

Received February 4, 2022, accepted March 14, 2022, date of publication March 21, 2022, date of current version March 28, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3160719

Generation of CNN Architectures Using the Harmonic Search Algorithm and Its Application to Classification of Damaged Sewer

YOUNG-WOON JEONG^{ID1}, KWEE-BO SIM^{ID1}, (Member, IEEE),
SEUNGMIN PARK^{ID2}, (Member, IEEE), JEILL OH³, AND WOO JUNE CHOI^{ID1}, (Member, IEEE)

¹School of Electrical and Electronics Engineering, Chung-Ang University, Seoul 06974, Republic of Korea

²Department of Software, Dongseo University, Busan 47011, Republic of Korea

³Department of Urban Design and Study, School of Civil & Environmental Engineering, Chung-Ang University, Seoul 06974, Republic of Korea

Corresponding authors: Jeill Oh (ohjeill@cau.ac.kr) and Woo June Choi (cecc78@cau.ac.kr)

This work was supported by Korea Environment Industry & Technology Institute (KEITI) through project for developing innovative drinking water and wastewater technologies, funded by Korea Ministry of Environment (MOE) (2020002700015).

ABSTRACT When used as an image processing method, convolutional neural networks (CNNs) cannot verify conditions that can achieve high performance; however, they show sufficiently high performance and are used in various fields. The architecture of a CNN and various parameters determine its performance, and it is impossible to verify the number of all cases that determine the performance of the CNN. Therefore, well-known CNN models are generally used. Recently, various methods for adjusting the parameters of CNNs or generating CNN architectures have been studied in various ways. Methods using metaheuristic algorithms often focus on parameter tuning, or the use of simple hierarchical architectures. This paper proposes a method to create a CNN model with a complex CNN architecture that can be applied to different datasets using the harmony search (HS) algorithm from among metaheuristic algorithms. This study aimed to generate a CNN architecture using fewer computing resources and to verify the results. To make the CNN model in units of cells, the internal and hierarchical architecture of the cell was created based on the learning of the CIFAR image dataset through HS and the performance of the created CNN model was tested by applying it to the classification of a damaged sewer pipe image dataset. The 10-GPU-day computed result showed that the accuracy of classifying the damaged sewers was at least 5% higher than that of VGGNet, the highest among the existing CNN models.

Keywords: convolutional neural network; auto machine learning; metaheuristic algorithm; harmony search algorithm; heuristic algorithm, optimization, artificial neural network, image classification.

INDEX TERMS Heuristic algorithm, machine learning, optimization, neural network, multi-layer neural network.

I. INTRODUCTION

In image classification, which made machine learning popular, CNNs [1] have become a frequently used method. CNNs, which perform well in image processing, are used in various fields, and models using various layers and connection methods are released every year. Since the emergence of AlexNet [2], which increased the popularity of CNNs, other CNN models such as GoogleNet [3], VGGNet [4], ResNet [5], and DenseNet [6] have been created. Newly

created CNN models have shown that the performance of CNNs can be improved by the features of each model. However, these models are CNN models produced by a person directly determining the type of layer, the connection architecture of each layer, and the parameters. Therefore, the performance of a CNN model depends on the type of problem to which it is applied. However, it is difficult for humans to design different CNN architectures according to each type of problem. When creating a CNN model, it is impossible to make the best model by checking the performance of each model and manually attempting to adjust the type of layer, the connection architecture of each layer, and the parameters.

The associate editor coordinating the review of this manuscript and approving it for publication was Khin Wee Lai^{ID}.

Therefore, various studies are underway to generate better CNN models. This paper proposes a method for generating CNN architectures using the harmony search (HS) [7] algorithm. This is executed using the CIFAR image dataset, and is applied to a damaged sewer pipe.

One of the methods to improve the performance of a CNN model is to tune the parameters of the layers used in the CNN. For parameter tuning, there are methods that use genetic algorithms [8]–[10], particle swarm optimizers [11]–[14], or HS [15], [16]. The parameter tuning method is slightly different for each study, but the commonality is that the type and order of the layers are fixed; and the kernel size, stride, padding, and some types of output channels of each layer are changed to obtain better results. Parameter tuning has the advantage that it takes less time because the CNN layer structure itself is fixed; however, it has the disadvantage that the performance cannot be greatly improved in the original CNN model as the layer type and order are fixed.

There have also been studies proposing a method for generating the CNN architecture itself. Studies on CNN architecture generation have been published using various methods such as neural architecture search (NAS) [19] that use reinforcement learning [17], and genetic CNNs [20] that use genetic algorithms (GAs) [18]. The following is a list of some of the CNN architecture generation methods that have been published: Meta-QNN [21], hierarchical evolution [22], large-scale evolution [23], genetic programming CNN (CGP-CNN) [24], efficient architecture search (EAS) [25], evolving deep CNN [26], advanced neural architecture search (NASNet) [27], and CNN-genetic algorithms [28]. Each study uses a slightly different method to construct the CNN architecture. The CNN architecture generation consists of several different methods; such as fixing the overall architecture of the CNN and changing only the connection relationship between layers; adjusting both the CNN architecture and the parameters of the layers; and a method to apply the CNN architecture obtained from one dataset to another dataset. Evidently, this topic has been studied from a variety of perspectives.

Generating CNN architectures and comparing their performances is difficult. Because CNN architecture generation requires a lot of computational resources, the performance of the CNN model depends on the amount of time for which it is used—even if the same method is used. In the case of NAS, 800 graphic processing units (GPUs) were used for 28 days to generate a CNN architecture with 93.99% accuracy with the CIFAR10 dataset. Therefore, if one GPU was used, approximately 20,000 days would be required. Contrastingly, in the case of the genetic CNN, 17 GPUs were used for one day to build a model with 92.90% accuracy on the CIFAR10 dataset. Accuracy is an important factor in a CNN classification model, but considering the time and computational resources used to obtain each of them, it is difficult to conclude whether NAS or genetic CNN is the better method. In addition, because each study has differences in the learning method used, data preprocessing method, and

generation purpose, etc., a given CNN architecture generation method cannot be appointed as the best method—even if it is performed on the same dataset. Each has its own advantages and disadvantages.

Using the metaheuristic algorithm in the existing CNN architecture generation methods ensures that the algorithm's population values match the layer type connection relationship, and adjusts the layer parameters. The method using the metaheuristic algorithm can check the result in less time and is easier to manufacture than the method using reinforcement learning. This is because the adjusted value is large in one model result confirmation. However, using a metaheuristic algorithm has the disadvantage that the size of the CNN architecture produced is determined according to the number of variables in the algorithm. Among these metaheuristic algorithms, the size of the CNN architecture to be generated while using HS is not fixed; in this paper, we propose using a CNN model produced in a dataset other than that used to produce the CNN architecture.

The main features of the CNN generation method using HS proposed in this study are as follows:

First, the CNN architectures generated using the existing metaheuristic algorithm mainly function by changing the connection architecture of the layers, or changing the type of layer in a CNN model with a fixed connection architecture. The existing methods have the advantage that they can be easily applied to various fields, but have the disadvantage of fixing the number of variables in the algorithm because the number of layers in the CNN architecture does not change. This study demonstrates that among the different types of metaheuristic algorithms, HS can be used to generate a complex model even in CNN architecture generation, and various variables that determine the cell structure are determined using the unit (cell) used in NASNet. The internal structure of the cell is divided into three levels, each of which is composed of three layers, and a cell unit model is created through the added part between the layers.

Second, in the existing CNN architecture generation methods, the epochs performed for CNN model performance comparison are different in each study, and the performance of the CNN model is compared using the accuracy metric. In this study, we compare the performance of the CNN model throughout 20 epochs. As a feature of the proposed method and the accuracy at 20 epochs, the accuracy before 20 epochs and the number of CNN model parameters were used to compare the CNN model performance. Using this method, the accuracy does not converge at 20 epochs, and a CNN model with as few parameters as possible is selected in a CNN architecture using multiple layers.

Third, to generate a CNN architecture, it is necessary to check the training results of various CNN models; thus the larger the CNN architecture to be created, the greater the computational resources and time required to obtain an appropriate CNN model. Therefore, as it takes a long time to learn the CNN architecture of a large image, the cell structure of NASNet is adopted. This study showed that a model trained

on a small sized CIFAR image dataset using a cell structure could be applied to a large sized sewer pipe image dataset to produce results.

The remainder of this paper is as follows. Related work is discussed in Section II. Section III describes the HS algorithm used to generate the CNN architecture and the method for setting the main parameters of the HS algorithm. In Section IV, we describe the generation of a CNN architecture using HS. In Section V, we describe the conditions used to generate the CNN architecture. In Section VI, we compare the CNN architecture generated using HS with that of other CNN models. Section VI presents the conclusions of this study.

II. PREVIOUS WORK

A. CONVOLUTIONAL NEURAL NETWORKS

CNNs are a type of neural network method that produces good results with fewer computations than a neural network with fully connected nodes. A CNN mainly uses two layers: a convolution layer, and a pooling layer. The convolution layer uses the concept of a filter to perform convolution operations on the input data. A filter is treated as a matrix of a size called the kernel size. Convolution repeats saving after adding the multiplied value of the filter with the same number of channels to the input image consisting of multiple channels. The filter moves along the given stride in the horizontal direction in the input image, moves to the end, then moves along the given stride in the vertical direction—then repeats this process. After all the work is completed, the resulting matrix of stored values becomes an output called a feature map. The number of feature maps is specified as a parameter in the corresponding layer. The convolution process can be performed by treating the edge of the input image as additionally padded with a value of 0, referred to as padding. The main parameters used in the convolution layer are the number of output channels, kernel size, stride, and padding. The pooling layer performs in the same way as the convolution layer, but the value of the filter itself does not exist because the highest value or average value in the filter is used, instead of adding values in the convolution operation. Because there is no filter with a value in the pooling layer, the number of input channels and output channels is the same. The main parameters used in the pooling layer are kernel size, stride, padding, and filter operation types. Among the parameters of these convolution and pooling layers, kernel size, stride, and padding determine the size of the feature map; thus the size of the input and output feature maps can be increased or decreased according to the values of the three types of parameters.

The convolution layer also has several names depending on how the convolution layer is used. One of them, depth-wise separable convolution, is composed of two layers: depth-wise convolution, and point-wise convolution [29]. Unlike general convolution layers, depth-wise convolution generates a feature map for each channel rather than adding all the values after multiplying the input channel by a filter. As a result, depth-wise convolution has a smaller amount

of computation compared to the number of output channels as the existing convolution layer. However, as the number of output channels increases, the amount of computation for the next operation increases significantly. Therefore, point-wise convolution is used to reduce the number of channels increased by depth-wise convolution. Point-wise convolution is a layer whose purpose is to adjust the number of output channels using a convolution layer with a 1×1 kernel size. As the kernel size is small, the amount of computation is small; therefore, it is used to reduce the computation in the next layer by reducing the number of output channels. Information loss, however, occurs as the number of channels is reduced. In this study, the number of channels is multiplied by 8 in depth-wise convolution. Among the methods used in CNNs, there is one called skip connection, which was first introduced in [30]. Backpropagation training, which is the primary learning method of neural networks, has a vanishing gradient problem in deep neural networks owing to the problem its poor performance. The vanishing gradient refers to cases where the multiplication of values less than one is accumulated during backpropagation using multiplication; as the value used in training is insufficient, the training is not performed properly. Therefore, to avoid these problems, a bypass is created that does not go through certain layers, called a skip connection. ResNet [5] is a representative CNN model using skip connections, which are an effective method.

B. RELATED WORK

There have been many studies on improving the performance of CNNs without human involvement.

In the early days, methods [8]–[16] were used to adjust parameters in the layers of CNNs. These methods fixed the type of layer and the connection architecture and changed only the layer parameters to change the performance of the CNN model. However, the disadvantage was the need to use an already created CNN model.

Subsequently, a method of generating a CNN architecture was developed; and methods such as genetic CNNs [20] using a metaheuristic algorithm determined the connection architecture between layers. The CNN architecture generation method, in which the number of variables in the algorithm is fixed, uses only a small number of variables to fix the type of layer and together with the major parameters; therefore, the order of connection of each layer of a small number of variables is determined. Further, it changes the connection order of each layer, or adds skip connections to generate a CNN architecture. Generating a CNN architecture using a layer-connected architecture has the advantage that the CNN architecture itself is not complicated; thus, it can be used easily, and it does not take a long time to evaluate the performance. However, the problem is that it is difficult to create a large CNN architecture, and the layer type is fixed. In contrast, NAS [19], which uses a recurrent neural network (RNN) [31] to determine almost all architectures of CNNs, is a representative method for determining the layer type and layer connection architecture in the generation of a

CNN architecture. The disadvantage of NAS is that it spends a significant amount of time generating a CNN architecture that has low accuracy. However, NASNet [27], abandoned several parameters and focused on the architecture of the CNN—showing that it can produce better results—and has influenced various studies on CNN architecture generation since then.

III. PSF-HARMONY SEARCH ALGORITHM

The HS algorithm [7] is a metaheuristic algorithm that uses harmony memory (HM) to store solutions with high fitness values in algorithm execution, and gradually obtains good fitness values. The characteristic of HS is that a part of the solution in HM is used for the individual solution to be checked in the next iteration.

HS has three main parameters: the harmony memory consideration rate (HMCR), pitch adjustment rate (PAR), and bandwidth. When creating a new individual in HS, one of the values of the corresponding solution part stored in the HM is used according to the probability of the HMCR for each part of the solution. If part of the new solution uses the value of HM, it is added to the solution by adding a random value within the bandwidth with the probability of PAR. If a part of the new solution does not refer to the value of HM, a random value is used. Therefore, in HS, HMCR, PAR, and bandwidth are important parameters that determine exploration and exploitation performance of the algorithm.

A parameter-setting-free (PSF) method [32] that automatically changes the HMCR and PAR bandwidths of HS according to the execution of the algorithm was devised; in this study, the advanced PSF-HS [33] is used. Advanced PSF-HS induces exploitation by increasing the values of two parameters; HMCR—which is the probability of bringing the value stored in HM—and PAR—which is the probability of adding a value within the bandwidth as the HS performance progresses. The advanced PSF-HS used in this study uses the HMCR, PAR, and bandwidths of equations (1)–(3) in the i -th iteration.

$$HMCR_i = 0.5 + 0.5 * \text{sigmoid} \times \left(10 - 10 \frac{fit_{obj} - fit_{mean}}{fit_{obj} - fit_{start}} - \frac{5}{\log(n(val))} \right) \quad (1)$$

$$PAR_i = HMCR_i * \text{sigmoid} \left(\frac{4}{n(val)} - 2 \right) \quad (2)$$

$$b_i = \begin{cases} (U - L) * \frac{fit_i - fit_{i-1}}{fit_{obj} - fit_{start}}, & b \geq (U - L) \\ (U - L) * \frac{fit_i - fit_{start}}{fit_{obj} - fit_{start}}, & \text{else} \end{cases} * 0.001 \quad (3)$$

where fit_{obj} is the target fitness, fit_{mean} is the average fitness of the solutions stored in the HM at the time of execution, fit_{start} is the average of the solutions entered in the first HM, and fit_i is the fitness of the performance of the i -th iteration—where $n(val)$ is the number of HS variables to be performed.

IV. PROPOSED METHOD

Herein, we propose a new method for determining the fitness and creating a cell composed of multiple layers to generate a CNN architecture with few parameters.

A. FITNESS

In most studies on CNN architecture generation, the accuracy is mainly used for a given classification problem to compare the performance of each CNN model. In some studies, the accuracy was compared by performing small epochs to evaluate the performance of a CNN model. [19], [21], [24]–[26] This study compared the performance of the generated CNN model at 20 epochs, and performed the final performance evaluation at 300 epochs. The fitness of the HS used in the performance comparison of the CNN model was determined by adding other factors rather than using only the accuracy of a specific epoch. Because the performance of the CNN model should be compared after performing 20 epochs, there is an inevitable difference from the learning results performed by 300 epochs. In the performance evaluation of the CNN model, to predict future performance with the result of a small number of epochs, the accuracy value at the intermediate epoch is used for fitness herein. After training the CNN model for 20 epochs, we used fitness by adding the difference between the accuracy at 10 epochs and the accuracy at 20 epochs to estimate the degree to which the accuracy converges to the accuracy obtained by 300 epochs. The learning possibility is considered after 20 epochs through the difference in accuracy at the beginning of training the CNN model. We also used the number of parameters of the CNN model for fitness. One of the goals of this study is to operate on minimal computing resources, and a model of a suitable size may be better than high accuracy. Therefore, in the HS performance, the corresponding value is added to the fitness such that the CNN model parameter has a small value. We use the fitness (4) created using these conditions.

$$fitness = acc_{20} + (acc_{20} - acc_{10}) + (5 - \log_{10} param) \quad (4)$$

where acc_i is the classification accuracy in the i -th epoch, and $param$ is the number of parameters of the CNN model.

B. CNN ARCHITECTURE

This paper is inspired by NASNet [27] and uses the cell structure for CNN architecture generation. The cell structure uses both normal and reduction cells, and the internal structure of each cell is based on Figure 1.

For the input h_i , a layer-add configuration consisting of a minimum of one and a maximum of three levels is used—and in the case of Figure 1, three levels are used. The HS determines the number of output channels of a layer, and layers of the same level have the same output channels. When using a convolution layer, batch normalization and leaky rectified linear units (ReLU) are connected. Each layer enters one of the three added parts according to the value set by the HS

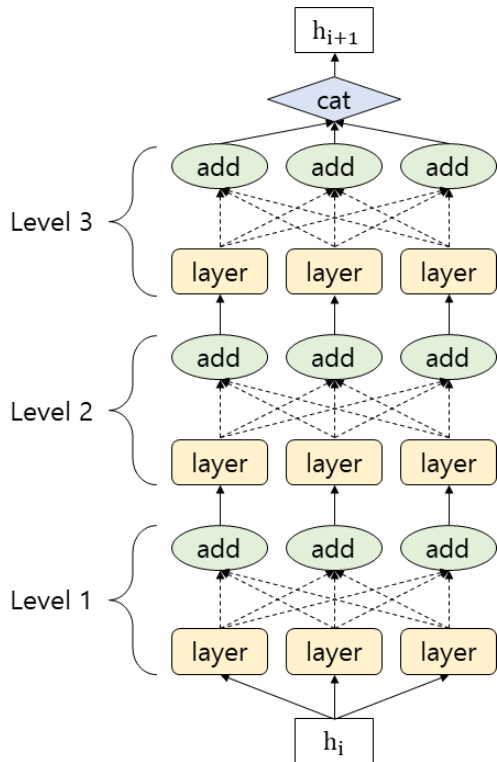


FIGURE 1. Basic cell structure.

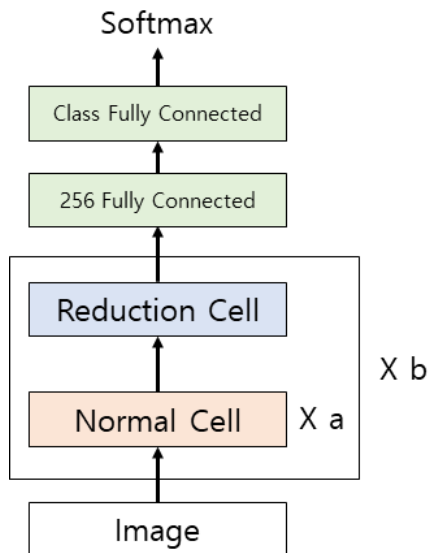


FIGURE 2. Cell connection structure.

algorithm. The added part is used as the input to the next layer and repeats. The last level adds the part that generates output h_{i+1} through a concatenation operation. The added part that is not connected to any layer in the middle places the input h_i and 1×1 convolution in the middle to make the number of channels in the feature map the same as the added part at the same level. This method makes it difficult to create a skip layer by adjusting the number of channels, but it uses an unused add part to achieve a similar result. A normal cell uses

the structure as it is, but for the reduction cell, a pooling layer that halves the feature map size is added immediately after concatenation.

As shown in Figure 2, normal and reduction cells are connected to form a CNN architecture. Normal cells are used repeatedly as many times as set in the HS. Normal cells and reduction cells form one group, and the group repeats a set number of times in the HS to connect the input image and the output softmax.

C. HARMONY SEARCH

Herein, HS determines the number of repetitions of the cell structure in the CNN architecture generation, the type and connection of each layer inside the cell, and the number of output channels per layer. The difference from the normal HS operation is that if there is an unused value—depending on the number of levels when generating the CNN architecture—this value is not stored in the HM. If the HS algorithm attempts to retrieve a value with the probability of HMCR for the next iteration, but there is no value stored in the HM, a random value is used. The pseudocode of the HS used in this study is outlined in Algorithm 1.

Algorithm 1 HS-CNN Algorithm

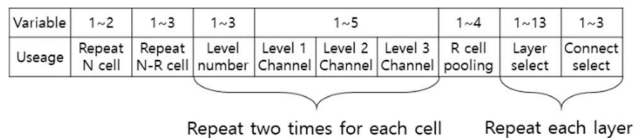
```

While (iteration < max iteration) and (time < max time)
  For (HS variable number)
    If (HMCR > rand) and (existing HM value)
      If (PAR > rand)
        Value = random HM value + (rand-0.5) * b
      Else
        Value = random HM value
      End if
    Else
      Value = random value in input range
    End if
  End for
  New Values = group of existing Value
  Fitness = calculate fitness using (4) in New Values CNN
  If (Fitness > Fitness in HM)
    Exchange low fitness Values in HM and New Values
  End if
  Change HMCR, PAR, b using (1-3)
End while

```

V. EXPERIMENTAL DESIGN

In this study, one GPU card was used to generate the CNN architecture—the GPU card was an Nvidia GeForce GTX 1060 model. The optimizer that was used for training used stochastic gradient descent (SGD). The default learning rate was 0.01, and was multiplied by 0.1, at 101 and 201 epochs, respectively, for a total of 300 epochs. Generating the CNN architecture from the CIFAR image dataset took 10 days. The training loss function used crossentropy to determine the accuracy of the training loss from the results using the sewer pipe image dataset.

**FIGURE 3.** Making CNN structure using HS.**TABLE 1.** List of layers.

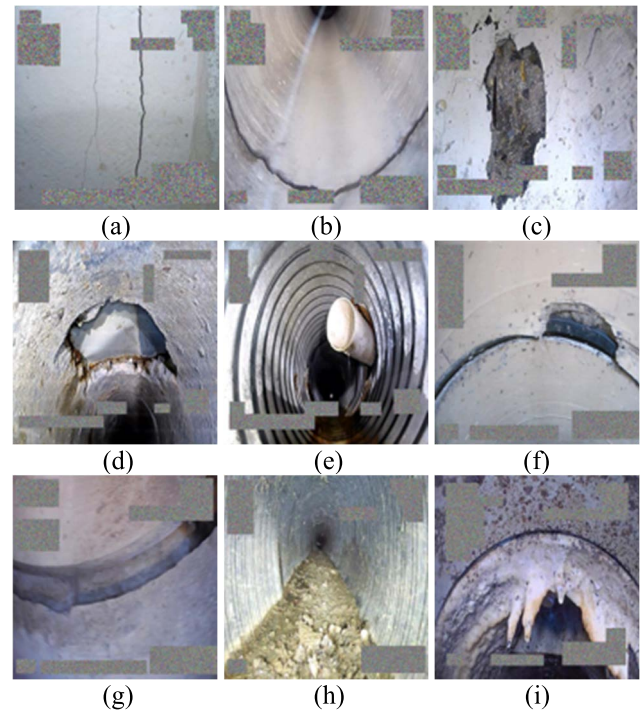
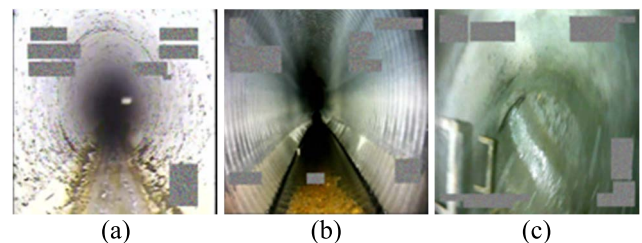
Layer number	Sort of Layer
1	3×3 Depth-wise separable convolution
2	5×5 Depth-wise separable convolution
3	1×1 Convolution
4	3×3 Convolution
5	5×5 Convolution
6	7×7 Convolution
7	$1 \times 3 + 3 \times 1$ Convolution
8	$1 \times 5 + 5 \times 1$ Convolution
9	$1 \times 7 + 7 \times 1$ Convolution
10	3×3 Maxpooling
11	5×5 Maxpooling
12	3×3 Averagepooling
13	5×5 Averagepooling

A. HARMONY SEARCH SETTING

The main parameters of HS are calculated with PSF-HS using Equations (1–3), thus there is no need to set the parameters separately. Therefore, when performing HS, a person must directly determine the input range of each variable. Each variable and input range was configured, as shown in Figure 3. Normal cell repetition was repeated 1–2 times, and normalization repetition was repeated 1–3 times. The level number is the number of levels, and determines how many levels each cell will have—it has 1–3 levels. Each level layer can have 32, 64, 128, 256, or 512 output channels, expressed as 1–5. The reduction cell reduces the size of the feature map by half by adding a pooling layer with a stride of 2 at the end of the cell structure, and expresses the type of the pooling layer as 1–4. Each layer sets the layer type with a value of 1–13, as shown in Table 1. The three added parts that each layer will be connected to are determined by a value of 1–3.

B. CIFAR IMAGE DATASET

In this study, the CIFAR image dataset [34] was used to generate the cell structure. The dataset comprised a total of 60,000 32×32 RGB images; 50,000 images were used for training, and 10,000 images were used for testing. The CIFAR image dataset is divided into CIFAR10 with 10 classes, and CIFAR100 with 100 classes. CIFAR10 has 5,000 pieces of training data and 1,000 pieces of test data for each class. CIFAR100 has 500 pieces of training data and 100 pieces of test data for each class. The 32×32 size CIFAR image dataset has been used as a benchmark dataset in various studies, and many studies have used the CIFAR image dataset

**FIGURE 4.** Damaged sewer pipe images: (a) longitudinal crack; (b) circumferential crack; (c) surface damage; (d) broken pipe; (e) lateral protruding; (f) faulty joint; (g) displaced joint; (h) silt deposits; (i) ETC.**FIGURE 5.** Undamaged sewer pipe: (a) pipe joint; (b) pipe inside; (c) invert.

in the field of CNN architecture generation. CNN architecture generation studies differ in the learning method of CNN architecture generation, epoch, GPU model, GPU days, and model generation method, etc. In CNN architecture generation studies, even if we compare the classification performance on the same dataset, no method is the best method. As it is impossible to compare the results of other studies, a specific method cannot be best for generating CNN architectures. Herein, the results of other studies using the CIFAR image dataset are tabulated, but no comparison of each study's results is given in detail.

In this study, fitness was determined only using the test dataset without using the validation dataset. In each direction of the dataset image, four pixels with zero values were padded and randomly cropped to a size of 32×32 , and a horizontal flip was performed with a probability of 0.5. As transforming the image dataset affects the performance results, only representative image transformation methods performed in the CNN architecture generation paper were applied.

C. SEWER PIPE IMAGE DATASET

The goal of this study was to create a model that classifies images of damaged sewer pipes. The sewer pipe image dataset consisted of 12 classes. Sewer pipe classes include an undamaged sewer pipe type consisting of three types: pipe joints, the inside of the pipe, and inverts. There are nine types of damage in the sewer pipes: longitudinal cracks, circumferential cracks, surface damage, broken pipes, lateral protruding, faulty joints, displaced joints, silt deposits, and ETC (Types of damaged sewer pipes not mentioned above). In the sewer pipe image dataset consisting of 2,000 images for each class, there is subtitle information included in the shooting, thus RGB noise was added to that part to prevent overfitting problems. The sewer pipe images were resized to 128×128 ; 19,200 images (80%) were used for training, whereas the remaining 4,800 images (20%) were used for testing. These images are shown in Figure 4 and Figure 5. In the CNN training using the sewer pipe image, 16 pixels with zero values were padded in each direction of the dataset image—was randomly cropped to a size of 128×128 —and then a horizontal flip was performed with a probability of 0.5. A total of 100 epochs were learned for CNN architecture learning using the transformed sewer pipe image.

VI. EXPERIMENTAL RESULT

The results of this study are comparable with those of other CNN models in other studies. The results of VGGNet [4], ResNet [5], and DenseNet [6] were obtained under the same conditions as HS-CNN using the model provided by PyTorch. Among them, VGGNet uses a model with batch normalization. As for the results of VGGNet, ResNet, and DenseNet, two cases were confirmed when transfer learning was used to train a pre-trained CNN model with the ImageNet image dataset, and when transfer learning was not used. An accurate comparison of results cited in other papers is not possible because the epochs, learning methods, image preprocessing methods, and types of GPU models used are different for each paper. This study is meaningful in comparing the CNN architecture generation study results with relatively few GPU days compared to other studies.

A. CIFAR10 IMAGE DATASET

The cell and cell connection method used in the CNN architecture obtained using the CIFAR10 image dataset is shown in Figure 6. A batch size of 4 was used in training for CNN architecture generation. A fit_{obj} of 95 was used in the advanced PSF-HS algorithm. The CNN architecture using the cell in Figure 6 repeats the normal cell twice, and the Normal-Reduction cell group also repeats twice; it is the structure of $a = 2$ and $b = 2$ in Figure 2.

The results of 300 epochs of training using a batch size of 32 in the CNN architecture produced with the structure in Figure 6 are indicated by HS-CNN in Table 2.

Compared with HS-CNN and VGGNet, which have the highest accuracy among the given CNN models, HS-CNN

TABLE 2. CIFAR10 classification results.

Model	Accuracy	Parameters	GPU days
VGGNet 19	89.75%	139.62M	-
VGGNet 19(pretrained)	92.48%	139.62M	-
ResNet 152	73.34%	58.16M	-
ResNet 152(pretrained)	89.17%	58.16M	-
DenseNet 201	81.16%	18.11M	-
DenseNet 201(pretrained)	88.71%	18.11M	-
HS-CNN	92.63%	20M	10

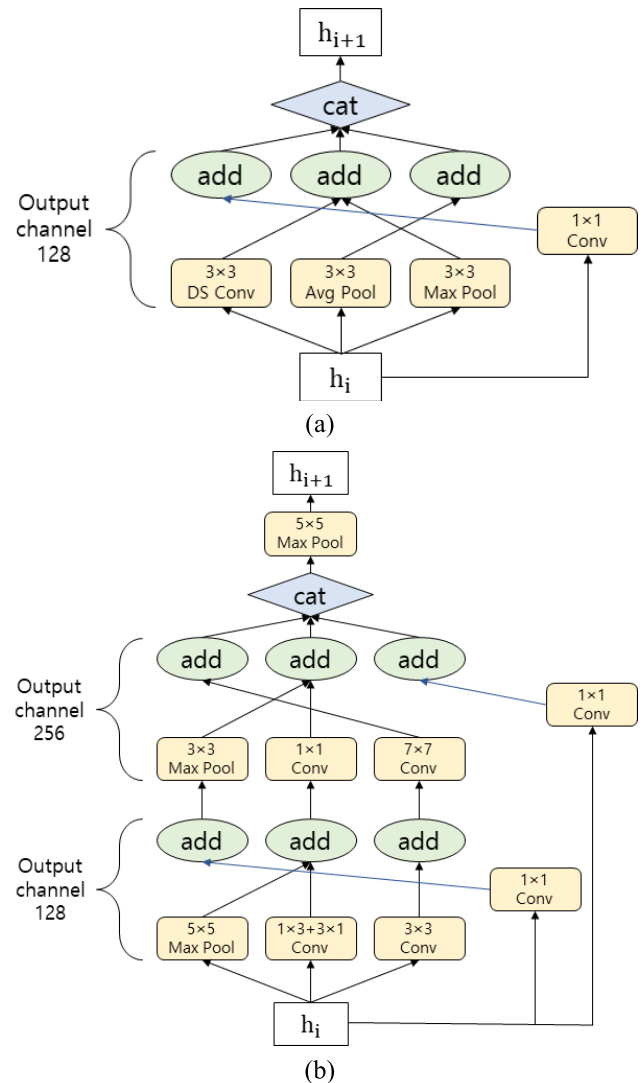


FIGURE 6. Cell structure of HS-CNN created using the CIFAR10 image dataset: (a) normal cell; (b) reduction cell.

uses fewer parameters, and the accuracy is 0.15% higher. As a result, HS-CNN took only 10 GPU days and showed better results than the pre-trained CNN model.

B. CIFAR100 IMAGE DATASET

The cell and cell connection method used in the CNN architecture obtained using the CIFAR100 image dataset is shown

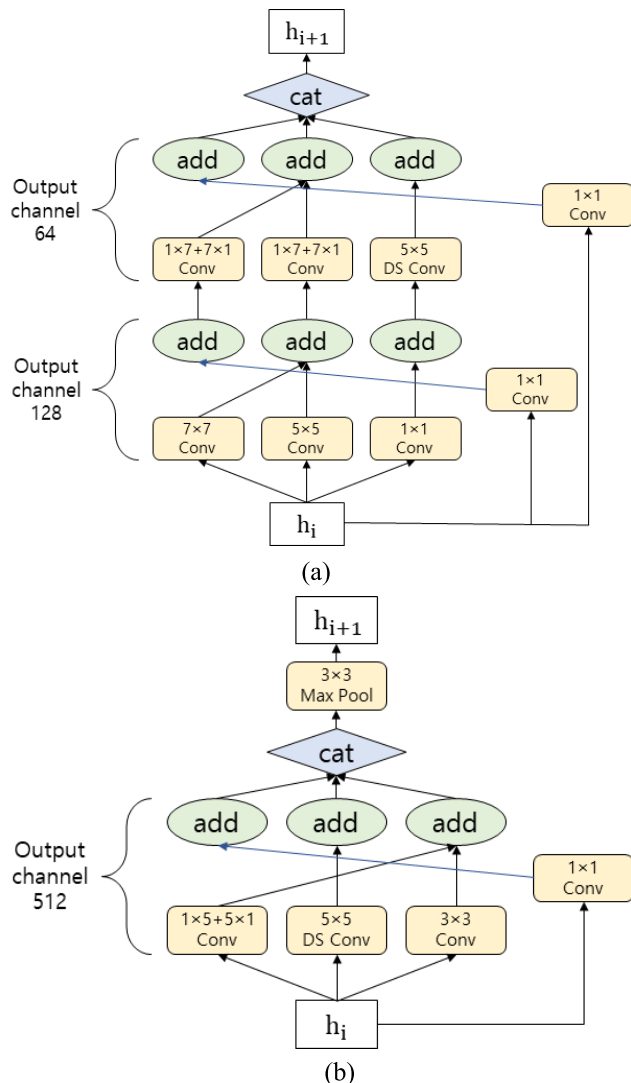


FIGURE 7. Cell structure of HS-CNN created using the CIFAR100 image dataset: (a) normal cell; (b) reduction cell.

in Figure 7. A batch size of 4 was used in training for CNN architecture generation. A fit_{obj} of 75 was used in the advanced PSF-HS algorithm. The CNN architecture using the cell in Figure 7 repeats the normal cell twice and the Normal-Reduction cell group twice, shown by $a = 2$ and $b = 2$ in Figure 2.

The results of 300 epochs of training using a batch size of 32 in the CNN architecture constructed with the structure in Figure 7 are indicated by HS-CNN in Table 3.

Compared with HS-CNN and VGGNet, which has the highest accuracy among the given CNN models, it can be seen that HS-CNN achieves a 2% higher accuracy.

C. SEWER PIPE IMAGE DATASET

Training using the sewer pipe image dataset was performed for 100 epochs, with learning rates of 0.01, of 0.001, and 0.0001 at the start, at 50 epochs, and 75 epochs, respectively. A batch size of 4 was used.

TABLE 3. CIFAR100 classification results.

Model	Accuracy	Parameters	GPU days
VGGNet 19	63.85%	139.99M	-
VGGNet 19(pretrained)	70.41%	139.99M	-
ResNet 152	37.63%	58.35M	-
ResNet 152(pretrained)	63.83%	58.35M	-
DenseNet 201	50.74%	18.29M	-
DenseNet 201(pretrained)	64.44%	18.29M	-
HS-CNN	72.41%	57.45M	10

TABLE 4. Sewer pipe classification results.

Model	Accuracy	Parameters	Train loss	Test loss
VGGNet 19	71.31%	139.63M	0.06081	0.63953
VGGNet 19 (pretrained)	67.60%	139.63M	0.00025	1.86006
ResNet 152	30.60%	58.17M	0.00071	9.51669
ResNet 152 (pretrained)	25.23%	58.17M	0.00001	6.85108
DenseNet 201	16.02%	18.12M	0.00079	10.1385
DenseNet 201 (pretrained)	19.81%	18.12M	0.00009	7.59148
GoogLeNet	23.73%	11.99M	0.00055	9.35565
Wide ResNet101	24.58%	124.86M	0.00208	10.8064
NasNet-A Mobile	24.06%	4.25M	0.00022	7.68038
NasNet-A Mobile (pretrained)	58.15%	4.25M	0.00004	2.26854
NasNet-A Large	26.92%	84.77M	0.00017	8.85162
NasNet-A Large (pretrained)	61.62%	84.77M	0.00004	2.28855
HS-CNN (CIFAR10)	73.27%	62.81M	0.00000	0.00199
HS-CNN (CIFAR100)	76.98%	104.89M	0.00000	0.00004

In Table 4, the results of GoogLeNet [3] and WideResNet [36] that do not use transfer learning are added to the

results of VGGNet, ResNet, and DenseNet. In addition, the results of NASNet-A Mobile and NASNet-A Large using the CNN model created in the NASNet [27] study were added. In Table 4, the results using VGGNet and ResNet show that when using the CFIAR image dataset, the accuracy may decrease when using transfer learning instead of a significant increase in accuracy.

The image of the sewer pipe has a low RGB value, as seen in Figure 4-5, and there are many parts close to 0 in the center of the image; therefore it shows a different appearance from the ImageNet image with a relatively high RGB value. As a result, Table 4 shows that the accuracy of VGGNet and ResNet decreased significantly when transfer learning was used.

The decreased accuracy due to transfer learning for a specific CNN model can be seen in the training loss. The corresponding CNN model tends to have a relatively lower training loss when transfer learning is used, which presents the problem of overfitting if an unsuitable pre-trained CNN model is used. Because the sewage pipe image dataset shows a different shape from the image dealt with in the existing classification problem, it can be seen that it is necessary to create a CNN model with high classification accuracy as transfer learning cannot be used with a general pre-trained CNN model.

The sewer pipe classification CNN model based on HS-CNN produced with the CIFAR image dataset confirmed two types. One created a CNN model by adding a Normal-Reduction structure to a model created using CIFAR10 once; and the other was a model created using CIFAR100 by adding a Normal-Reduction structure twice to create a CNN model. The classification accuracy of the sewer pipe image of HS-CNN produced based on CIFAR10 was 73.27%, which was 1.96% higher than that (71.31%) obtained using VGGNet—and is a significant difference compared to the difference in the accuracy of 0.15% shown in the result of CIFAR10. The classification accuracy of the sewer pipe image produced by HS-CNN based on CIFAR100 was 76.98%, which was 5.67% higher than that (71.31%) obtained using VGGNet—and is a significant difference compared to the difference in the accuracy of 2% shown in the result of CIFAR100.

NASNet-A is a CNN model showing the highest CIFAR10 classification accuracy among the CNN models of generation CNN architecture research papers [19]–[28] and is well known compared to the CNN model of other generation CNN architecture research papers. Also, NASNet-A Mobile showed an accuracy of 74.0%, and NASNet-A Large showed 82.7% in ImageNet classification, showing the superiority of the transferable architecture method in NASNet papers. When performing the NASNet-A CNN model on the sewer pipe classification problem, the size of the input image was converted to 224×224 for NASNet-A Mobile and 331×331 for NASNet-A Large. When performing the sewer pipe classification of NASNet-A, like other CNN models, the number of output channels of the linear layer of the final

output was changed to 12, the number of classes of the sewer pipe, and the rest of the learning conditions were also used in the same way. In Table 4, the sewage pipe classification accuracy of the NASNet-A model without transfer learning was very low. NASNet-A, which showed very low accuracy compared to training loss, seems to show low accuracy due to overfitting like ResNet and DenseNet. Sewer classification accuracy of NASNet-A model using transfer learning of pre-trained model with ImageNet dataset showed higher accuracy than when transfer learning was not used. It can be seen that the pre-train model of NASNet-A, which shows a significant difference in accuracy compared to other CNN models according to the use of transfer learning, has more suitable parameters for classifying sewage pipe images. However, since the sewage pipe classification accuracy using all NASNet-A models is lower than the sewage pipe classification accuracy using the VGGNet model, it can be confirmed that the NASNet-A model is not suitable for the sewage pipe image classification.

Therefore, it can be seen that the CNN model based on HS-CNN proposed in this paper performs better in classification of sewage pipe images.

VII. CONCLUSION

This paper proposed a method to generate a CNN architecture from a dataset using small-sized images with HS, and to create a classification model for large-sized images using the generated CNN architecture.

We generated the CNN architecture through HS using the CIFAR image dataset as a dataset of small-size images and compared the transfer learning results from other papers and known CNN models. It showed better accuracy than known CNN models. Unlike other CNN architecture generation methods, the CNN architecture using HS should be considered as a model built using computing resources with 10 GPU days.

Based on the HS-CNN produced by the CIFAR image dataset, we created a classification CNN model of the sewer pipe image dataset using large-size images and confirmed the results. Because the sewer pipe image dataset is an original image dataset that has not been used in other studies, it cannot be compared with the results of other studies. Therefore, the sewer pipe image classification was compared with existing CNN models. In the sewer pipe image dataset, the existing CNN models were able to evaluate the performance of the CNN model alone, as the accuracy decreased when transfer learning was performed with the pre-trained CNN model through the ImageNet image. The CNN architecture created using HS showed a classification accuracy that was at least 1.96%, and up to 5.67% higher than that of VGGNet, which has the highest accuracy among the existing CNN models.

From this result, therefore, we expect that our method would be useful as a promising deep-learning tool for image classification and image recognition to a variety of vision communities such as augmented reality (AR) or medical

imaging fields for early diagnostics, which will be near future applications.

REFERENCES

- [1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, Feb. 2015, doi: [10.1038/nature14539](https://doi.org/10.1038/nature14539).
- [2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: [10.1145/3065386](https://doi.org/10.1145/3065386).
- [3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jan. 2015, pp. 1–9, doi: [10.1109/CVPR.2015.7298594](https://doi.org/10.1109/CVPR.2015.7298594).
- [4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, San Diego, CA, USA, May 2015, pp. 1–14.
- [5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778, doi: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).
- [6] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2261–2269, doi: [10.1109/CVPR.2017.243](https://doi.org/10.1109/CVPR.2017.243).
- [7] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *J. Simul.*, vol. 76, no. 2, pp. 60–68, Feb. 2001, doi: [10.1177/003754970107600201](https://doi.org/10.1177/003754970107600201).
- [8] A. Rikhtegar, M. Pooyan, and M. T. Manzuri-Shalmani, "Genetic algorithm-optimised structure of convolutional neural network for face recognition applications," *IET Comput. Vis.*, vol. 10, no. 6, pp. 559–566, Sep. 2016, doi: [10.1049/iet-cvi.2015.0037](https://doi.org/10.1049/iet-cvi.2015.0037).
- [9] S. Gibb, H. M. La, and S. Louis, "A genetic algorithm for convolutional network structure optimization for concrete crack detection," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–8, doi: [10.1109/CEC.2018.8477790](https://doi.org/10.1109/CEC.2018.8477790).
- [10] H. Chung and K.-S. Shin, "Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction," *Neural Comput. Appl.*, vol. 32, no. 12, pp. 7897–7914, Jun. 2020, doi: [10.1007/s00521-019-04236-3](https://doi.org/10.1007/s00521-019-04236-3).
- [11] T. Yamasaki, T. Honma, and K. Aizawa, "Efficient optimization of convolutional neural networks using particle swarm optimization," in *Proc. IEEE 3rd Int. Conf. Multimedia Big Data (BigMM)*, Apr. 2017, pp. 70–73, doi: [10.1109/BigMM.2017.69](https://doi.org/10.1109/BigMM.2017.69).
- [12] Y. Wang, H. Zhang, and G. Zhang, "CPSO-CNN: An efficient PSO-based algorithm for fine-tuning hyper-parameters of convolutional neural networks," *Swarm Evol. Comput.*, vol. 49, pp. 114–123, Sep. 2019, doi: [10.1016/j.swevo.2019.06.002](https://doi.org/10.1016/j.swevo.2019.06.002).
- [13] T. Serizawa and H. Fujita, "Optimization of convolutional neural network using the linearly decreasing weight particle swarm optimization," 2020, *arXiv:2001.05670*.
- [14] W.-C. Yeh, Y.-P. Lin, Y.-C. Liang, and C.-M. Lai, "Convolution neural network hyperparameter optimization using simplified swarm optimization," 2021, *arXiv:2103.03995*.
- [15] W.-Y. Lee, S.-M. Park, and K.-B. Sim, "Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm," *Optik*, vol. 172, pp. 359–367, Nov. 2018, doi: [10.1016/j.ijleo.2018.07.044](https://doi.org/10.1016/j.ijleo.2018.07.044).
- [16] S. H. Kim, Z. W. Geem, and G. T. Han, "Hyperparameter optimization method based on harmony search algorithm to improve performance of 1D CNN human respiration pattern recognition system," *Sensors*, vol. 20, no. 13, 3697, Jul. 2020, doi: [10.3390/s20133697](https://doi.org/10.3390/s20133697).
- [17] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996, doi: [10.1613/jair.301](https://doi.org/10.1613/jair.301).
- [18] D. Whitley, "A genetic algorithm tutorial," *Statist. Comput.*, vol. 4, no. 2, pp. 65–85, Jun. 1994, doi: [10.1007/BF00175354](https://doi.org/10.1007/BF00175354).
- [19] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*.
- [20] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Aug. 2017, pp. 1379–1388, doi: [10.1109/ICCV.2017.154](https://doi.org/10.1109/ICCV.2017.154).
- [21] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," 2016, *arXiv:1611.02167*.
- [22] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," 2017, *arXiv:1711.00436*.
- [23] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin, "Large-scale evolution of image classifiers," 2017, *arXiv:1703.01041*.
- [24] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proc. Genetic Evol. Comput. Conf.*, 2017, pp. 497–504, doi: [10.1145/3071178.3071229](https://doi.org/10.1145/3071178.3071229).
- [25] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 2787–2794. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/11709>
- [26] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 394–407, May 2019, doi: [10.1109/TEVC.2019.2916183](https://doi.org/10.1109/TEVC.2019.2916183).
- [27] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Aug. 2018, pp. 8697–8710, doi: [10.1109/CVPR.2018.00907](https://doi.org/10.1109/CVPR.2018.00907).
- [28] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, Sep. 2020, doi: [10.1109/TCYB.2020.2983860](https://doi.org/10.1109/TCYB.2020.2983860).
- [29] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Aug. 2017, pp. 1251–1258, doi: [10.1109/CVPR.2017.195](https://doi.org/10.1109/CVPR.2017.195).
- [30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735).
- [31] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," 2013, *arXiv:1312.6026*.
- [32] Z. W. Geem and K.-B. Sim, "Parameter-setting-free harmony search algorithm," *Appl. Math. Comput.*, vol. 217, no. 8, pp. 3881–3889, Dec. 2010, doi: [10.1016/j.amc.2010.09.049](https://doi.org/10.1016/j.amc.2010.09.049).
- [33] Y. W. Jeong, S. M. Park, Z. W. Geem, and K. B. Sim, "Advanced parameter-setting-free harmony search algorithm," *Appl. Sci.*, vol. 10, no. 7, 2586, 2020, doi: [10.3390/app10072586](https://doi.org/10.3390/app10072586).
- [34] A. Krizhevsky and G. Hinton. (2009). *Learning Multiple Layers of Features From Tiny Images*. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.9220&rep=rep1&type=pdf>
- [35] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks Trade*, vol. 7700. Berlin, Germany: Springer, 2012, pp. 421–436, doi: [10.1007/978-3-642-35289-8_25](https://doi.org/10.1007/978-3-642-35289-8_25).
- [36] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.



YOUNG-WOON JEONG received the B.S. and M.S. degrees from the Department of Electronic Engineering, Chung-Ang University, Seoul, South Korea, in 2020 and 2022, respectively. His research interests include artificial intelligent, machine learning, neural networks, heuristic algorithm, and harmony search algorithm.



KWEE-BO SIM (Member, IEEE) received the B.S. and M.S. degrees from the Department of Electronics Engineering, Chung-Ang University, South Korea, in 1984 and 1986, respectively, and the Ph.D. degree from the Department of Electronics Engineering, The University of Tokyo, Japan, in 1990. Since 1991, he has been a Faculty Member with the School of Electrical and Electronics Engineering, Chung-Ang University, where he is currently a Professor. His research interests include brain-computer interface (BCI), emotion and intention recognition, intelligent robots, intelligent systems, computational intelligence, intelligent home, ubiquitous computing and sense networks, machine learning and deep learning, convolutional neural networks, fuzzy systems, and evolutionary computation. He is a member of the SICE, RSJ, IEEK, KIEE, KIIS, KROS, and IEMEK. He is an ICROS Fellow.



SEUNGMIN PARK (Member, IEEE) received the B.S. and Ph.D. degrees in control and system engineering from Chung-Ang University, South Korea, in 2010 and 2019, respectively. He joined the Faculty of the Department of Software, Dongseo University, Busan, South Korea, in 2019, where he is currently an Assistant Professor with the Department of Software. He is the General Affairs Director, Korean Institute of Intelligent Systems Society (KIIS) and the Director for Education,

Korea Society of Computer Information (KSCI). His current research interests include machine learning, brain computer interface, pattern recognition, intention recognition, and deep learning.



JEILL OH received the B.S. degree from Yonsei University, South Korea, in 1992, and the M.S. and Ph.D. degrees in civil and environmental engineering from the University of Colorado at Boulder, USA, in 1995 and 1998, respectively. He is currently a Professor with the Department of Civil and Environmental Engineering, Chung-Ang University. He worked as a Postdoctoral Research Fellow with the School of Environmental Engineering, Pohang University of Science and Technology (POSTECH), prior to joining Chung-Ang University, in 2000. He has

been serving as a member for the Executive Committee of Korean Society of Water & Wastewater (KSWW), and Korean Society of Water Environment (KSWE) for 20 years. His research interests include diverse aspects of information and communication technology (ICT) coupled with environmental system, such as real-time monitoring (RTM), real-time control (RTC), optimization, and deep learning technology in urban water systems.



WOO JUNE CHOI (Member, IEEE) received the bachelor's degree in electronics from Chung-Ang University, Seoul, South Korea, in 2005, and the M.S. and Ph.D. degrees in biomedical optics from the Gwangju Institute of Science and Technology (GIST), Gwangju, South Korea, in 2007 and 2012, respectively. Since 2012, he has been a Postdoctoral Fellow with the Korea Basic Science Institute (KBSI), Daejeon, South Korea, and followed by the University of Washington, Seattle,

USA, in 2013. He is currently an Assistant Professor with the School of Electrical and Electronics Engineering, College of ICT Engineering, Chung-Ang University. His research interests include develop novel optical imaging techniques and methodologies in photonics, in specific, optical coherence tomography angiography (OCTA), surface-enhanced Raman microscopy (SERS), and fluorescence microscopy.

...