electronics

MDPI

Article

Knowledge Distillation for Image Signal Processing Using Only the Generator Portion of a GAN

Youngjun Heo and Sunggu Lee *

Department of Electrical Engineering, Pohang University of Science and Technology, Pohang 37673, Republic of Korea
* Correspondence: slee@postech.ac.kr

Abstract: Knowledge distillation, in which the parameter values learned in a large teacher network are transferred to a smaller student network, is a popular and effective network compression method. Recently, researchers have proposed methods to improve the performance of a student network by using a Generative Adverserial Network (GAN). However, because a GAN is an architecture that is ideally used to create *realistic* synthetic images, a pure GAN architecture may *not* be ideally suited for knowledge distillation. In knowledge distillation for image signal processing, synthetic images do *not* need to be realistic, but instead should include features that help the training of the student network. In the proposed Generative Image Processing (GIP) method, this is accomplished by using *only* the generator portion of a GAN and utilizing special techniques to capture the distinguishing feature capability of the teacher network. Experimental results show that the GIP method outperforms knowledge distillation using GANs as well as training using only knowledge distillation.

Keywords: image signal processing; knowledge distillation; model compression; generative adversarial network

Citation: Heo, Y.; Lee, S. Knowledge Distillation for Image Signal Processing Using Only the Generator Portion of a GAN. *Electronics* **2022**, *11*, 3815. https://doi.org/10.3390/electronics11223815

Academic Editors: Leonardo Pantoli, Egidio Ragonese, Paris Kitsos, Gaetano Palumbo and Costas Psychalinos

Received: 29 September 2022
Accepted: 17 November 2022
Published: 20 November 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

In recent years, the performance of image classification based on image signal processing which extracts image feature with neural networks have been significantly improved with the aid of highly complex *deep* neural network models. At the same time, significant advances have been made in network compression [1], in which the complexity of the neural network models are reduced while attempting to minimize the accompanying loss in accuracy of the models used.

Knowledge distillation [2,3] has recently attracted attention as an effective model compression method. Knowledge distillation is a method that transfers knowledge from a pre-trained large neural network model (teacher network model) to a relatively small and low-performance neural network model (student network model). By doing so, the student network model can achieve higher performance than when it is trained from scratch using a given training dataset. There are also other model compression methods such as neural network quantization [4,5], pruning [6,7] and binarization [8–13]. These latter methods sometimes used in conjunction with knowledge distillation.

The training method used in knowledge distillation varies according to the knowledge types and knowledge distillation algorithms used. According to the survey paper written by Gou et al. [14], knowledge types can be divided into response-based, feature-based, and relation-based knowledge. Also, depending on the primary techniques used, distillation algorithms are classified into adversarial knowledge distillation, data-free knowledge distillation and other types of knowledge distillation [15–19].

Adversarial knowledge distillation is a method that uses a Generative Adversarial Network (GAN) [20] for neural network compression using knowledge distillation. GANs can be used in conjunction with knowledge distillation to improve the performance of

knowledge distillation by allowing the student network to better mimic the teacher network. Since the student network has a relatively smaller capability to learn from a given dataset, adversarial knowledge distillation has been used to provide better knowledge about the distribution of the dataset to the student network.

In [15], images are augmented to the existing training dataset, and this improved the performance of knowledge distillation due to the augmented dataset. However, to train the student network, this training method requires three additional neural networks—i.e., the teacher network, generator network and discriminator network. Furthermore, before training the targeted student network, additional training is required for the GAN architecture.

In this paper, we propose a novel knowledge distillation training strategy based on generative image processing method, which leverages the inherent knowledge of the teacher network captured by the generator network. Unlike previous adversarial knowledge distillation using both the generator and discriminator parts of GANs to create synthesized training images, the proposed method uses *only* a generator network to create the synthesized images. Furthermore, since a generator network is trained simultaneously with the student network, no additional training processes are required for the generator network. The intrinsic knowledge of the teacher network is used not only for knowledge distillation itself but also for training the generator to synthesize the images.

Several techniques are utilized to improve the effectiveness of the generator network used. Unlike in typical GANs architecture, the generator is not trained separately, but together with the student network. A loss function is used that is based on the information inherent in the intermediate features of the teacher network and the predictions of the teacher network on a given dataset. Using this loss function, the synthetic images generated are not necessarily realistic looking images—the generator is specifically *designed* to improve student network training rather than to produce realistic images. However, the loss function is also designed, using the pre-trained batch normalization statistics [21,22] of the teacher network, so that the synthetic images produced have an image data distribution that is *close* to a real dataset image distribution.

We conducted experiments to verify the effectiveness of the proposed method not only on the full-precision neural network model but also on an extremely quantized neural network model, which only uses 1-bit values for weight and activation parameters. For further verification, we compared the performance with other GAN-based knowledge distillation methods [15]. CIFAR-10 and CIFAR-100 datasets [23] were used for the experiments.

2. Background and Related Work

In this section, background knowledge and related work on knowledge distillation, adversarial knowledge distillation and binarized neural networks will be explained.

2.1. Knowledge Distillation

Knowledge distillation has attracted much attention as an effective model compression method. In response-based knowledge distillation [2], shown in Figure 1, the student network model is trained with the aim of narrowing the distance between its output vectors and output vectors of the teacher network. On the other hand, feature-based knowledge distillation [24–26] also uses the knowledge of the intermediate layers of the teacher network model. In FitNets [24], intermediate features were used to train the student network, and the performance of the student network model was improved by allowing the student network model to mimic the feature outputs of the teacher network. Furthermore, attention-transfer [27] methods, which distill knowledge by transferring an attention-map, have also been proposed. Finally, relation-based knowledge distillation utilizes the correlation between specific layers rather than using the outputs of the layers in the teacher network model [28].

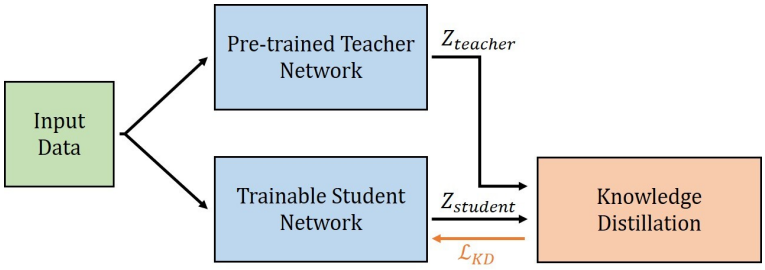

Figure 1. The response-based knowledge distillation training process: the output vector $z_{student}$ of the student network model and the output vector $z_{teacher}$ of the teacher network model are used to calculate a loss function $\mathcal{L}_{KD}$.

2.2. Adversarial Knowledge Distillation

Adversarial knowledge distillation attempts to improve knowledge distillation by using GANs [20]. A number of papers [15–19] have suggested methods to create synthetic images in various ways. In [15], fake images synthesized from a generator labeled with the teacher's predictions were augmented to the training dataset with a certain ratio p_{fake}. The generator and discriminator network were trained using a real image dataset. In the case of [16], synthetic images adjacent to the decision boundaries which is hard to classify the classes are generated so that the student network model can perform well without an actual dataset. In the case of [17], images were created by introducing various generator loss terms based on the intrinsic knowledge of the generator. Xu et al. [22] suggested a method whereby the network model could be compressed further by applying a data-free knowledge distillation method with batch normalization statistics to a quantized student neural network.

2.3. Binarized Neural Network

A Binarized Neural Network (BNN) [8–13] is a convolutional neural network with only 1-bit values for weight and activation parameters. Deep neural networks have achieved incredibly high performance, but are accompanied by huge amounts of computation and memory usage. Accordingly, BNNs are being actively studied as one of the promising methods to compress neural networks. Recently, a BNN model has been combined with knowledge distillation during the training process for higher performance [12,13]. By reducing the Kuleback-Leibler divergence between the output of the real-valued teacher network and the output of the binarized student network, the binarized student network can better understand the distributions of the real-valued teacher network.

3. Proposed Method: Generative Image Processing

In the proposed *Generative Image Processing* (*GIP*) based knowledge distillation training strategy, synthetic images based on a pre-trained teacher network model's intrinsic knowledge are augmented to a training dataset in order to improve the performance of the student network. The structure of the proposed GIP method strategy is illustrated in Figure 2.

Synthetic images are processed and generated from a generator trained with loss functions based on the intermediate layers, output layer, and batch normalization statistics of the teacher network model. In other words, since the generator generates images from the teacher network model rather than from the image dataset, there is no need for a separate training process, and it is trained simultaneously with the student network model. Furthermore, the generated synthetic images are generated anew during each epoch and used together with the previous real image training dataset.

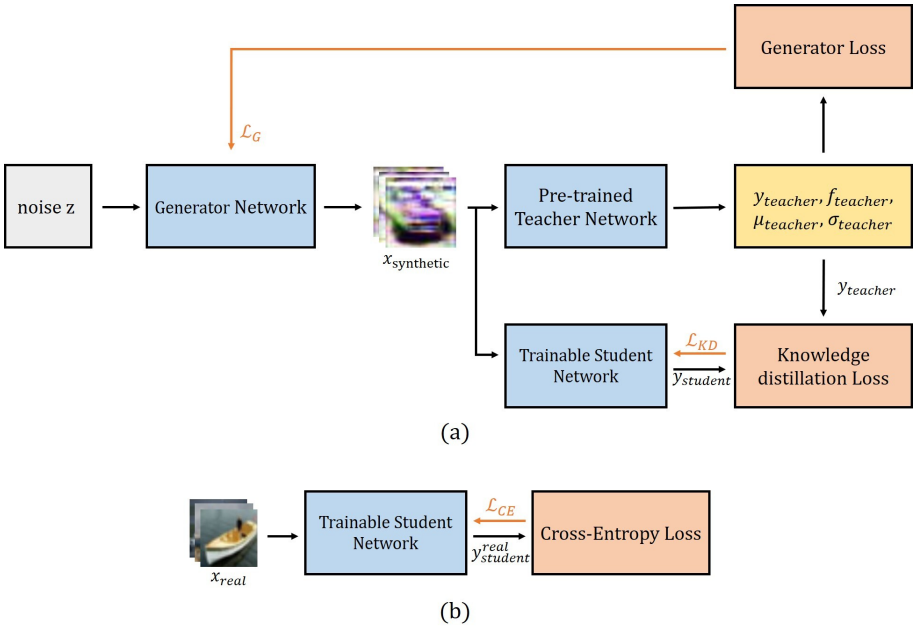

Figure 2. This figure shows the two stages of the proposed GIP method. In Stage 1, shown in part (**a**), $x_{synthetic}$ is generated from a generator network G and used as an input to both the pre-trained teacher network and the student network. $y_{teacher}$, $f_{teacher}$, $\mu_{teacher}$, $\sigma_{teacher}$ is extracted from the teacher network to calculate the generator loss $\mathcal{L}_G$. The generator network is updated with knowledge extracted from the teacher network, and the student network is updated using knowledge distillation loss $\mathcal{L}_{KD}$ calculated with $y_{teacher}$ and $y_{student}$. In Stage 2, shown in part (**b**), real dataset images x_{real} [23] are used as the input to the student network only. The cross-entropy loss $\mathcal{L}_{CE}$ is calculated and the student network is updated.

3.1. Definitions and Equations

In this section, definitions and equations related to knowledge distillation, including Kullback-Liebler diversity loss and knowledge distillation loss, will be explained.

3.1.1. Logit Vector

The output of the teacher network model can be expressed as a *logit vector* y. Logit is a function described using Equation (1), which uses the natural logarithm value of the ratio between the probabilities p and $1 - p$. Therefore, knowledge distillation loss can be represented by a logit vector $y_{student}$ of the student network model and a logit vector $y_{teacher}$ of the teacher network model as shown in Figure 1.

$$y_i = logit(p_i) = \ln \frac{p_i}{1 - p_i}. \tag{1}$$

3.1.2. Softmax Function with Temperature

Hinton et al. [2] introduced a softmax temperature T to "soften" the logit vector of the teacher network model. By doing this, the student network model can be trained with information that indicates the relationship among the entire set of classes, and not just a specific class. This concept can be encapsulatd in a *softmax function*, which can be expressed as follows.

$$p(y_j, T) = \frac{e^{y_j/T}}{\sum_j^N e^{y_j/T}}. \tag{2}$$

3.1.3. KD Loss

In Equation (2), j represents the j-th class of the dataset and T is the softmax temperature factor. As Factor T increases, the distribution of probabilities becomes smoother. Based

on the above softmax function, the loss between the student network model and the teacher network model can be expressed as follows.

$$L_{KD} = \mathcal{L}_{KD}(p(y_{teacher}, T), p(y_{student}, T)) \tag{3}$$

In the knowledge distillation training process, Kuleback-Leibler divergence loss is used as the loss function. *Kullback-Leibler diversity loss* (*KL loss*) is the difference between cross-entropy function and entropy function. The softmax value of the teacher network model and the softmax value of the student network model are used as inputs to KL loss. The cross-entropy loss function and KL loss function used for the experiments can be expressed as follows.

$$L_{CE} = \mathcal{H}(p, q) = -\sum_i p_i log q_i \tag{4}$$

$$L_{KL} = \mathcal{H}(p, q) - \mathcal{H}(p) = \sum_i p_i log \frac{p_i}{q_i} \tag{5}$$

By using the above formula as a loss function for knowledge distillation training, the loss function for knowledge distillation with the softmax function can be expressed as follows.

$$L_{KL} = \mathcal{L}_{KD}(p(y_{teacher}, T), p(y_{student}, T)) \tag{6}$$

$$\mathcal{L}_{KD}(p(y_{teacher}, T), p(y_{student}, T)) = \sum_i p(y_{teacher}, T) log \frac{p(y_{teacher}, T)}{p(y_{student}, T)} \tag{7}$$

Allowing the student network model to imitate the logits of the teacher network model can achieve higher performance compared to training without knowledge distillation.

3.2. Generative Adversarial Networks

In the proposed method, GANs architecture is used to generate synthetic images. GANs architecture consists of generator (G) and discriminator (D). G generates synthetic images from real image dataset and D discriminates synthetic images from real image dataset. Therefore, G learns to generate synthetic images that is difficult to distinguish, and D is trained to well distinguish between two image groups. Through this process, G creates synthetic images close to real image dataset. Formula of objective function, which indicates GANs structure, can be expressed as follows.

$$\mathcal{L}_{GAN} = \mathbb{E}_{r \sim p_{data}(r))}[logD(r)] + \mathbb{E}_{z \sim p_z(z))}[log(1 - D(G(z)))] \tag{8}$$

where noise vector z indicates input of G, and r means real dataset. Optimized G after training process can be expressed with optimized D as follows.

$$G_{optimized} = \arg\min_G \mathbb{E}_{z \sim p_z(z))}[log(1 - D_{optimized}(G(z)))] \tag{9}$$

As such, a general GANs structure requires a real image dataset, but [17] method suggests a way to utilize the GANs architecture in case where the real image dataset is not available. More specifically, the optimized D was fixed with a pre-trained teacher network model so that it is not necessary to train D separately.

In our proposed method, we maximized the utilization of the teacher network model by applying the generator structure of the corresponding GANs architecture to a general knowledge distillation training process rather than a restricted scenario where the real image dataset is not available. Since the generator object function of a typical GANs architecture is not suitable for utilizing the knowledge of a teacher network model, new loss functions should be introduced, and the corresponding loss functions will be described in more detail in the later part of this section.

3.3. Batch Normalization Statistics

According to [21], the distribution information for the training dataset can be preserved in the batch normalization layer of the neural network model trained with the corresponding dataset. Therefore, batch normalization statistics (BNS), which means the values of mean and variance of the batch normalization layers, can be used to manage the distribution of the generated images. Accordingly, it is possible to synthesize images that improve the performance of knowledge distillation by utilizing the knowledge inherent in the BNS of the teacher network model. Detailed loss functions to leverage corresponding BNS of the teacher network model will be described in the next part of this section.

3.4. Proposed Generative Image Processing (GIP) Algorithm

In the GIP Algorithm, shown in Algorithm 1, the method we proposed consists of two parts. The first part shown in Figure 2a is a training process using synthesized images created from the generator. SI refers to the number of iterations for training with synthetic images. Noise vectors of a pre-determined batch size are randomly generated from a normal distribution and used as an input of a generator. After that, $x_{synthetic}$, which is the synthetic images of batch size same as input noise vectors, are generated from the generator. Input the generated $x_{synthetic}$ into the teacher network $\mathcal{N}_{teacher}$ and extract $y_{teacher}$, $f_{teacher}$, $\mu_{teacher}$, $\sigma_{teacher}$. The output $y_{teacher}$ is the prediction of the last layer of the teacher model, $f_{teacher}$ is the output just before the final fully-connected layer, and $\mu_{teacher}$ and $\sigma_{teacher}$ are BNS of the teacher network. Calculate the loss function of the generator composed of the corresponding values and update the generator. After that, also input the same $x_{synthetic}$ into the student network, and calculate the knowledge distillation loss value to update the student network.

The second part shown in Figure 2b is a training process using real dataset images. RI refers to the number of iterations for training with real dataset images. In this part, input the real image dataset into the student network and update the student network by calculating the cross entropy loss between the output prediction of the student network and the ground truth label of real image dataset.

Algorithm 1 Knowledge distillation training improvement using generative image processing

Input: Teacher network $\mathcal{N}_{teacher}$, Student network $\mathcal{N}_{student}$, Hyper-parameters, Real dataset images

Output: Optimized student network $\mathcal{N}_{student}^{optimized}$

Initialize: Initialization of generator network G and student network $\mathcal{N}_{student}$

for $l = 1, 2, ..., total\ epochs$ **do**
  for $m = 1, 2, ..., SI$ **do** ▷ **Part1: Training with synthetic images**
    Generate random noise vector size of a batch: $\{z^i\}_{i=1}^{N}$;
    Generate the synthetic images from generator: $x_{synthetic} \leftarrow G(z)$;
    Input the generated synthetic images to teacher network:
      $y_{teacher}, f_{teacher}, \mu_{teacher}, \sigma_{teacher} \leftarrow \mathcal{N}_{teacher}(x_{synthetic})$;
    Calculate the generator loss $\mathcal{L}_G$ and update G;
    Input the synthetic images to student network: $y_{student} \leftarrow \mathcal{N}_S(x_{synthetic})$;
    Calculate student loss $\mathcal{L}_{KD}$ with Equation (7) and update $\mathcal{N}_{student}$;
  end for
  for $n = 1, 2, ..., RI$ **do** ▷ **Part2: Training with real dataset images**
    Input the real dataset images to student network:
      $y_{student}^{real} \leftarrow \mathcal{N}_{student}(x_{real})$;
    Calculate cross-entropy loss as student loss with Equation (4): $\mathcal{L}_{CE}$;
    Update student network $\mathcal{N}_{student}$;
  end for
end for

3.5. Loss Functions

In the proposed algorithm, loss functions of the generator consist of a total of four terms. The first loss function is one-hot loss $\mathcal{L}_{OH}$ [17]. If the synthesized images are similar to the real dataset images which was used to train teacher network, the output of the teacher network that receives the synthesized image as input will be closer to the one-hot vector. Since the total number of the classes for the image dataset is k and index of the classes is j, o^i of Equation (11) is a one-hot vector in which only the element with the largest value among $y^i_{teacher}$ that passes through the softmax function is 1, and the others are 0. Therefore, by training the generator with one-hot loss function, the output of the teacher network will be close to the one-hot vector when using synthesized images from generator as input.

$$o^i = \arg\max_j (y^i_{teacher})_j \tag{10}$$

$$\mathcal{L}_{OH} = \frac{1}{n}\sum_i \mathcal{H}_{CE}(y^i_{teacher}, o^i) \tag{11}$$

Next, an information entropy loss $\mathcal{L}_{IE}$ is introduced to maintain the class balance of generated synthetic images [17]. In Equation (12), p_j is the average logit in a class j. And $\mathcal{H}_{INFO}$ is the information entropy, which represents the amount of information p has. As the number of the class is k, $\mathcal{L}_{IE}$ has a minimum value when all p s are equal to $\frac{1}{k}$. Therefore, the frequency of each class of output prediction can be made uniform by optimizing information entropy loss function.

$$\mathcal{H}_{INFO}(p) = -\frac{1}{k}\sum_j p_j log p_j \tag{12}$$

$$\mathcal{L}_{IE} = -\mathcal{H}_{INFO}(\frac{1}{n}\sum_i y^i_{teacher}) \tag{13}$$

The intermediate feature extracted by the convolution layer of the neural network also has information about the input image. When the input image x_i is used, $f^i_{teacher}$ is the feature of the output right before the final fully-connected layer of the teacher network. The closer the input image is to the real dataset image, the higher the activation value of the feature map. Therefore, by using activation loss $\mathcal{L}_{ACT}$, the generator can synthesize images with a high activation value and make the synthesized image close to the real image dataset [17].

$$\mathcal{L}_{ACT} = -\frac{1}{n}\sum_i \left\| f^i_{teacher} \right\|_{L1} \tag{14}$$

The last loss function is batch normalization statistics loss $\mathcal{L}_{BNS}$ [22]. The distribution information for the training dataset can be extracted from the batch normalization layer of the neural network model trained with the corresponding dataset. When training a teacher network, i.e., a pre-trained model, means and variances of batch normalization layers are learned. When the synthesized images are used as input to the teacher network, the more the newly calculated means and variances based on synthetic images are similar to the pre-trained BNS, the distribution in which the synthesized images are generated becomes similar to the distribution of the real image dataset. Therefore, by using the BNS loss function to train the generator to reduce the difference between the fixed pre-trained BNS and the mean and variance calculated from the synthesized image, images generated from the generator become more similar to the distribution of training data.

$$\mathcal{L}_{BNS} = \sum_i \left\| \mu^{new}_i - \mu^{fixed}_i \right\|^2_{L2} + \left\| \sigma^{new}_i - \sigma^{fixed}_i \right\|^2_{L2} \tag{15}$$

Therefore, the final loss function for training the generator network composed of the aforementioned loss functions is written as follows.

$$\mathcal{L}_G = \alpha\mathcal{L}_{OH} + \beta\mathcal{L}_{IE} + \gamma\mathcal{L}_{ACT} + \delta\mathcal{L}_{BNS} \tag{16}$$

Here, $\alpha, \beta, \gamma, \delta$ are the coefficient for each loss function. Each coefficient is used as a hyperparameter to adjust the weight of each loss function. The values for α, β, γ are selected as 1, 5, and 0.1, respectively, which are the same values used in [17]. The performance is compared by increasing δ, which is the weight of $\mathcal{L}_{BNS}$, at intervals of 0.2 from 1 to 10. After that, the final weight values are selected as $\alpha = 1$, $\beta = 5$, $\gamma = 0.1$, and $\delta = 5$ in Equation (16).

4. Experiments

In this section, implementation details and results of the experiments conducted are explained. Also, an ablation study related to various loss functions of the proposed method will be reported.

4.1. Implementation Details

All of the experiments are conducted with the PyTorch [29] machine learning framework on two NVIDIA GeForce RTX 3090 GPUs. Details about the datasets and neural network models used are described below.

4.1.1. Datasets

There are two datasets used in the experiment of the proposed method, CIFAR-10 and CIFAR-100 dataset [23]. Both datasets consist of a total of 60,000 32×32 size, 3 channel color images, of which 50,000 images are for training and 10,000 images are for testing. CIFAR-10 is a dataset consisting of 10 classes, and CIFAR-100 is a dataset consisting of 100 classes. Accordingly, CIFAR-10 consists of 6000 images per class and CIFAR-100 consists of 600 images per class. Data augmentation methods such as random cropping, lighting, and random horizontal flipping were applied to the corresponding datasets, and were commonly applied to the teacher network model training, baseline training, general knowledge distillation training and proposed method.

4.1.2. Neural Network Model Settings

We used ResNet-34 [30] as a teacher network model and ResNet-18 [30] and ReActNet-18 [13] as a student network model. ReActNet-18 is a binarized neural network model based on ResNet-18, and knowledge distillation is used during training. Therefore, other model compression methods such as binarization were included in the experiment to show the effectiveness with our proposed method.

In all experiments, the dimension of noise input to generator was 1000, the batch size of syntheic images made from generator was 1024, and the number of training iterations through syntheic image was 120. The generator neural network model was the same as the model used in [17,31]. Adam optimizer [32] was used for generator network model training, and the initial learning rate was 1e-2, without additional learning rate scheduler. The same pre-trained ResNet-34 model was used for two different student network models.

For ResNet-18 student network model training, 200 epoch training was performed, and the mini batch size for the real image dataset was 128. Stochastic Gradient Descent (SGD) optimizer [33] was used, with Nesterov momentum of 0.9 and weight decay of 5×10^{-4}. The initial learning rate was 1×10^{-1}, and the learning rate was reduced by 10 times at 80 epochs and 120 epochs, respectively.

For ReActNet-18 training, we referred to the training settings of [13] and trained with two-step training strategy. In the first step, only activations of the neural network model are binarized, and in the second step, fine-tuning is performed based on the model parameter learned in the first step, and not only activations but also weights are binarized. In each step, the student network model was trained 256 epochs, and the mini-batch size for the

real image dataset was 128. Adam optimizer was used and weight decay was 1×10^{-5} at first stage and 0 at second stage. The initial learning rate was 5×10^{-4}, and the learning rate was linearly reduced from the initial value to 0.

4.2. Experiment Results

In Tables 1 and 2, dataset indicates the dataset used for the experiments, baseline means the training of the student network model without using knowledge distillation method. And KD refers to the method of training only by knowledge distillation method without using the proposed method. The hyperparameter settings of baseline and KD for ResNet-18 model and ReActNet-18 are the same as that used in the proposed method. All the values in the table are mean and standard deviation values of three repeated experiments of each case. More specifically, each values in the table are indicating *mean* $\pm$ *standard deviation*. In all cases, ResNet-34 was used as the teacher network model. ResNet-34 model was pretrained on each dataset and the accuracy was 95.54% on CIFAR-10 dataset and 77.62% on CIFAR-100 dataset. The time required for training with the proposed method in our experimental settings was about 108 s per epoch for the ResNet-18 model and about 129 s per epoch for the ReActNet-18 model.

Table 1. Experiment results for the ResNet-18 model are shown. ResNet-34 with 95.54% of accuracy was used as the teacher model for CIFAR-10 and ResNet-34 with 77.62% of accuracy was used as the teacher model for CIFAR-100. The baseline is the student network model used without a knowledge distillation method. KD refers to a basic knowledge distillation training. The proposed method is knowledge distillation with the proposed GIP method. Each value in the table includes the *mean* $\pm$ *standard deviation*, which is calculated after three repeated experiments.

Dataset	Baseline (%)	KD (%)	Proposed Method (%)
CIFAR-10	94.81 ± 0.04	94.96 ± 0.13	95.13 ± 0.08
CIFAR-100	77.22 ± 0.08	77.73 ± 0.08	78.30 ± 0.25

As shown in Table 1, with the CIFAR-10 dataset, the ResNet-18 model was improved by 0.17% compared to general response-based knowledge distillation training. Also, it was improved by 0.32% compared to the baseline. With the CIFAR-100 dataset, the ResNet-18 model was improved by 0.57% compared to the general KD method, and was improved by 1.08% compared to the baseline method.

In the case of ReActNet-18 model, as shown in Table 2, with the CIFAR-10 dataset, the ReActNet-18 model was improved by 0.91% compared to the KD method, and was improved by 1.20% compared to the baseline method. With the CIFAR-100 dataset, the ReActNet-18 model was improved 3.06% compared to the general KD method, and was improved by 3.40% compared to the baseline method.

Table 2. Experiment results for the ReActNet-18 model are shown. ResNet-34 with 95.54% of accuracy was used as the teacher model for CIFAR-10 and ResNet-34 with 77.62% of accuracy was used as the teacher model for CIFAR-100. The baseline is the student network model used without a knowledge distillation method. KD refers to a basic knowledge distillation training. The proposed method is knowledge distillation with the proposed GIP method. Each value in the table includes the *mean* $\pm$ *standard deviation*, which is calculated after three repeated experiments.

Dataset	Baseline (%)	KD (%)	Proposed Method (%)
CIFAR-10	92.17 ± 0.15	92.46 ± 0.10	93.37 ± 0.19
CIFAR-100	68.53 ± 0.14	68.87 ± 0.37	71.93 ± 0.05

The results show that the performance in all cases improved when using our proposed method compared to baseline and general knowledge distillation method. And the proposed method was more effectively improved the performance of the student network

model when the number of images per class is small, such as CIFAR-100. ResNet-18 model even outperformed the teacher network with the proposed method and exceeded 0.68% on the CIFAR-100 dataset. This means that the performance degradation of knowledge distillation training caused by the lack of dataset can be recovered through our proposed method.

More interestingly, the performance improvement of our proposed method was significant even when used with other model compression method. While neural network binarization has a high compression effect, it suffers from serious performance degradation. By leveraging our proposed knowledge distillation training strategy, performance degradation caused by parameter binarization can be recovered significantly.

We also conducted experiment to compare our proposed method with other related method. In [15], fake images synthesized from generator was augmented to the training dataset. It suggested to mix the real image dataset and fake images with certain ratio p_{fake}. Generator network and discriminator network was optimized with additional training process by using real image dataset. In the Table 3, *Teacher* refers to the performance of WideResNet-28-10 which is used for teacher network on each method. And *Student* refers to the performance of student network trained by each method. Since the performance values of teacher network in the two methods are different, new metric called *Restored ratio* is introduced. *Restored ratio* indicates the percentage of student network accuracy to teacher network accuracy. It is calculated by dividing the student accuracy with teacher accuracy of each method.

Table 3. Performance comparison with related work [15] on CIFAR-10 dataset. The result of GAN-TSC is from [15]. WideResNet-28-10 is used for teacher network and ResNet-18 model is used for student network on both methods. Since the performances of teacher network in the two methods are different, we introduce a metric, restored ratio, that represents the percentage of student network accuracy to teacher network accuracy.

Methods	**Teacher (WideResNet-28-10) (%)**	**Student (ResNet-18) (%)**	**Restored Ratio (%)**
GAN-TSC [15]	95.80	95.00	99.16
Our method	96.17	95.50 ± 0.04	99.30 ± 0.04

As shown in Table 3, *Restored ratio* of our proposed method was better than that of GAN-TSC in [15]. Our proposed method leverages knowledge of teacher network model which contains information related to the images that teacher network was trained and teacher network itself. Therefore, student network model trained with our method was better mimic the teacher network. Furthermore, our method does not require additional process of training discriminator network. Therefore, our proposed method successfully improved the performance of the related knowledge distillation method using GANs architecture.

Since the proposed method is closely related to Data-Free Learning (DAFL) [17], we compare the performance between the proposed method and the DAFL method using the Restored ratio metric. As shown in Table 4, the *Restored ratio* of our method was better than the DAFL method because the proposed method is using not only the synthesized images from the generator but also the real training dataset. This implies that synthesized images generated by the proposed method are compatible with the real training dataset for knowledge distillation training.

Table 4. Performance comparison with related work [17] on CIFAR-10 and CIFAR-100 datasets. The result for DAFL is extracted from [17]. ResNet-34 is used for the teacher network and the ResNet-18 model is used for the student network in both methods. Since the performance of the teacher network in the two methods are different, we introduce a metric, restored ratio, that represents the ratio of student network accuracy to teacher network accuracy.

Dataset	Method	Teacher (%)	Student (%)	Restored Ratio (%)
CIFAR-10	DAFL [17]	95.58	92.22	96.48
	Our method	95.54	95.13	99.57
CIFAR-100	DAFL [17]	77.84	74.47	95.67
	Our method	77.62	78.30	100.88

4.3. Ablation Study

In this section, we conduct massive ablation study related to various loss functions that we used for training the generator network. The ablation study is conducted on our proposed knowledge distillation training strategy with CIFAR-100 dataset and ResNet-18 model. 200 epochs training was performed with the mini batch size of 256 for the real image dataset. The initial learning rate was 1×10^{-1}, and the learning rate was decreased by 10 times at 80 epochs and 120 epochs. Stochastic Gradient Descent optimizer was used, with Nesterov momentum of 0.9 and weight decay of 5×10^{-4}. Other experimental settings are equivalent to preceding experiment of our proposed method except generator loss functions. Every possible combinations of the generator loss functions are tested. The ablation study was repeated three times for each combinations and test accuracy of each combinations are reported with its mean value.

As shown in Table 5, some combinations of the generator loss functions are not suitable for our proposed knowledge distillation training strategy. When the information entropy loss functions are not included, the student network model was not trained except for the case which only the BNS loss function was included. It shows that maintaining the class balance of the synthetic images plays an important role for training the student network with the proposed method. Interestingly, when the BNS loss function is used only, it was possible to train the student network, even though the information entropy loss function was not included. It implies that BNS also inhere the knowledge related to the class balance of the dataset which was used for training of corresponding batch normalization layers. Accordingly, the combination of the information entropy loss function and the BNS loss function showed the third good performance which achieves 77.41%. When the one-hot loss function is included to this combination, it becomes the second good performance case which achieves 77.45%. Finally, when all of the loss functions that we suggested are used, the performance of the student model achieves 77.68% which is the best among the combinations.

Ablation study experiments we conducted suggest that each of the generator loss functions are important for our proposed method. And the experiments also showed that the absence of a specific loss function made the student network completely untrained.

Also, further experiments were conducted to evaluate the effectiveness of the training epoch on the performance of the proposed method. The evaluation was conducted on both ResNet-18 model and ReActNet-18 model with CIFAR-100 dataset. The ResNet-18 model was trained for 100 epochs for the case of fewer epochs and 300 epochs for the case of more epochs. The ReActNet-18 model was trained for 128 epochs for the case of fewer epochs and 384 epochs for the case of more epochs.

As shown in Table 6, for ResNet-18, increasing training epochs does not always improve accuracy level. The result after 200 epochs, which is the result of the proposed method, provides better results than after 300 epochs. However, for ReActNet-18 shown in Table 7, the accuracy gain and number of training epochs show a positive correlation. This is an interesting result and worth further exploration.

Table 5. Ablation study experiment results of CIFAR-100 datasets.

One-Hot Loss	Information Entropy Loss	Activation Loss	BNS Loss	Accuracy (%)
✓				1.00
	✓			77.07
		✓		1.00
			✓	77.14
✓	✓			77.03
✓		✓		1.00
✓			✓	1.00
	✓	✓		76.83
	✓		✓	77.41
		✓	✓	1.00
✓	✓	✓		77.05
✓	✓		✓	77.45
✓		✓	✓	76.62
	✓	✓	✓	76.81
✓	✓	✓	✓	77.68

Table 6. Experiment of comparing the performance of different training epochs for ResNet-18 model on CIFAR-100 dataset.

Model	Accuracy (%)		
	100 Epochs	200 Epochs	300 Epochs
ResNet-18	76.51	78.30	78.15

Table 7. Experiment of comparing the performance of different training epochs for and ReActNet-18 model on CIFAR-100 dataset.

Model	Accuracy (%)		
	128 Epochs	256 Epochs	384 Epochs
ReActNet-18	70.96	71.93	72.23

5. Conclusions

To further improve the performance of knowledge distillation training, we proposed a generative image processing based knowledge distillation training strategy that leverages the inherent knowledge of the pre-trained teacher network model. The generator network does not need to be pre-trained to synthesize the images by processing the random noise. It is simply trained while undergoing knowledge distillation training.

Experiments were conducted on widely used ResNet-18 models. In addition, experiments with ReActNet-18, an extremely quantized network that uses only 1-bit values for weight and activation parameters, were also performed to show the effectiveness of this method in conjunction with other model compression methods such as neural network binarization. The experiments showed that our proposed method significantly improved the performance of the knowledge distillation training.

Author Contributions: Conceptualization, Y.H. and S.L.; methodology, Y.H.; software, Y.H.; validation, Y.H., S.L.; formal analysis, Y.H.; investigation, Y.H.; resources, Y.H.; data curation, Y.H.; writing—original draft preparation, Y.H.; writing—review and editing, Y.H. and S.L; visualization,

Y.H.; supervision, S.L.; project administration, S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available in CIFAR-10 and CIFAR-100 dataset https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 4 July 2022).

Acknowledgments: This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-01906, Artificial Intelligence Graduate School Program (POSTECH)).

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

References

1. Cheng, Y.; Wang, D.; Zhou, P.; Zhang, T. Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges. *IEEE Signal Process. Mag.* **2018**, *35*, 126–136. https://doi.org/10.1109/MSP.2017.2765695.
2. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, *2*. arXiv:1503.02531.
3. Bucilua, C.; Caruana, R.; Niculescu-Mizil, A. Model compression. In Proceedings of the 12 th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, 20–23 August 2006; Volume 3.
4. Wu, J.; Leng, C.; Wang, Y.; Hu, Q.; Cheng, J. Quantized Convolutional Neural Networks for Mobile Devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
5. Denton, E.L.; Zaremba, W.; Bruna, J.; LeCun, Y.; Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. *Adv. Neural Inf. Process. Syst.* **2014**, *27*. Available Online: https://proceedings.neurips.cc/paper/2014/file/2afe4567e1bf64d32a5527244d104cea-Paper.pdf (accessed on 2 July 2022).
6. Han, S.; Pool, J.; Tran, J.; Dally, W. Learning both weights and connections for efficient neural network. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. Available Online: https://proceedings.neurips.cc/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf (accessed on 4 July 2022).
7. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2015**. arXiv:1510.00149.
8. Courbariaux, M.; Bengio, Y.; David, J.P. Binaryconnect: Training deep neural networks with binary weights during propagations. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. Available Online: https://proceedings.neurips.cc/paper/2015/file/3e15cc11f979ed25912dff5b0669f2cd-Paper.pdf (accessed on 4 July 2022).
9. Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to + 1 or −1. *arXiv* **2016**. arXiv:1602.02830.
10. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proceedings of the European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 525–542.
11. Zhou, S.; Wu, Y.; Ni, Z.; Zhou, X.; Wen, H.; Zou, Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv* **2016**. arXiv:1606.06160.
12. Liu, Z.; Shen, Z.; Savvides, M.; Cheng, K.T. Reactnet: Towards precise binary neural network with generalized activation functions. In *Proceedings of the European Conference on Computer Vision*; Springer: Cham, Switzerland, 2020; pp. 143–159.
13. Chen, T.; Zhang, Z.; Ouyang, X.; Liu, Z.; Shen, Z.; Wang, Z. "BNN - BN = ?": Training Binary Neural Networks Without Batch Normalization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Nashville, TN, USA, 20–25 June 2021; pp. 4619–4629. Available Online: https://openaccess.thecvf.com/content/CVPR2021W/BiVision/html/Chen_BNN_-_BN___Training_Binary_Neural_Networks_Without_CVPRW_2021_paper.html (accessed on 6 July 2022).
14. Gou, J.; Yu, B.; Maybank, S.J.; Tao, D. Knowledge distillation: A survey. *Int. J. Comput. Vis.* **2021**, *129*, 1789–1819.
15. Liu, R.; Fusi, N.; Mackey, L. Teacher-student compression with generative adversarial networks. *arXiv* **2018**. arXiv:1812.02271.
16. Micaelli, P.; Storkey, A.J. Zero-shot knowledge transfer via adversarial belief matching. *Adv. Neural Inf. Process. Syst.* **2019**, *32*. Available Online: https://proceedings.neurips.cc/paper/2019/file/fe663a72b27bdc613873fbbb512f6f67-Paper.pdf (accessed on 2 July 2022).
17. Chen, H.; Wang, Y.; Xu, C.; Yang, Z.; Liu, C.; Shi, B.; Xu, C.; Xu, C.; Tian, Q. Data-free learning of student networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3514–3522.
18. Luo, L.; Sandler, M.; Lin, Z.; Zhmoginov, A.; Howard, A. Large-scale generative data-free distillation. *arXiv* **2020**. arXiv:2012.05578.
19. Yu, X.; Yan, L.; Ou, L. Conditional Generative Data-Free Knowledge Distillation based on Attention Transfer. *arXiv* **2021**. arXiv:2112.15358.
20. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.* **2014**, *27*. Available Online: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf (accessed on 1 July 2022).

21. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; PMLR: Lille, France, 2015; pp. 448–456.
22. Xu, S.; Li, H.; Zhuang, B.; Liu, J.; Cao, J.; Liang, C.; Tan, M. Generative low-bitwidth data free quantization. In *Proceedings of the European Conference on Computer Vision*; Springer: Cham, Switzerland, 2020, pp. 1–17.
23. Krizhevsky, A.; Hinton, G.; Learning multiple layers of features from tiny images. *Technical Report* **2009**. Available Online: http://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf (accessed on 4 July 2022).
24. Romero, A.; Ballas, N.; Kahou, S.E.; Chassang, A.; Gatta, C.; Bengio, Y. Fitnets: Hints for thin deep nets. *arXiv* **2014**. arXiv:1412.6550.
25. Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828.
26. Heo, B.; Lee, M.; Yun, S.; Choi, J.Y. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 3779–3787.
27. Komodakis, N.; Zagoruyko, S. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In Proceedings of the ICLR, Toulon, France, 24–26 April 2017.
28. Yim, J.; Joo, D.; Bae, J.; Kim, J. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4133–4141.
29. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; De Vito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic differentiation in pytorch. **2017**. Available Online: https://openreview.net/pdf?id=BJJsrmfCZ (accessed on 9 November 2022).
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
31. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv* **2015**. arXiv:1511.06434.
32. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**. arXiv:1412.6980.
33. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**. arXiv:1609.04747.