavicchioli, R.; Capotondi, A. Exploiting Multiple Sequence Lengths in Fast End to End Training for Image Captioning. In *2023 IEEE International Conference on Big Data (BigData)*; IEEE: Sorrento, Italy, 2023; pp. 2173–2182.
27. Li, C.; Qian, Y. Deep audio-visual speech separation with attention mechanism. In Proceedings of the ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; IEEE: New York, NY, USA, 2020; pp. 7314–7318.
28. Iuzzolino, M.L.; Koishida, K. AV(se)^2: Audio-visual squeeze-excite speech enhancement. In Proceedings of the ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; IEEE: New York, NY, USA, 2020; pp. 7539–7543.
29. Yang, X.; Zhang, H.; Cai, J. Learning to collocate neural modules for image captioning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 4250–4260.
30. Li, J.; Wen, Y.; He, L. Scconv: Spatial and channel reconstruction convolution for feature redundancy. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 6153–6162.
31. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 8697–8710.
32. Bender, G.; Kindermans, P.J.; Zoph, B.; Vasudevan, V.; Le, Q. Understanding and simplifying one-shot architecture search. In Proceedings of the International Conference on Machine Learning (PMLR), Stockholm Sweden, 10–15 July 2018; pp. 550–559.
33. Liu, H.; Simonyan, K.; Yang, Y. DARTS: Differentiable architecture search. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
34. Loni, M.; Sinaei, S.; Zoljodi, A.; Daneshtalab, M.; Sjödin, M. DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems. *Microprocess. Microsyst.* **2020**, *73*, 102989. [CrossRef]
35. Sun, Y.; Xue, B.; Zhang, M.; Yen, G.G. Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification. *IEEE Trans. Cybern.* **2020**, *50*, 3840–3854. [CrossRef]
36. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9.
37. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.

38. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, 1097–1105. Available online: https://dblp.org/rec/conf/nips/KrizhevskySH12.html (accessed on 18 June 2024).
39. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? The kitti vision benchmark suite. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Providence, Phode Island, 16–21 June 2012; pp. 3354–3361.
40. Wang, H.; Wang, C.; Li, Y.; Chen, H.; Zhou, J.; Zhang, J. Towards Adaptive Consensus Graph: Multi-View Clustering via Graph Collaboration. In *IEEE Transactions on Multimedia*; IEEE: New York, NY, USA, 2022.
41. Liang, M.; Yang, B.; Chen, Y.; Hu, R.; Urtasun, R. Multi-task multi-sensor fusion for 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 7345–7353.
42. Ku, J.; Mozifian, M.; Lee, J.; Harakeh, A.; Waslander, S.L. Joint 3d proposal generation and object detection from view aggregation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1–8.
43. Vora, S.; Lang, A.H.; Helou, B.; Beijbom, O. Pointpainting: Sequential fusion for 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–18 June 2020; pp. 4606–4612.
44. Heinzler, R.; Schindler, P.; Seekircher, J.; Ritter, W.; Stork, W. Weather influence and classification with automotive lidar sensors. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 1527–1534.
45. Sebastian, G.; Vattem, T.; Lukic, L.; Burgy, C.; Schumann, T. RangeWeatherNet for LiDAR-only weather and road condition classification. In Proceedings of the 2021 IEEE Intelligent Vehicles Symposium (IV), Nagoya, Japan, 11–17 July 2021; pp. 1527–1534.
46. Heinzler, R.; Piewak, F.; Schindler, P.; Stork, W. CNN-Based Lidar Point Cloud De-Noising in Adverse Weather. *IEEE Robot Autom. Lett.* **2020**, *5*, 2514–2521. [CrossRef]
47. Seppänen, A.; Ojala, R.; Tammi, K. 4DenoiseNet: Adverse Weather Denoising from Adjacent Point Clouds. *IEEE Robot. Autom. Lett.* **2022**, *8*, 456–463. [CrossRef]
48. Cai, B.; Xu, X.; Jia, K.; Qing, C.; Tao, D. Dehazenet: An end-to-end system for single image haze removal. *IEEE Trans. Image Process.* **2016**, *25*, 5187–5198. [CrossRef] [PubMed]
49. Hasirlioglu, S.; Riener, A. A general approach for simulating rain effects on sensor data in real and virtual environments. *IEEE Trans. Intell. Veh.* **2019**, *5*, 426–438. [CrossRef]
50. Shao, Y.; Liu, Y.; Zhu, C.; Chen, H.; Li, Z.; Zhang, J. Domain Adaptation for Image Dehazing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 2808–2817.
51. Wu, A.; Li, W.; Zhou, W.; Cao, X. Instance-Invariant Domain Adaptive Object Detection via Progressive Disentanglement. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 4178–4193. [CrossRef]
52. Wu, A.; Li, W.; Zhou, W.; Cao, X. Vector-Decomposed Disentanglement for Domain-Invariant Object Detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 11–17 October 2021; pp. 9342–9351.
53. Wu, A.; Deng, C. Single-Domain Generalized Object Detection in Urban Scene via Cyclic-Disentangled Self-Distillation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 847–856.
54. Simonyan, K.; Zisserman, A. Very deep convolutional networks for largescale image recognition. In Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
55. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
56. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M.A. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
57. Tan, M.; Pang, R.; Le, Q.V. EfficientDet: Scalable and Efficient Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 10781–10790.
58. Tan, M.; Le, Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In Proceedings of the 36th International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
59. Zhang, Q.-L.; Yang, Y.-B. ResT: An Efficient Transformer for Visual Recognition. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 15475–15485.
60. Meyer, G.P.; Laddha, A.; Kee, E.; Vallespi-Gonzalez, C.; Wellington, C.K. LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 12677–12686.
61. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
62. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.