


Article

Ensemble-Guided Model for Performance Enhancement in Model-Complexity-Limited Acoustic Scene Classification

Seokjin Lee ^{1,2,*} , Minhan Kim ¹, Seunghyeon Shin ¹, Seungjae Baek ¹, Sooyoung Park ³ and Youngho Jeong ³

¹ School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, Korea; kmh7576@knu.ac.kr (M.K.); sineva123@gmail.com (S.S.); bgsoj61@naver.com (S.B.)

² School of Electronics Engineering, Kyungpook National University, Daegu 41566, Korea

³ Media Research Division, Electronics and Telecommunications Research Institute, Daejeon 34129, Korea; sooyoung@etri.re.kr (S.P.); yhcheong@etri.re.kr (Y.J.)

* Correspondence: sjlee6@knu.ac.kr; Tel.: +82-53-950-5523

Abstract: In recent acoustic scene classification (ASC) models, various auxiliary methods to enhance performance have been applied, e.g., subsystem ensembles and data augmentations. Particularly, the ensembles of several submodels may be effective in the ASC models, but there is a problem with increasing the size of the model because it contains several submodels. Therefore, it is hard to be used in model-complexity-limited ASC tasks. In this paper, we would like to find the performance enhancement method while taking advantage of the model ensemble technique without increasing the model size. Our method is proposed based on a mean-teacher model, which is developed for consistency learning in semi-supervised learning. Because our problem is supervised learning, which is different from the purpose of the conventional mean-teacher model, we modify detailed strategies to maximize the consistency learning performance. To evaluate the effectiveness of our method, experiments were performed with an ASC database from the Detection and Classification of Acoustic Scenes and Events 2021 Task 1A. The small-sized ASC model with our proposed method improved the log loss performance up to 1.009 and the F₁-score performance by 67.12%, whereas the vanilla ASC model showed a log loss of 1.052 and an F₁-score of 65.79%.

Keywords: acoustic scene classification; low model complexity; consistency learning; mean-teacher model



Citation: Lee, S.; Kim, M.; Shin, S.; Baek, S.; Park, S.; Jeong, Y.

Ensemble-Guided Model for Performance Enhancement in Model-Complexity-Limited Acoustic Scene Classification. *Appl. Sci.* **2022**, *12*, 44. <https://doi.org/10.3390/app12010044>

Academic Editor: Jongweon Kim

Received: 24 November 2021

Accepted: 17 December 2021

Published: 21 December 2021

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



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Machine learning techniques, which utilize a lot of data to tackle problems, have improved a lot in recent years. In research topics in acoustic signal processing, machine recognition of sound signals has been of particular researchers' interest [1–4] to understand a user's situation or find particular events. Such machine-learning-based acoustic signal analysis and classification techniques have been recently studied in various purposes, including anomalous machine sound detection [5], monitoring domestic activity [6], sound event localization and detection [7]. Moreover, the sound classification techniques have been applied in the medical systems, e.g., snore sound classification to detect obstructive sleep apnea [8] and bio-inspired voice evaluation [9].

Especially, the machine-based acoustic environment understanding task is related to acoustic scene classification (ASC), which focuses on the recognition of long audio clips, e.g., a 10-second clip, to classify an acoustic environment. Research for ASC tasks started to classify a set of classes, including people, voices, subways, and traffic, with recurrent neural networks and a k-nearest neighbor criterion [10], which was later expanded to a technique using hidden Markov model [11]. Recent techniques for ASC tasks utilize recently developed models based on convolutional neural network (CNN) families [12–14], such as ResNet [15] and Inception [16].

State-of-the-art ASC models use auxiliary techniques to enhance the performance, e.g., model ensembles and data augmentations. In particular, ASC systems with model

ensemble techniques have shown good performance in ASC competitions. In the ASC competition of the Detection and Classification of Acoustic Scenes and Events (DCASE) 2020 Task 1A, 7 models of the top 10 models used ensembles of subsystems. The best models of Suh et al. [12], Hu et al. [17], and Gao and McDonnell [18], which placed first, second, and third team ranking, consisted of ensembles of three, eight, and three subsystems, respectively. In the ASC task, the ensemble of various model instances seems to be helpful.

On the other hand, the ASC and sound event detection (SED) tasks have been expanded to various tasks with specific conditions, e.g., semi-supervised [19], multiple devices, or model-complexity-limited conditions [20]. The semi-supervised SED tasks utilize a dataset that consists of strongly-labeled, weakly-labeled, and unlabeled data to train the classifier, and the ASC with multiple devices utilizes a dataset that consists of data recorded with multiple devices whose frequency characteristics are different from each other. The model-complexity-limited ASC is to construct a small-size model to classify acoustic scenes.

Because the small-size ASC model is developed considering mobile devices, there is an upper limit to the model size, e.g., 128 KB. In order to satisfy the complexity constraints, researchers have tried various techniques to reduce the size of large ASC models, including quantization [21], decomposition [22], and pruning [23]. Each algorithm has contributed to the model size reduction in its own way, but the performance aspect was rather complicated [14].

We want to approach the model-complex-limited ASC problem from a different perspective, to develop an auxiliary enhancement method that can be easily applied and that can be applied independently of the existing technique. In order to achieve this goal, we first started with the ensemble technique. As mentioned earlier, the ensemble technique is simple and effective to enhance the performance, but it requires large complexity because it needs several subsystems. For example, if we want to enhance the performance with an ensemble of three subsystems of 128 KB, the entire system requires at least 384 KB. Therefore, the ensemble technique cannot be applied directly to the model-complexity-limited ASC problem, because requiring a large-size model is a serious penalty in the complexity-limited problem.

In order to develop the performance enhancement technique without any additional model size, we turned our attention to another sound classification: the semi-supervised SED task. In order to deal with the weakly labeled and unlabeled data, several self-ensemble methods have been developed, including the Π -model [24], temporal ensembling [24], and mean-teacher model [25]. The Π -model ensembles several model networks that are generated through random dropout of the target model, so it requires multiple inference steps in a single training step. The temporal ensembling technique utilizes the ensemble effect by time-averaged prediction with exponential tapering. It requires only one inference step per each training step, same as an ordinary training procedure, but it requires additional memory to save time-averaged prediction results. The mean-teacher model removes the additional memory requirements by using a teacher model consisting of temporally moving-averaged weights to generate ensembled prediction results.

Techniques that utilize ensemble-guided structures, including the temporal ensembling and mean-teacher model techniques, are not useful generally in supervised ASC tasks. Making an ensemble system of several subsystems may be more useful. However, ensemble-guided structures do not require additional model complexity, so they may be useful in the model-complexity-limited ASC task, if they are applied carefully. Therefore, the purpose of this paper is twofold: to establish a parameter control strategy suitable for supervised learning, and to verify that the ensemble-guided model is also effective in model-complexity-limited supervised learning.

2. Problem Description

The ASC task is to classify audio data into one of known acoustic scene classes, e.g., airport, indoor shopping mall, or metro station, as shown in Figure 1. The ASC tasks

have been investigated to help understand an environment with sound signals. ASC tasks have a relatively long history among machine-learning-based sound signal analyses. The challenge on DCASE, which is one of the famous competitions related to the analysis of sound signals, has dealt with the ASC problem since 2013 [26]. Nowadays, the task has been developed for complicated and target-specific problems.

One of the interesting problems of the ASC task is implementation on mobile devices in recent ASC research. To achieve this goal, two issues should be considered: The performance should not be affected by the difference in response between devices, and the complexity of the ASC model should be as small as possible. Therefore, ASC models have been developed to process input signals from multiple devices with a certain complexity limit, as shown in Figure 1.

As mentioned in Section 1, there are several techniques to reduce the complexity of existing ASC models, but they are not of interest in this paper. Our problem is to develop a method that can enhance an ASC model without using additional model complexity. In order to develop the method, we focus on self-ensemble techniques of semi-supervised learning tasks.

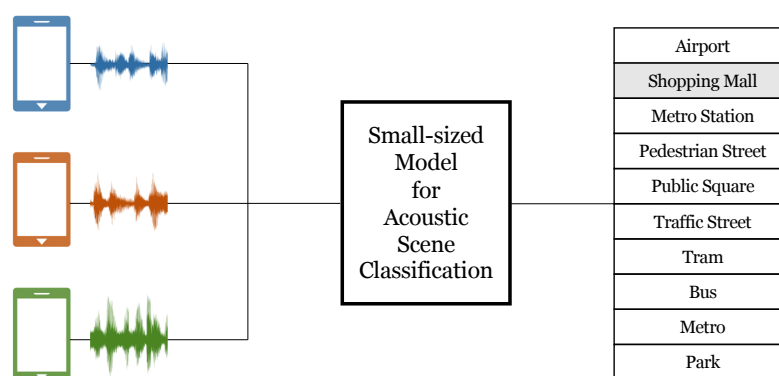


Figure 1. Problem description for the model-complexity-limited acoustic scene classification (ASC) task.

3. Ensemble-Guided Models

3.1. Π -Model and Temporal Ensembling for Semi-Supervised Learning

This section describes self-ensembling models for the semi-supervised learning. In early research works for the semi-supervised task, there were two mainly improved methods that surpass previous research works: Π -model and temporal ensembling [24]. The Π -model utilizes the consistency between same network structures with different dropouts in order to handle the unlabeled data. As shown in Figure 2, the input data are processed twice by networks with different dropouts to generate two inference results, i.e., \mathbf{z}_i and $\tilde{\mathbf{z}}_i$. There are two components of loss function. The first is a classification cost between the inference results and the labels, which is commonly measured by cross-entropy. The second is a consistency cost between the models with different dropouts. The consistency cost can be measured by any distance functions, but the mean-squared error is a good choice in the previous research works. The Π -model successfully provides a prototype of consistency learning, but it has two main drawbacks: The reference of consistency, $\tilde{\mathbf{z}}_i$, is noisy because it is simply generated by another model with a random dropout, and the model should perform the inference step multiple times for a training step, thus making the training step slow.

In order to compensate for the drawbacks of the Π -model, a temporal ensembling technique is developed as shown in Figure 2. The temporal ensembling generates the reference of consistency $\tilde{\mathbf{z}}_i$ by exponential moving average of inference outputs of previous epochs, instead of another model with different dropout, as [24]

$$\mathbf{z}_{ema}(n) = \alpha \mathbf{z}_{ema}(n - 1) + (1 - \alpha) \mathbf{z}(n) \tag{1}$$

$$\tilde{\mathbf{z}}(n) = \frac{\mathbf{z}_{ema}(n)}{1 - \alpha^n} \tag{2}$$

where n is an epoch number, \mathbf{z} and \mathbf{z}_{ema} are the prediction result and its exponential moving average, respectively, and $\tilde{\mathbf{z}}$ is a target vector with bias correction. The temporal ensembling makes better results with faster speed than the Π -model; however, it requires additional storage space to save the ensemble predictions of the whole epoch, and the implementation is complicated.

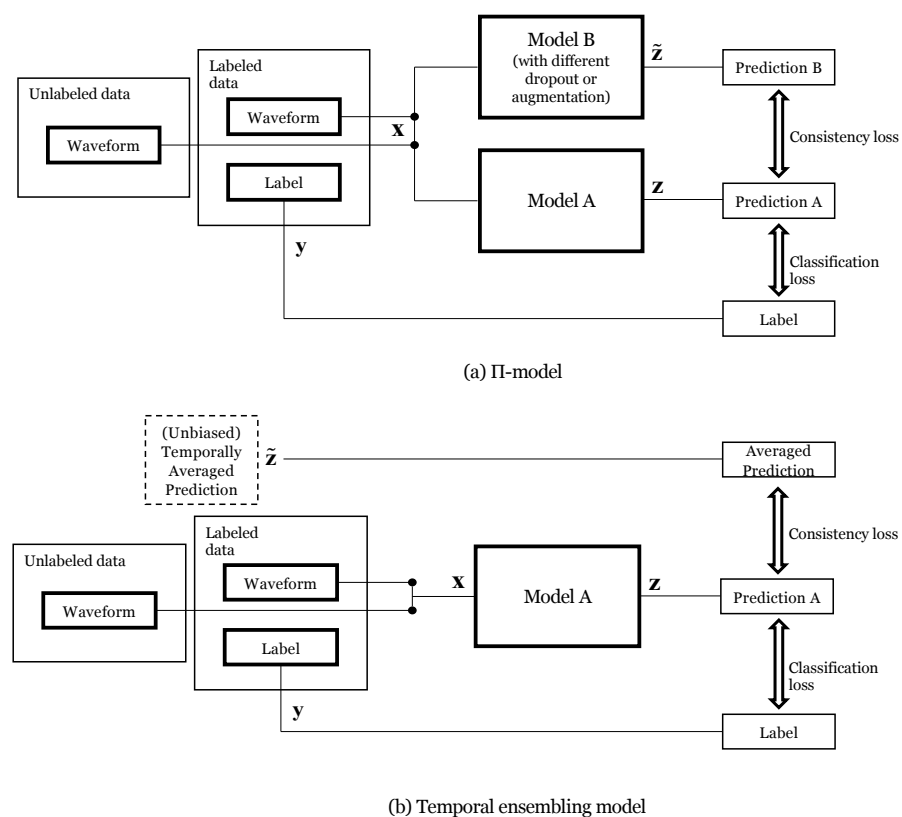


Figure 2. Schematic diagrams of consistency learning models.

3.2. Mean-Teacher Model for Semi-Supervised Learning

The mean-teacher model is a consistency learning structure based on a student–teacher model, as shown in Figure 3. The weights of student modules are trained as ordinary neural networks, but the weights of teacher modules are fixed during training steps. The teacher model weights are updated as an exponentially tapered moving average as follows at the end of the training epoch [25]:

$$\theta_{teacher}(n) = \alpha \theta_{teacher}(n - 1) + (1 - \alpha) \theta_{student}(n) \tag{3}$$

where $\theta_{teacher}$ and $\theta_{student}$ are the model parameters of the teacher and student models, respectively. The loss function is defined with the combination of classification and consistency costs as

$$C(\theta_{student}) = C_{class}(\theta_{student}) + \beta w(n) C_{consist}(\theta_{student}) \tag{4}$$

where C_{class} and $C_{consist}$ are the classification and consistency cost functions, respectively; β is a global weight coefficient; and $w(n)$ is a time-varying weight function for the consistency cost. The classification cost is commonly defined as the cross-entropy between the prediction results and the labels if exist, and the consistency cost is defined as mean-squared error between outputs from student and teacher models.

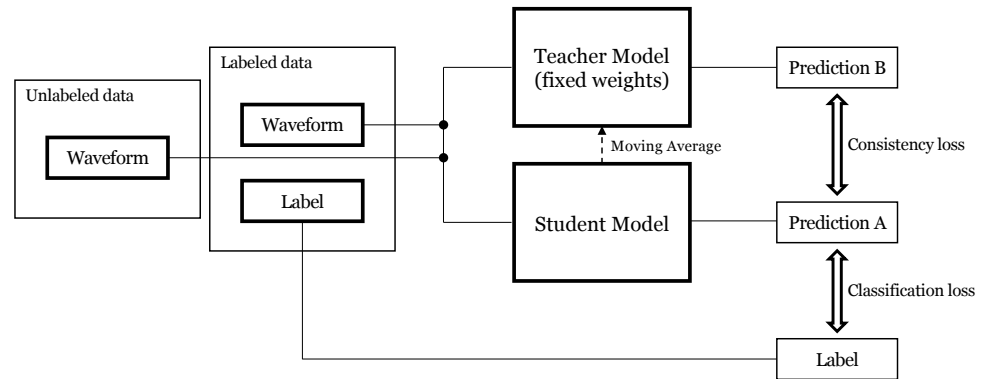


Figure 3. Schematic diagram of the mean-teacher model for semi-supervised learning.

3.3. Proposed Ensemble-Guided Model for Model-Complexity-Limited ASC

Our ensemble-guided model for the model-complex-limited ASC task is based on the mean-teacher model [25]. However, the original mean-teacher model is developed for the semi-supervised problem, so we modified the model to suit our model-complexity-limited supervised problem. Of course, there is no path for the unlabeled data, because our model is used for the enhancement of supervised learning without increasing the model complexity.

Then, the time-varying weight function $w(n)$ is modified. In order to handle the unlabeled data, the time-varying weight is a ramp-up function from the start of training, e.g., $\exp[-5(1-r)]$, where r is linearly increased from 0.0 to 1.0 during the ramp-up period, e.g., first 80 epochs, in the conventional mean-teacher model. However, in supervised learning, early application of consistency loss degrades the performance. It seems that the performance is degraded as the teacher model, which has become an ensemble of immature student models, guides the learning of the student model. To prevent degradation, the weight function is modified into a unit step function as

$$w(n) = \begin{cases} 0 & \text{if } n < n_{act_ems} \\ 1 & \text{if } n \geq n_{act_ems} \end{cases} \quad (5)$$

where n_{act_ems} is the number of the start epoch of consistency learning. Furthermore, the tapering parameter α is set to 0.0 for $n < n_{act_ems}$ to prevent ensembling of immature student models. We also propose a strategy to choose an appropriate n_{act_ems} at the epoch number at which the performance of the vanilla model is saturated.

Lastly, the strategy of adding noise is modified. In the conventional mean-teacher model, Gaussian noise is added to both inputs of the student and teacher models throughout the entire training process. However, Gaussian noise is added to the input data of the student model only when $n > n_{act_ems}$ in our model to prevent negative effect on the early training process of supervised learning.

4. Evaluation

4.1. Evaluation Settings

In order to evaluate the ensemble-guided model for the model-complexity-limited task, evaluations with development data from DCASE 2021 Task 1A [20] were performed. The audio data of the task consisted of recordings from 12 cities in 10 acoustic scene classes with 4 recording devices, and synthetic data from 11 mobile devices [27]. Details of the recording devices and environments can be found in [4].

Each audio clip was 10 s long with a sampling frequency of 44.1 kHz, and it was transformed into a 256-bin mel-frequency spectrum with a 2048-sample Hanning window and a 50% overlap. The input features, which were fed to the models, consisted of three channels of the log mel spectrum and its delta and delta-delta values.

In recent studies, most ASC models have been developed based on CNN families, e.g., ResNet [15], Inception [16], and their modifications. Most of the models submitted to DCASE 2020 Task 1 and DCASE 2021 Task 1A, which are competitions in the ASC, consisted of ResNet and Inception, and variants of ResNet such as BC-ResNet [28] and SE-ResNet [29]. Therefore, we constructed the CNN-based architecture for the ASC. Especially, we constructed multi-path structure, which divided the features into multiple frequency groups, inspired by the Trident model [12], which showed the best performance in DCASE 2020 Task 1A. However, the Trident model was significantly large, so we applied canonical polyadic (CP) ResNet [22], which showed good performance with small-sized model in DCASE 2020 Task 1B, to the multi-path structure.

The student model was a dual-path structure consisting of two subnetworks responsible for the low (from 1st to 128th bins) and high (from 129th to 256th bins) frequency components, as shown in Table 1. In Table 1, CNN ($a \times b, c$) means a convolutional layer with an ($a \times b$)-sized kernel and c output channels, and Residual Block ($a \times b$) means a residual block with a output channels and b depthwise repetition. The dual-path structure was inspired by the Trident structure [12]. The residual block consisted of the ResNet block [15] with complexity reduction by canonical polyadic (CP) decomposition [22] and parameter sharing [30], as shown in Figure 4. The model weights were quantized to a 16-bit floating point after training, so the model size became 125.8 KB and satisfied the complexity limit of DCASE 2021 Task 1A, which is 128 KB.

Table 1. Structure of the student model. CNN ($a \times b, c$) means a convolutional layer with ($a \times b$)-sized kernels and c output channels, and Residual Block ($a \times b$) means a residual block with a output channels and b depthwise repetition.

Low Frequency (1st–128th bins)	High Frequency (129th–256th bins)
CNN ($1 \times 1, 32$)	CNN ($1 \times 1, 32$)
Batch Normalization	Batch Normalization
ReLU Activation	ReLU Activation
Residual Block (32×2)	Residual Block (32×2)
Max Pooling (2×2)	Max Pooling (2×2)
Residual Block (64×2)	Residual Block (64×2)
Max Pooling (2×2)	Max Pooling (2×2)
Residual Block (64×2)	Residual Block (64×2)
CNN ($1 \times 1, 32$)	CNN ($1 \times 1, 32$)
CNN ($1 \times 1, 64$)	CNN ($1 \times 1, 64$)
Batch Normalization	Batch Normalization
Concatenation along freq. axis	
CNN ($1 \times 1, 10$)	
Global Average Pooling	
Softmax Activation	

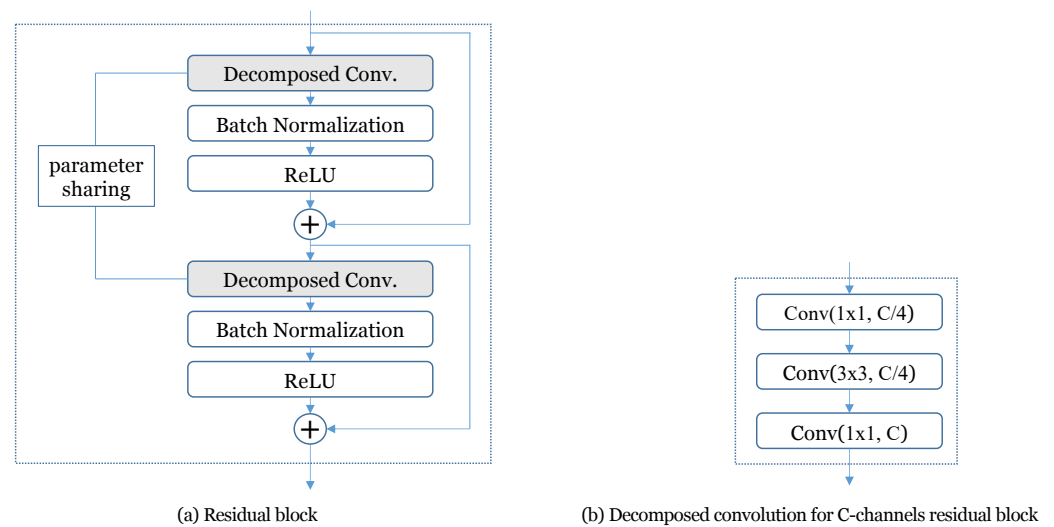


Figure 4. Block diagrams of (a) the residual block and (b) CP-decomposed convolution block in Table 1. C in (b) is the number of channels of the residual block.

As mentioned earlier, the objective function consists of the classification and the consistency cost functions. The classification cost function was defined as the categorical cross-entropy between the prediction results of the student model and the class labels, and the consistency cost function was defined as the mean squared error between the prediction results of the student and teacher models. The tapering parameter α was set to 0.99, and the consistency cost weight β was set to 2.5. Stochastic gradient descent [31] was used as the optimization algorithm, and the learning rate was controlled by a cosine annealing scheduler with restart epochs of [2, 7, 15, 30, 60, 90, 120, 150, 180, 210, 240] and has an initial value of 0.05 and a minimum value of 10^{-5} , as shown in Figure 5. The maximum value was decayed by 0.9 at each restart epoch. We observed that supervised learning for the vanilla model (without ensemble guidance) was saturated between 230 and 260 epochs, even with more learning rate restarts. Accordingly, we set the start epoch n_{act_ems} to 260, and the batch size was set to 32.

The development dataset of DCASE 2021 Task 1A consisted of 13,962-clip training data and 2968-clip evaluation data. We used 80% of the training data to train our model and 20% of the data as the validation data to choose the best model during the training procedure. The training process was performed for 600 epochs, and the best model was selected as the model with the smallest classification cost for the validation data. The optimization performance was measured by cross-entropy loss (log loss), and the accuracy performance was measured by the macro-averaged F_1 -score, the same as the criteria in the DCASE 2021 task 1A. In the cases of mean-teacher-based model, the performances of the student models were measured. The cross-entropy loss and macro-averaged F_1 -score were measured by Scikit-learn python package [32].

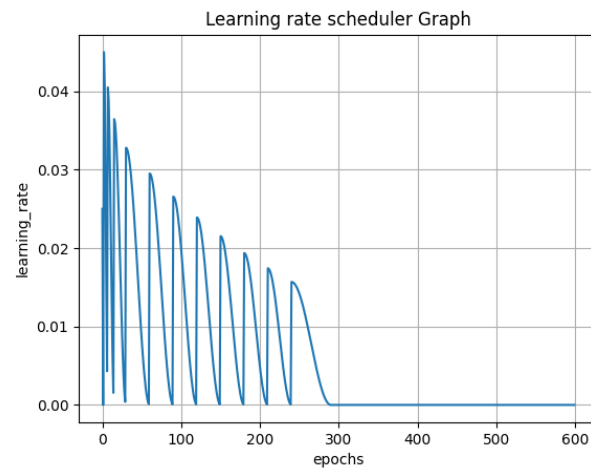


Figure 5. Learning rate scheduling.

4.2. Performance Comparison with Other Ensemble-Guided and Vanilla Models

In order to evaluate our model, we compared the performance of the proposed model with those of the conventional mean-teacher method, the temporal ensembling models [24], and the vanilla model without any ensemble guidance. The consistency cost weight β and the tapering parameter α of the conventional mean-teacher model were set to 2.5 and 0.99, respectively, which are the same as those of the proposed model. The time-varying weight $w(n)$ was set as a ramp-up function as $\exp[-5(1-r)]$ with linearly increased r from 0.0 to 1.0 during the first 80 epochs for the conventional temporal ensembling and mean-teacher models, which are the same as the values in [24] and [25]. The consistency cost weight of the temporal ensembling was set to 2 because it showed the best performance among [0.5, 1, 2, 3, 4, 5]. All the tested models had the same model size of 125.8 KB with 16-bit quantization after training, the same as in the vanilla model.

Table 2 shows the representative performance of the proposed model with the performances of the vanilla model and other ensemble-guided models. Comparing the performance of the proposed model (log loss of 1.009 and F_1 -score of 67.12%) to that of the vanilla model (log loss of 1.052 and F_1 -score of 65.79%), our ensemble-guided model can enhance the optimization and accuracy performances of the ASC model without increasing the model size.

Table 2. Performance comparison between the proposed model, the conventional mean-teacher model, the temporal ensembling models, and the vanilla model without any ensemble guidance.

Model	Log Loss	Macro-Averaged F_1 -Score	Model Size
Vanilla	1.052	65.79%	125.8 KB
Conventional Temporal Ensembling	1.042	62.26%	125.8 KB
Conventional Mean-Teacher	1.069	66.47%	125.8 KB
Temporal Ensembling w/ Unit Step $w(n)$	1.023	65.93%	125.8 KB
Modified Mean-Teacher (Proposed)	1.009	67.12%	125.8 KB

Simple application of temporal ensembling, which is a consistency learning technique for semi-supervised learning, can enhance slightly the log loss performance, but the F_1 -score performance is severely degraded. Simple application of the mean-teacher method can enhance slightly the F_1 -score, but the log loss performance is degraded. On the other hand, the proposed model can successfully enhance the performance, showing that our modification to suit to the supervised learning problem is meaningful. The temporal ensembling with modified $w(n)$ as (5) and n_{act_ems} of 260, which are the same as the settings of the proposed model, shows better performance than the conventional temporal

ensembling method. However, the proposed model still performs better even though it does not require additional space to save averaged prediction results, whereas the temporal ensembling method requires the additional space.

Table 3 shows the performance comparison with the state-of-the-art performances. The performance values of compared models in Table 3 are from the values reported in the original papers. The tested model with the proposed mean-teacher structure shows considerably improved performance compared to the baseline model, but slightly degraded performance compared to the state-of-the-art systems. They use the same database to train their models, but have different training methodologies including extraction and pre-processing of features, learning rate control, optimizer, etc. Therefore, it is difficult to analyze exactly which factors cause the performance differences, but we think that the effective uses of the model lightweight and data augmentation methods may be the dominant factors.

Table 3. Performance comparison with the state-of-the-art performances. All performances are presented using the values reported in referenced papers.

Method	Log Loss	Macro-Averaged F ₁ -Score	Model Size
DCASE 2021 Task 1A Baseline (in [20])	1.47	47.7%	90.3 KB
Vanilla model (ours)	1.05	65.8%	125.8 KB
Mean-Teacher model (ours)	1.01	67.1%	125.8 KB
Liu’s model (in [21])	0.92	68.0%	42.5 KB
Liu’s model + Feature Reuse (in [21])	0.91	68.2%	106.7 KB
Koutini’s model (in [14])	0.89	69.4%	127.7 KB
Koutini’s model + Domain Adaptation (in [14])	0.88	69.5%	127.5 KB
Heo’s model (in [33])	-	68.5%	124.1 KB
Heo’s model + Knowledge Distillation (in [33])	-	70.5%	124.1 KB

Additionally, there are interesting results regarding the performance-enhancing technique, which is the subject of this paper. As shown in Table 3, several performance improvement techniques specific to their models have been tried, e.g., feature reuse in [21], domain adaptation in [14], and knowledge distillation [33]. The techniques have F₁-score improvements of 0.2%, 0.1%, and 2%, respectively, and log loss improvements of about 0.01. Therefore, the 1.3% F₁-score improvement and the 0.04 log loss improvement of the proposed mean-teacher model may be meaningful.

Figure 6 shows the classwise F₁-score performances of the vanilla and proposed models. Comparing the F₁-scores of the vanilla model and the proposed model, the performance improvements of two classes are notable: the public square (5.74% improvement) and the pedestrian street (5.81% improvement), indicated by green arrows in Figure 6. These two classes showed the lowest performance in the vanilla model, so the results show that the proposed method can improve the low-performance classes.

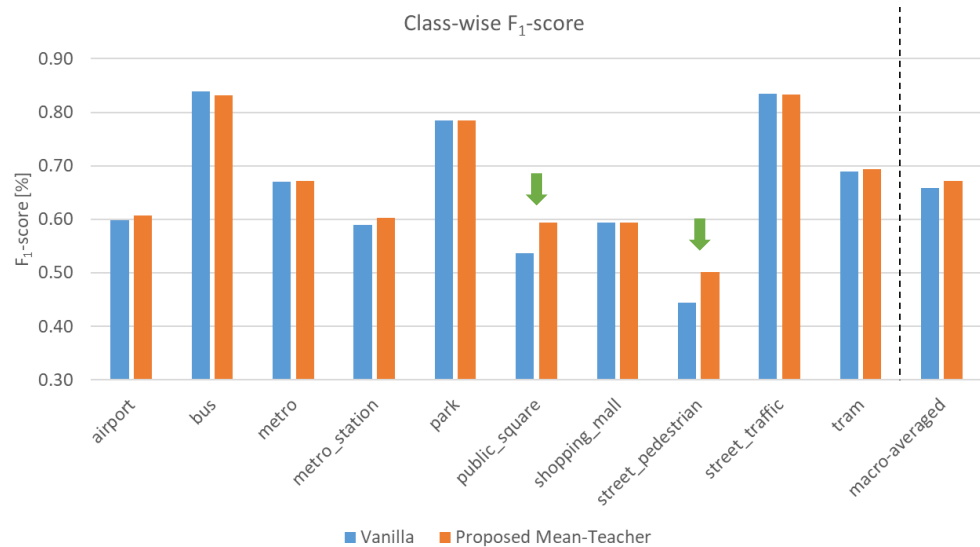


Figure 6. Classwise F₁-score performances. Green arrows indicate classes with significantly improved F₁-score performance.

In the training step, additional computation time is required to calculate the prediction result of the teacher model in the proposed mean-teacher structure. Of course, there is no additional time required for updating the coefficient, i.e., backpropagation, because the coefficients of the teacher model are fixed. In our experiment using a GPU (NVIDIA Geforce RTX 3090), the elapsed time to train the vanilla model and the proposed mean-teacher model are 105 and 146 s per epoch, respectively. With using a CPU, the elapsed time to train the vanilla and the proposed models are 55 and 76 minutes per epoch, respectively. Each elapsed time was measured as an average of ten epochs for the GPU and three epochs for the CPU. In the inference step, the proposed and the vanilla models have the same elapsed times because the proposed model uses only the student model.

4.3. Performance Comparison with Various Parameters

Table 4 shows the performances of the proposed model with various consistency weights. For $\beta = 0$, the performance is of course similar to that of the vanilla model. In our experiment, the performance is the best when $\beta = 2.5$. The performances of the proposed model in the cases of $\beta = 5$ and $\beta = 7.5$ are slightly lower than the best performance, but these cases have better performance than the case with $\beta = 0$, which is practically the same as that of the vanilla model.

Table 4. Performance comparison with various consistency weights, β .

Consistency Weight	Log Loss	Macro-Averaged F ₁ -Score [%]
$\beta = 0$	1.045	65.66%
$\beta = 2.5$	1.009	67.12%
$\beta = 5$	1.013	66.07%
$\beta = 7.5$	1.019	66.03%

As mentioned in Section 3.3, our proposed model has modified strategies for consistency learning: removing unlabeled data path, modification of the time-varying weight function $w(n)$, and modification of adding noise strategy. In this section, we present the results of additional experiments to validate and to the second and third modification. The first modification, removing unlabeled data path, is not verified because it is obvious. The effectiveness of the modification of $w(n)$ is shown in Table 5, and the effect of adding noise strategy is evaluated in Tables 6 and 7.

Table 5 shows the performances of the proposed model with various start epoch parameters n_{act_ems} . As mentioned earlier, the training of the vanilla model was saturated between 230 and 260 epochs, and the best performance was obtained when ensemble-guided learning was started at 260 epochs, which is immediately after saturation. Early application of the ensemble-guided learning, such as $0 \leq n_{act_ems} \leq 100$, seems to interfere in the early-stage of supervised learning due to the ensembles of immature student models. Moreover, the last row in Table 5 shows the performance of the proposed system when only $w(n)$ is replaced with an exponential ramp-up function for 260 epochs. Comparing the result of the exponential ramp-up $w(n)$ with the results of the unit step $150 \leq n_{act_ems} \leq 300$, the results of unit step weights are better than that of the exponential ramp-up weight function.

Table 5. Performance comparison with various start epoch parameters n_{act_ems} for ensemble guidance.

Start Epoch	Log Loss	Macro-Averaged F ₁ -Score
$n_{act_ems} = 0$	1.060	63.55%
$n_{act_ems} = 50$	1.048	64.32%
$n_{act_ems} = 100$	1.049	65.53%
$n_{act_ems} = 150$	1.040	66.04%
$n_{act_ems} = 200$	1.022	66.20%
$n_{act_ems} = 260$	1.009	67.12%
$n_{act_ems} = 300$	1.027	65.70%
Ramp-up for 260 epochs	1.045	65.82%

Table 6 shows the performances of the proposed model with various signal-to-noise ratios (SNRs) when adding Gaussian noise. As mentioned earlier, the noise was added to the student model inputs only in the activated period of the ensemble-guided model (i.e., when $n \geq n_{act_ems}$). In the 70-dB SNR case, the noise is too small, so the performance is very similar to that of the model trained without noise. The case with 60-dB SNR has the best performance among the various SNR settings. For the 50-dB SNR case, the performance is slightly worse than the best performance, probably due to the excessive noise, but it performs better than the no-noise case.

Table 6. Performance comparison with various signal-to-noise ratios (SNRs) of additive noise.

SNR	Log Loss	Macro-Averaged F ₁ -Score
Without noise	1.041	65.97%
70 dB	1.041	66.03%
60 dB	1.009	67.12%
50 dB	1.028	66.39%

Table 7 shows the performance comparison between noise addition strategies of the conventional and proposed methods. The conventional mean-teacher model does not recommend any specific noise addition strategy but generally uses a strategy of adding noise to each model's input throughout the training process for semi-supervised learning. In our experiment for supervised learning, the performance was better when adding noise only in the period of $n \geq n_{act_ems}$, instead of adding noise throughout the training process. As shown in Table 7, when noise is added throughout the training process to both models, the F₁-score is improved, but the log loss is slightly degraded. Both performance indices, the F₁-score and the log loss, improve when noise is applied only to the student model, and there is a greater improvement when the noise application interval is adjusted to $n \geq n_{act_ems}$.

Table 7. Performance comparison with various noise addition strategies.

Method	Log Loss	Macro-Averaged F ₁ -Score
Without noise	1.041	65.97%
Both models, throughout training	1.045	66.30%
Student model only, throughout training	1.025	66.92%
Student model only, $n \geq n_{act_ems}$	1.009	67.12%

4.4. Application to State-of-the-Art Model

In order to evaluate our ensemble-guided model, we applied it to Liu’s wide ResNet (WRN) with 1-bit quantization [21], which achieved fifth in team ranking with an extremely small model size in DCASE 2021 Task 1A. The model that we implemented with reference to the study in [21] is shown in Figure 7. All convolutional layers consisted of 1-bit quantized weights as follows except for the gray-colored layer in Figure 7 [21,34]:

$$W_{ob,i} = \sqrt{\frac{2}{K_i C_{in,i}}} \text{sgn}(W_i) \tag{6}$$

where W_i and $W_{ob,i}$ are the weight tensors before and after quantization, respectively; $\text{sgn}(\cdot)$ is the element-wise sign function; K_i is the kernel size; and $C_{in,i}$ is the number of input channels of i th convolutional layer. If the convolutional layer has a two dimensional kernel, e.g., $(a \times b)$, the kernel size is calculated by multiplication of each dimension, e.g., ab . The training parameters and mean-teacher model parameters were set to the same values as Section 4.1 including learning rate scheduling.

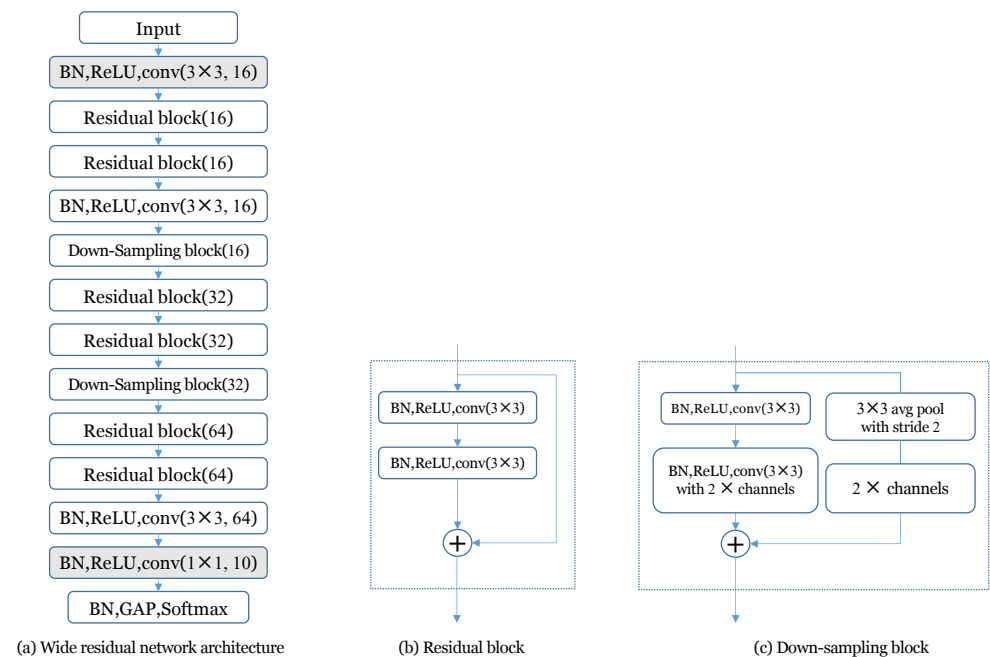


Figure 7. Schematic diagrams of (a) the ASC model with wide ResNet (WRN), (b) the residual block, and (c) the down-sampling block. The “2 × channels” layer in the down-sampling block is a layer for doubling the number of channels with zero paddings for the skip connection.

Table 8 shows the performances of the ensemble-guided model with 1-bit-quantized WRN. Both the log loss and F₁-score performances are improved for the cases of the proposed mean-teacher model with $\beta = 0.5, 1, 2.5$ compared with the vanilla model, and the case with $\beta = 2.5$ shows the best performance. The proposed model with $\beta = 5$ has the same log loss performance as the vanilla model, but it shows improved F₁-score

performance. Therefore, the proposed mean-teacher model is also effective for the 1-bit quantized WRN model structure in our experiment.

Table 8. Performances of the 1-bit-quantized WRN models.

Model	Log Loss	Macro-Averaged F ₁ -Score	Model Size
Vanilla	1.035	65.44%	43.6 KB
Proposed Mean-Teacher with $\beta = 0.5$	1.027	66.15%	43.6 KB
Proposed Mean-Teacher with $\beta = 1$	1.012	67.05%	43.6 KB
Proposed Mean-Teacher with $\beta = 2.5$	1.004	67.49%	43.6 KB
Proposed Mean-Teacher with $\beta = 5$	1.035	66.55%	43.6 KB

We expect that the proposed technique will be well applied to the models of the top four teams of DCASE 2021 Task 1A, because the models also consist of CNN variants, and various learning strategies or data augmentation techniques do not conflict with the proposed algorithm. It is expected that the proposed method can be applied even when pruning is used to reduce the model size, but additional verification is required in this case. However, some models of Kim's [35] and Heo's [33] systems utilize the knowledge distillation technique, which uses teacher–student structure, so it may be necessary to design a different strategy to apply the proposed technique in this case.

5. Conclusions

In this paper, we have researched to find the performance enhancement method for model-complexity-limited ASC tasks. We first paid attention to the model ensemble techniques as the method for performance enhancement. Ensembles of several models are a well-known auxiliary method for performance enhancement, but they require a large model size. Therefore, the model ensemble technique is hard to use for the model-complexity-limited ASC problem. In order to tackle this problem, we modify the mean-teacher model, which is developed for consistency learning in the semi-supervised learning, and apply it to our problem.

In order to evaluate our model, ASC experiments were performed with a database from DCASE 2021 Task 1A. First, our mean-teacher model was applied to a ResNet-based low-model-complexity ASC model to enhance the classification performance. As the evaluation result, our proposed method was able to improve the log loss performance up to 1.009 and F₁-score performance by 67.12%, whereas the vanilla model showed a log loss performance of 1.052 and an F₁-score performance of 65.79%. Moreover, our proposed model was more effective than the simple application of the temporal ensembling and mean-teacher model techniques, which are developed for the semi-supervised learning. Finally, our model was applied to another model, namely the 1-bit quantized WRN model. The model with the proposed technique was able to improve the log loss performance up to 1.004 and the F₁-score performance by 67.49%, whereas the vanilla model showed a log loss of 1.035 and an F₁-score of 65.44%.

Author Contributions: Conceptualization, S.L., S.P. and Y.J.; methodology, S.L., validation, S.L., M.K., S.S. and S.B.; data curation, M.K., S.S. and S.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00050, Development of Human Enhancement Technology for auditory and muscle support).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: <http://dcase.community/challenge2021/task-acoustic-scene-classification> (accessed on 15 December 2021).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ASC	Acoustic scene classification
CNN	Convolutional neural network
CP	Canonical polyadic
DCASE	Detection and classification of acoustic scenes and events
SED	Sound event detection
WRN	Wide ResNet

References

1. Chu, S.; Narayanan, S.; Kuo, C.C.J.; Mataric, M.J. Where am I? Scene recognition for mobile robots using audio features. In Proceedings of the 2006 IEEE International Conference on Multimedia and Expo, Toronto, ON, Canada, 9–12 July 2006; IEEE: Piscataway, NJ, USA, 2006; pp. 885–888.
2. Ellis, D.P.; Lee, K. Minimal-impact audio-based personal archives. In Proceedings of the the 1st ACM workshop on Continuous Archival and Retrieval of Personal Experiences, New York, NY, USA, 15 October 2004; pp. 39–47.
3. Barchiesi, D.; Giannoulis, D.; Stowell, D.; Plumbley, M.D. Acoustic scene classification: Classifying environments from the sounds they produce. *IEEE Signal Process. Mag.* **2015**, *32*, 16–34.
4. Mesaros, A.; Heittola, T.; Virtanen, T. A multi-device dataset for urban acoustic scene classification. *arXiv* **2018**, arXiv:1807.09840.
5. Koizumi, Y.; Saito, S.; Uematsu, H.; Harada, N.; Imoto, K. ToyADMOS: A dataset of miniature-machine operating sounds for anomalous sound detection. In Proceedings of the 2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, USA, 20–23 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 313–317.
6. Lee, S.; Pang, H.S. Feature extraction based on the non-negative matrix factorization of convolutional neural networks for monitoring domestic activity with acoustic signals. *IEEE Access* **2020**, *8*, 122384–122395.
7. Politis, A.; Adavanne, S.; Krause, D.; Deleforge, A.; Srivastava, P.; Virtanen, T. A Dataset of Dynamic Reverberant Sound Scenes with Directional Interferers for Sound Event Localization and Detection. *arXiv* **2021**, arXiv:2106.06999.
8. Amiriparian, S.; Gerczuk, M.; Ottl, S.; Cummins, N.; Freitag, M.; Pugachevskiy, S.; Baird, A.; Schuller, B. Snore sound classification using image-based deep spectrum features. In Proceedings of the Interspeech, Stockholm, Sweden, 20–24 August 2017; pp. 3512–3516.
9. Połap, D.; Woźniak, M.; Damaševičius, R.; Maskeliūnas, R. Bio-inspired voice evaluation mechanism. *Appl. Soft Comput.* **2019**, *80*, 342–357.
10. Sawhney, N.; Maes, P. Situational awareness from environmental sounds. *Proj. Rep. Pattie Maes* **1997**, 1–7.
11. Clarkson, B.; Sawhney, N.; Pentland, A. Auditory context awareness via wearable computing. *Energy* **1998**, *400*, 20.
12. Suh, S.; Park, S.; Jeong, Y.; Lee, T. Designing Acoustic Scene Classification Models with CNN Variants. Technical Report. In Proceedings of the DCASE2020 Challenge, online, 1 July 2020.
13. Yang, C.H.H.; Hu, H.; Siniscalchi, S.M.; Wang, Q.; Yuyang, W.; Xia, X.; Zhao, Y.; Wu, Y.; Wang, Y.; Du, J.; Lee, C.H. A Lottery Ticket Hypothesis Framework for Low-Complexity Device-Robust Neural Acoustic Scene Classification. Technical Report. In Proceedings of the DCASE2021 Challenge, online, 1 July 2021.
14. Koutini, K.; Jan, S.; Widmer, G. Cpjku Submission to Dcase21: Cross-Device Audio Scene Classification with Wide Sparse Frequency-Damped CNNs. Technical Report. In Proceedings of the DCASE2021 Challenge, online, 1 July 2021.
15. 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.
16. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
17. Hu, H.; Yang, C.H.H.; Xia, X.; Bai, X.; Tang, X.; Wang, Y.; Niu, S.; Chai, L.; Li, J.; Zhu, H.; et al. Device-Robust Acoustic Scene Classification Based on Two-Stage Categorization and Data Augmentation. Technical Report. In Proceedings of the DCASE2020 Challenge, online, 1 July 2020.
18. Gao, W.; McDonnell, M. Acoustic Scene Classification Using Deep Residual Networks with Focal Loss and Mild Domain Adaptation. Technical Report. In Proceedings of the DCASE2020 Challenge, online, 1 July 2020.

19. Turpault, N.; Serizel, R.; Parag Shah, A.; Salamon, J. Sound event detection in domestic environments with weakly labeled data and soundscape synthesis. In Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events, New York, NY, USA, 25–26 October 2019.
20. Martín-Morató, I.; Heittola, T.; Mesáros, A.; Virtanen, T. Low-complexity acoustic scene classification for multi-device audio: Analysis of DCASE 2021 Challenge systems. *arXiv* **2021**, arXiv:2105.13734.
21. Liu, Y.; Liang, J.; Zhao, L.; Liu, J.; Zhao, K.; Liu, W.; Zhang, L.; Xu, T.; Shi, C. DCASE 2021 Task 1 Subtask A: Low-Complexity Acoustic Scene Classification. Technical Report. In Proceedings of the DCASE2021 Challenge, online, 1 July 2021.
22. Koutini, K.; Henkel, F.; Eghbal-zadeh, H.; Widmer, G. CP-JKU Submissions to DCASE'20: Low-Complexity Cross-Device Acoustic Scene Classification with RF-Regularized CNNs. Technical Report. In Proceedings of the DCASE2020 Challenge, online, 1 July 2020.
23. Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; Darrell, T. Rethinking the value of network pruning. *arXiv* **2018**, arXiv:1810.05270.
24. Laine, S.; Aila, T. Temporal ensembling for semi-supervised learning. *arXiv* **2016**, arXiv:1610.02242.
25. Tarvainen, A.; Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv* **2017**, arXiv:1703.01780.
26. Stowell, D.; Giannoulis, D.; Benetos, E.; Lagrange, M.; Plumbley, M.D. Detection and classification of acoustic scenes and events. *IEEE Trans. Multimed.* **2015**, *17*, 1733–1746.
27. Heittola, T.; Mesáros, A.; Virtanen, T. Acoustic scene classification in dcase 2020 challenge: generalization across devices and low complexity solutions. *arXiv* **2020**, arXiv:2005.14623.
28. Kim, B.; Chang, S.; Lee, J.; Sung, D. Broadcasted Residual Learning for Efficient Keyword Spotting. *arXiv* **2021**, arXiv:2106.04140.
29. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
30. Wu, J.; Wang, Y.; Wu, Z.; Wang, Z.; Veeraraghavan, A.; Lin, Y. Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm Sweden, 10–15 July 2018; pp. 5363–5372.
31. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.
32. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
33. Heo, H.S.; weon Jung, J.; jin Shim, H.; Lee, B.J. Clova submission for the DCASE 2021 challenge: Acoustic scene classification using light architectures and device augmentation. Technical Report. In Proceedings of the DCASE2021 Challenge, online, 1 July 2021.
34. McDonnell, M.D. Training wide residual networks for deployment using a single bit for each weight. *arXiv* **2018**, arXiv:1802.08530.
35. Kim, B.; Yang, S.; Kim, J.; Chang, S. QTI submission to DCASE 2021: Residual normalization for device imbalanced acoustic scene classification with efficient design. Technical Report. In Proceedings of the DCASE2021 Challenge, online, 1 July 2021.