IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 14, 2022, accepted May 12, 2022, date of publication May 18, 2022, date of current version May 31, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3176003

NAS-TasNet: Neural Architecture Search for Time-Domain Speech Separation

JOO-HYUN LEE[1], JOON-HYUK CHANG[1], (Senior Member, IEEE), JAE-MO YANG[2], AND HAN-GIL MOON[2]

[1]Department of Electronic Engineering, Hanyang University, Seoul 04763, South Korea
[2]Samsung Electronics, Suwon-si, Gyeonggi-do 16677, South Korea

Corresponding author: Joon-Hyuk Chang (jchang@hanyang.ac.kr)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean Government [Ministry of Science and ICT (MSIT)], development of ultra-high speech quality technology for remote multi-speaker conference system, under Grant 2021-0-00456.

ABSTRACT The fully convolutional time-domain speech separation network (Conv-TasNet) has been used as a backbone model in various studies because of its structural excellence. To maximize the performance and efficiency of Conv-TasNet, we attempt to apply a neural architecture search (NAS). NAS is a branch of automated machine learning that automatically searches for an optimal model structure while minimizing human intervention. In this study, we introduce a candidate operation to define the search space of NAS for Conv-TasNet. In addition, we introduce a low computational cost NAS to overcome the limitations of the backbone model that consumes large GPU memory for training. Next, we determine the optimized separation module structures using two search strategies based on gradient descent and reinforcement learning. In addition, when NAS is simply applied, there is an imbalance in the updating of architecture parameters, which are NAS parameters. Therefore, we introduce an auxiliary loss method that is appropriate for the Conv-TasNet architecture for a balanced architecture parameter update of the entire model. Furthermore, we determine that the auxiliary loss technique mitigates the imbalance of architecture parameter updates and improves the separation accuracy.

INDEX TERMS Automated machine learning (AutoML), convolutional neural network (CNN), deep learning, end-to-end, speech processing, speech separation, neural architecture search, time-domain speech separation.

I. INTRODUCTION

Speech communication in the real world frequently occurs in crowded multispeaker settings. Speech processing systems should be able to differentiate speech from distinct speakers in such situations. Speech separation aims to separate monaural speech recordings into their constituent sounds. Automatic speech separation allows machines to listen selectively to distinct sounds in an acoustic mixture. Selective machine listening enables robust sound processing in real-world acoustic environments. Applications such as speech communication and automatic speech recognition often require automated source separation for robustness.

Most previous speech separation approaches operate on the time-frequency domain separation that uses a spectrogram of the mixture signal, estimated using the short-time Fourier transform (STFT) [1]. Former methods that use spectrograms as inputs estimate the spectrogram representation of each source and use clean-source spectrograms as the target [2], [3]. The mask estimation approach is an alternative method that separates a mixture signal from the spectrogram of each source by estimating the weighting function (mask) and multiplying it by the mixture representation. In recent years, deep learning has achieved significant improvements in the performance of masking methods [4]–[12]. However, the spectrogram method has two major issues. First, the reconstruction of the clean source phase is also a nontrivial problem. The error in estimating the phase results in an upper bound of the separation performance. Second, successful separation from the spectrogram requires high-resolution frequency decomposition, and a long temporal window is necessary for the STFT

The associate editor coordinating the review of this manuscript and approving it for publication was Joanna Kołodziej.

of the mixture signal for better separation performance. High-resolution STFT limits its applicability in real-time, resulting in low-latency applications. Thus, direct separation in the time domain [13]–[16] has been attempted to overcome the disadvantages of time-frequency domain separation. Among several time-domain separation techniques, the fully convolutional time-domain speech separation network (Conv-TasNet) [17] has received interest from many researchers because of its remarkable performance improvement compared to the previous methods.

Conv-TasNet is a time-domain mask estimation speech separation model composed of a temporal convolutional network (TCN) [18]–[20]. Stacked dilated 1-D convolutional neural network (CNN) blocks can cover a long receptive field, showing better performance than the previous method. In addition to its separation performance, convolution allows parallel processing and accelerates the separation. Models using recurrent neural networks [21], U-Nets [22], and transformers [23], [24] have been proposed. As various models have been introduced, Conv-TasNet appears to be outdated. However, considering the trade-off between various factors such as model size and latency owning to computational complexity, it is still a superior model compared to latest models. Owning to its excellent structure, Conv-TasNet is used for speech and the sound source separation of other domains using some modifications [25]–[27] and continues to be used as a reference model for new separation model training methods [28]–[30].

Deep learning has significantly impacted almost all fields, including the speech separation domain, and recent deep learning models have outperformed existing methods. However, as deep learning models mature, the complexity of the model is getting incorporated. Thus, considerable field knowledge and time are often required to design models with excellent performances. Consequently, the importance of the neural architecture search (NAS), which automates deep learning architectural engineering, has become more prominent. NAS can be a sub-field of automated machine learning (AutoML), similar to hyperparameter optimization and meta-learning. Models found by NAS with efficient configurations with fewer parameters have outperformed manually designed architectures in tasks such as image classification and language modeling [31]–[43]. Manually designed models generally use common model hyperparameters. For example, in a general model composed of CNN layers, all layers share the same kernel size or the model is constructed by increasing the number of kernels per layer at a specific rate. In contrast, the optimized model with an intricately configured combination of model hyperparameters, searched through NAS, shows performance and efficiency that exceed manually designed models. Finding complex combinations manually, as designed by NAS, requires enormous time and computing resources. Therefore, NAS is essential for optimizing the model structure to the extreme. Conv-TasNet also comprises various model hyperparameters. A previous study [17] attempted to use different model hyperparameters to derive optimal performance. The experiment in [17] demonstrated that the performance difference depended on the hyperparameter combination. However, attempts have not been made to find the optimal combination by varying the hyperparameters for each layer (1-D convolutional block). As architecture search has shown remarkable results in designing models, we can expect Conv-TasNet to have better performance and efficiency if the optimal combination of hyperparameters is found for each layer using the NAS.

As NAS has recently gained efficiency and become popular, research on NAS, which was active only in the existing image classification or language modeling fields, is being conducted in various fields. Recently, NAS has proven its excellence in the medical fields [44]–[47] and image segmentation [48]–[50]. However, the application of NAS in the audio deep learning field has scarcely been studied, except for a few cases applied to automatic speech recognition tasks [51], [52]. Therefore, we intended to prove the value of NAS research in the audio deep learning field by validating its effectiveness in speech separation tasks.

In this study, we introduce the method of applying NAS to Conv-TasNet and the results of the application. To apply NAS to Conv-TasNet, the candidate operations that define the NAS search space are determined. Additionally, efficient low computational cost NAS will be applied as the backbone model consumes a high GPU memory for training. Moreover, an imbalanced update of the architecture parameters, which are the NAS parameters throughout the model, is observed if NAS is simply applied. To address this issue, we exploit the auxiliary loss in the Conv-TasNet structure, which is a method used to improve the ability to capture long-term dependencies [53] or train deep models [54]. In addition, the application of auxiliary loss in Conv-TasNet model training significantly improved the separation performance. Finally, we induce models through gradient-based or reinforcement-learning-based NAS, outperforming the reference network. Similar to the model proposed in a previous study, NAS-Unet [44], the derived model is denoted as NAS-TasNet. The contributions of this study are as follows.

1) To the best of our knowledge, this study is the first approach to apply neural architecture search to the speech separation model.
2) We define an appropriate search space for Conv-TasNet NAS and propose a NAS application method that overcomes the high GPU memory requirement to train the model.
3) A loss computation method suitable to Conv-TasNet is devised to improve the separation model's neural architecture search and training.
4) We demonstrate that the performance of our searched model, NAS-TasNet, has fewer parameters and outperforms Conv-TasNet and its improved version (TDCN++) on the WSJ0-2mix dataset and its performance is similar to the existing performance on the WSJ0-3mix dataset.

II. RELATED WORKS

As deep learning models for different tasks develop and model designs become complex, designing networks requires considerable effort from experts. Therefore, some industry experts and scholars have focused on algorithmic solutions to automate the manual architectural design process. NAS is a field that automates the design of deep learning models. NAS has proven its capabilities by automatically discovering image classification CNN and recurrent neural network (RNN) designs in the language field to discover model designs that are superior to existing models.

NAS methods can be categorized into search spaces, search strategies, and performance-estimation strategies [55]. The search space is defined to determine the candidate elements of the model to be optimized. Incorporation of prior knowledge of the typical properties of architectures adequate for a task to determine the candidate element, can simplify the search by reducing the size of the search space. However, such reflection on prior knowledge and human bias makes it impossible for NAS to determine a new model beyond human knowledge. The determination of a search strategy incorporates a similar exploration-exploitation trade-off problem in determining the search space. The best strategy is an algorithmic solution that quickly searches for an optimal model, while avoiding rapid convergence to a region of suboptimal architectures. Search algorithms mainly include evolutionary algorithms [31]–[33], reinforcement learning [34]–[37], [43], and gradient-based methods [39]–[41], [43]. The performance estimation strategy measures the performance of a derived model on unseen data during the NAS process. The most straightforward method is to build a sampled model, run standard training from scratch until the performance converges, and evaluate the architecture on a test split. However, this naïve method is computationally expensive, limiting the search for various architectures. Consequently, recent studies have focused on developing strategies to lower the cost of these performance assessments. For example, some studies [38]–[41], [43] have proposed a method that represents an architecture as a directed acyclic graph (DAG) and construct an over-parameterized network with all candidate operations. Using this approach, different architectures can be constructed by simply changing the edges and nodes of an over-parameterized network. Thus, multiple model performance can be measured without rebuilding and re-training.

Some attempts have been made the NAS process faster and more resource-efficient. GPU-days is a measure of NAS efficiency, defined as GPU-days $= N \times D$, where N represents the number of GPUs and D represents the actual number of days spent searching. The conventional NAS algorithm required 22,400 GPU-days (800 GPUs $\times$ 28 days). Consequently, operating NAS is difficult for individuals or small research groups. However, the efficiency of NAS significantly improved after the over-parameterized network DAG approach was introduced. For example, the efficient neural architecture search (ENAS) [38] outperformed existing methods with only 0.45 GPU-days by utilizing the over-parameterized network DAG concept, even though it employed the same reinforcement-learning-based search strategy as the conventional NAS. By taking advantage of the characteristics of the over-parameterized network concept, differentiable architecture search (DARTS) [39], assigning weights to candidate computation edges of over-parameterized networks, and optimizing the weights through gradient descent, have been proposed. DARTS is a relatively intuitive method that searches for a model with good performance within a few GPU-days. Subsequently, a more efficient DARTS method that reduces the GPU consumption [43] was proposed. The constraint that restricts execution for NAS with large input dimensions was resolved by reducing GPU memory consumption.

III. SEARCH SPACE CONFIGURATION FOR SEPARATION NETWORK

Single-channel speech separation can be defined as the estimation of N sources $\mathbf{s}_1, \ldots, \mathbf{s}_N \in \mathbb{R}^{1\times T}$ in a mixture waveform $\mathbf{x} \in \mathbb{R}^{1\times T}$:

$$\mathbf{x} = \sum_{i=1}^{N} \mathbf{s}_i. \tag{1}$$

Conv-TasNet [17] is composed of three instances: encoder, separator, and decoder. Figure 1 overviews the Conv-TasNet architecture. Encoder $\mathcal{E}$ transforms the input mixture signal $\mathbf{x}$ into a latent representation $\mathbf{v}_\mathbf{x} = \mathcal{E}(\mathbf{x}) \in \mathbb{R}^{C_\mathcal{E}\times L}$. Separator $\mathcal{S}$ estimates the corresponding masks $\widehat{\boldsymbol{m}}_i \in \mathbb{R}^{C_\mathcal{E}\times L}$ for each of the N sources $\mathbf{s}_1, \ldots, \mathbf{s}_N \in \mathbb{R}^T$ that constitute the mixture. The estimated latent representation for each source in latent space $\widehat{\mathbf{v}}_i$ is retrieved by multiplying the element-wise estimated mask $\widehat{\mathbf{m}}_i$ with the encoded mixture representation $\mathbf{v}_\mathbf{x}$. Decoder $\mathcal{D}$ estimates the target sources from latent space vectors $\widehat{\mathbf{s}}_i = \mathcal{D}(\widehat{\mathbf{v}}_i)$. The separator consists of stacked 1-D dilated convolutional blocks motivated by a TCN [18]–[20]. Each 1-D convolutional block consists of a convolution with an exponentially increasing dilation factor d. This method allows the model to efficiently have a large receptive field. Consequently, the dilated convolutional network has a sufficiently large temporal context window to handle the long-term dependencies of inputs. The separation model consists of X convolutional block stack repeats R. Moreover, we used an advanced version of Conv-TasNet's separation module (TDCN++) [25] as our backbone network, rather than the original Conv-TasNet. This improved architecture enabled three additional improvements over the original network. First, global layer normalization within 1-D convolutional blocks, which normalizes the overall features and frames, was replaced with feature-wise layer normalization over the frames. The second improvement was the longer-range skip-residual connections from earlier block-stack repeat inputs to later repeat inputs after passing them through dense layers. The third advancement was the learnable scaling parameter. The scaling parameter, which was applied immediately before the residual connection for

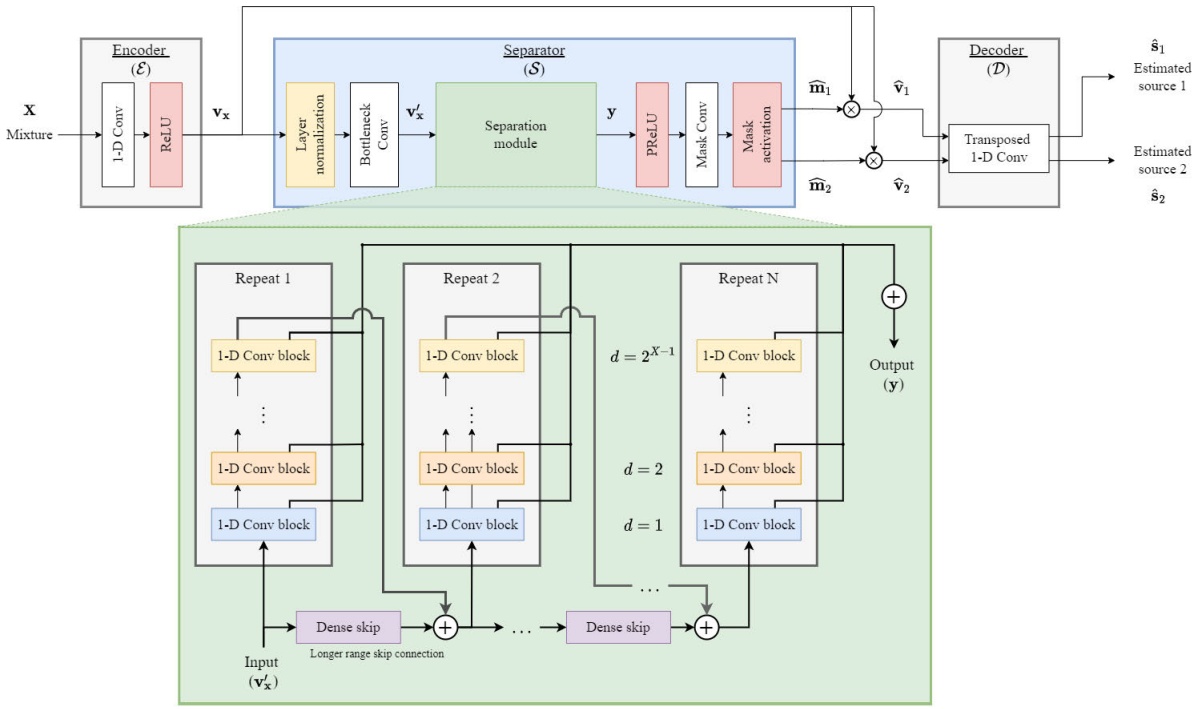


FIGURE 1. Conv-TasNet architecture for the separation of two sources.

the second 1-D convolutional block in each repeat, was initialized to an exponentially decaying scalar equal to 0.9^X, where X was the index of the block of a stack repeat.

Table 1 lists the hyperparameters of Conv-TasNet. Except for variables N and L, all hyperparameters are related to the separation module. The previous study [17] demonstrated that the performance varies with combinations of hyperparameter values through many attempts. The experiments in [17] attest that the autoencoder variables (N and L) are also essential performance factors. However, owing to the complexity, we intend to apply NAS only to 1-D convolutional block-based separation modules. In this study, we design the NAS to determine the optimal combination of H, P, X, and R of the separation model.

TABLE 1. Network hyperparameters.

Symbol	Description
N	Number of filters in the auto-encoder
L	Length of the filters (in samples)
B	Number of channels in the bottleneck, skip-connection and residual path in a 1-D convolutional block
H	Number of channels in the point-wise conv in a 1-D convolutional block
P	Kernel size of the depth-wise conv in a 1-D convolutional block
X	Number of 1-D convolutional blocks in each repeat
R	Number of repeats

Two types of search spaces exists: the network- and cell-based. The network-based method explores the entire network, whereas the cell-based method constructs a network by conducting NAS for cells and repeating the cell structures with a fixed number. This study employed a network-based approach; thus, each 1-D convolutional block was given individual model hyperparameters.

Our search space consist of the design choices for kernel size P and expansion ratio H. We allow a set of 1-D convolutional blocks with various kernel sizes {3, 5} and expansion ratios $\{\times 1, \times 2, \times 4\}$ (Figure 2(a)). The expansion ratio represents multiple kernels compared with the number of input channels. Moreover, we add a zero layer (Figure 2(b)) to the group of candidate operations to design the NAS for layers and repeat counts (X and R). The zero layer outputs zero tensors in the same dimension as the input. Because all 1-D convolutional blocks consist of residual and skip connections, the placement of a zero layer between blocks has the same effect as omitting one block. Adjustment of the layer count in a repeat may break the exponentially increasing dilation pattern proposed in [17]. Accordingly, the dilation of each 1-D convolutional block was adjusted by re-assigning the index of one repeat layer, except for the zero layer before a forward operation. A mixed operation constitutes of seven candidate operations: two cases for the kernel size, three cases for 1-D convolutional block channel numbers, and a zero layer.

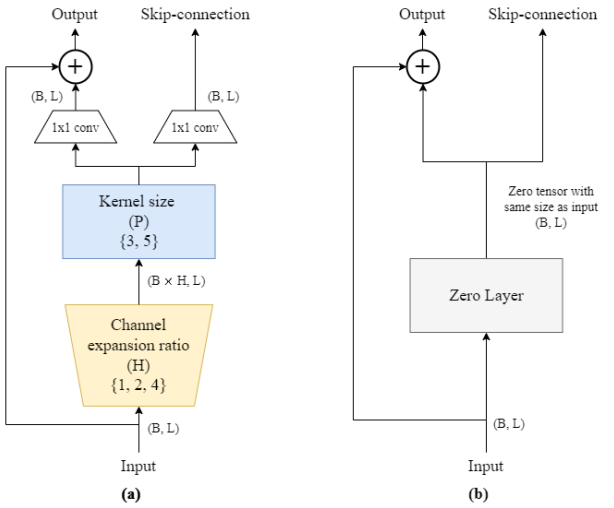


FIGURE 2. Candidate operations compose search space: (a) 1-D convolutional block with various kernel sizes and expansion ratios. (b) Zero layer that controls the depth of a repeat.

Note that B, $R_{\max}$, and $X_{\max}$ are the NAS hyperparameters, where B is the bottleneck size of separator $\mathcal{S}$, and we set it to 128, which is the best performance value in [17]. $R_{\max}$ represents the maximum number of repeats and $X_{\max}$ is the maximum number of 1-D convolutional blocks in each repeat. We fix the value of $X_{\max}$ to 8, which is the value of the best performance in [17]. In the case of $R_{\max}$, experiments are conducted for two cases: 3 and 4.

IV. ARCHITECTURE SEARCH STRATEGY

This section describes how we conduct an architecture search on the network-based search space determined in the previous section. First, we describe the implementation of DARTS [39] and the construction of a mixed operation network with all candidate paths. Next, we describe how we conduct the computationally cost-efficient gradient-based NAS (ProxylessNAS) proposed by Cai *et al.* [43]. Subsequently, we introduce an objective function for the architecture parameter update that considers the performance and model size during the architecture search. Finally, we describe a reinforcement learning-based search implementation as a REINFORCE-based search that can be performed on our mixed operation network.

A. DIFFERENTIABLE ARCHITECTURE SEARCH

DARTS proposes the relaxation of the discrete set of a candidate operations. DARTS relaxes the categorical choice of a particular operation to a softmax function over all possible operations. Because relaxation makes the search space continuous, architecture optimization through gradient descent is possible. To proceed with DARTS, we configure an over-parameterized network with mixed operations that contains all candidate operations and architecture parameters α. An over-parameterized network is also called a mixed operation network. We represent the architecture as a DAG, denoting a mixed operation network as $\mathcal{N}(e, \ldots, e_n)$, where e_i represents a certain edge in the DAG. To construct a mixed operation network that includes any architecture in the search space, we set each edge as a mixed operation, $m_{\mathcal{O}}$, with the set of candidate primitive operations denoted as $\mathcal{O} = \{o_i\}$ and the network expressed as $\mathcal{N}(e = m^1_{\mathcal{O}}, \ldots, e_n = m^n_{\mathcal{O}})$.

If a mixed operation and its input are defined as x and $m_{\mathcal{O}}$, respectively, in DARTS, $m_{\mathcal{O}}(x)$ is the weighted sum of $\{o_i(x)\}$, where the weights are calculated by applying softmax to N real-valued architecture parameters α_i, such that

$$m_{\mathcal{O}}(x) = \sum_{i=1}^{N} p_i o_i(x) = \sum_{i=1}^{N} \frac{\exp(\alpha_i)}{\sum_j \exp(\alpha_j)} o_i(x). \tag{2}$$

The architecture parameter α_i determines the probabilities of the corresponding candidate operations in a mixed operation via using softmax. With an over-parameterized network, DARTS alternately updates the model parameters (weight, w) and architecture parameters α to minimize the objective function of the architecture (bi-level optimization). As the training progresses, the α_i of an operation that improves the performance increases, and the rest decreases. After model training, we prune all operations, except candidate operations with the largest alpha from mixed operations and determine it as an optimized cell architecture. 3 illustrates the DARTS procedure.

B. ARCHITECTURE SEARCH WITH GPU MEMORY SAVING

As shown in Eq. (2), the output feature maps of all N paths are calculated and stored in memory, whereas the training of a compact model involves only one path. DARTS requires approximately N times the GPU memory and GPU hours compared with training a compact model. This computational approach leads to out-of-memory issues for large-scale input datasets and extends the pool of candidate operations. Because the input of the time-domain speech separation model is large raw audio data, regular DARTS cannot be conducted on the separation model owing to the memory overuse of DARTS's weighted sum computation. Instead of regular DARTS, we applied ProxylessNAS [43], a memory-efficient method, to proceed with DARTS, reducing the memory footprint by maintaining only one path in an over-parameterized mixed operation network through binarization. To binarize the paths, we transform N real-valued path weights $\{\alpha_i\}$ into binary gates as follows:

$$g = \text{binarize}(p_1, \ldots, p_N) = \begin{cases} [1, 0, \ldots, 0] \\ \quad \ldots \\ [0, 0, \ldots, 1] \end{cases}. \tag{3}$$

The output of a mixed operation that applied binary gates g, is given by:

$$m^{\text{Binary}}_{\mathcal{O}}(x) = \sum_{i=1}^{N} g_i o_i(x) = \begin{cases} o_1(x) \\ \quad \ldots \\ o_N(x) \end{cases}. \tag{4}$$

In a mixed operation with a binary gate, only one path of activation is active in the memory at runtime; thus, the

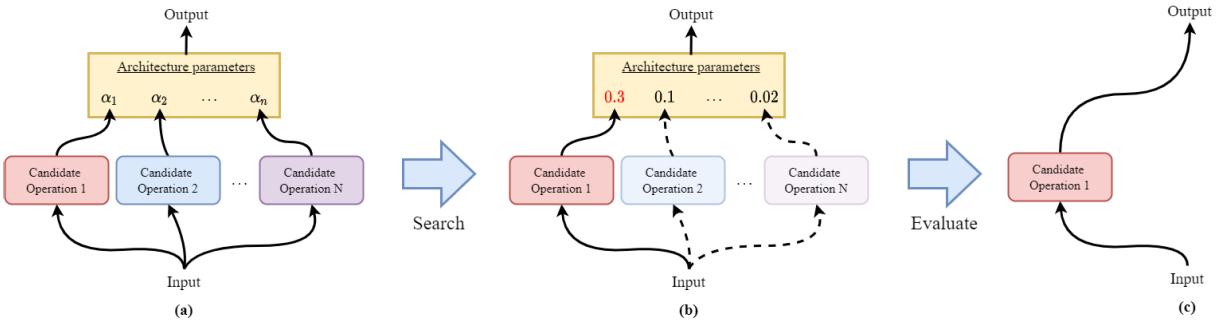


FIGURE 3. Overview of DARTS: (a) Continuous relaxation of the search space by placing a mixture of candidate operations on each edge. (b) Joint optimization of mixing probabilities (architecture parameters) and network weights by bi-level optimization. (c) Deriving the final compact model from the learned mixture probabilities.

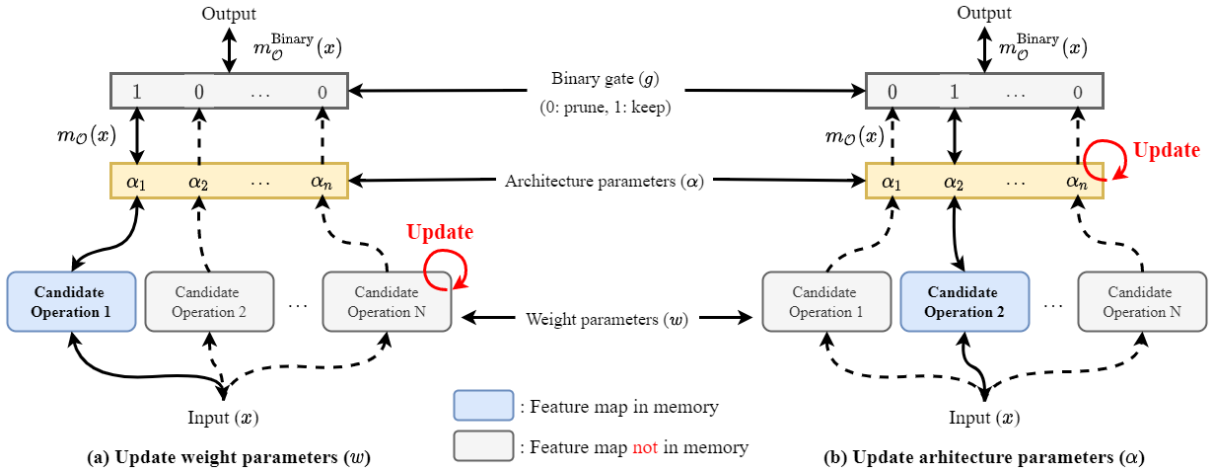


FIGURE 4. Overview of ProxylessNAS: (a) Continuous relaxation of the search space by placing a mixture of candidate operations on each edge. (b) Joint optimization of mixing probabilities and network weights by solving a bi-level optimization problem.

memory requirement to train the over-parameterized network is reduced to the same level of training a compact model.

Figure 4 shows the training procedure for the binarized architecture parameters in the mixed operation network. First, to train the network weight parameters, the architecture parameters (α) are frozen and binary gates are stochastically sampled for each batch of input data. Subsequently, the weight parameters of the active paths are updated via standard gradient descent on the training dataset (Figure 4(a)). To train the architecture parameters (α), the weight parameters (w) are frozen, the binary gates are reset, and the architecture parameters are updated on the validation split (Figure 4(b)). Unlike weight parameters w, the architecture parameters α are not directly involved in the mixed operation computation; therefore, they cannot be updated using regular gradient descent. Nevertheless, the binary gates (g) involved in the computation graph, $\partial m_{\mathcal{O}}^{\text{Binary}}(x)/\partial g$ can be calculated using backpropagation, as shown in Eq. (4). However, computing $\partial m_{\mathcal{O}}^{\text{Binary}}(x)/\partial g$ requires calculating and storing the output of mixed operations ($m_{\mathcal{O}}^{\text{Binary}}(x)$). Thus, updating the architecture parameters still require approximately N times the GPU memory compared with training a compact model. To solve this problem, the task of choosing one path out of N candidates is factorized into multiple binary selection operations, assuming that, if a path is the best choice in a particular position, it must have been a better choice than any other path. Accordingly, we first sample two paths based on the multinomial distribution $(p_1, \ldots, p_N)$ and mask all the other paths. Consequently, the number of candidates temporarily decreases from N to 2, meaning that the number of paths involved in the mixed operation and feature maps cast on the GPU is reduced from N to 2. Thus, the architecture parameters of these two sampled paths are updated. The update steps for these two weight and architecture parameters are alternatively performed. When the training of the architecture parameters is completed, a compact architecture is derived by pruning redundant paths.

C. MODEL SIZE-AWARE OBJECTIVE FUNCTION

The separation task aims to maximize the scale-invariant source-to-distortion ratio (SI-SDR) [56]. Thus, as an objective function, we use the negative permutation-invariant

SI-SDR [8]. The loss function $\mathcal{L}_{-\text{SI-SDR}}$ is defined between the target clean source $\mathbf{t}$ and estimates $\widehat{\mathbf{s}}$ as follows:

$$\mathcal{L}_{-\text{SI-SDR}} = -\text{SI-SDR}(\mathbf{t}^*, \widehat{\mathbf{s}}) = -10\log_{10}\left(\frac{\|\alpha\mathbf{t}^*\|^2}{\|\alpha\mathbf{t}^* - \widehat{\mathbf{s}}\|^2}\right), \tag{5}$$

where $\mathbf{t}^*$ denotes the permutation of the sources that maximizes the SI-SDR and $\alpha = \widehat{\mathbf{s}}^\top\mathbf{t}^*/\|\mathbf{t}\|^2$ is a scalar. In addition to SI-SDR, we attempt to optimize the model size with fewer parameters during the architecture search. As in [57], an efficient architecture can be designed by considering the number of floating-point operations (FLOPs). However, unlike negative SI-SDR, can be optimized with the gradient of the objective function, FLOPs are non-differentiable. To solve this problem, we use the architecture search object function that considers the FLOPs in which the loss function term proposed by ProxylessNAS was applied.

Consider a mixed operation with a candidate set $\{o_j\}$, where each o_j is associated with a path weight p_j that represents the probability of selecting o_j. Thus, the expected FLOPs of a mixed operation can be defined as

$$\mathbb{E}[\text{FLOPs}_i] = \sum_j p_j^i \times F(o_j^i), \tag{6}$$

where $\mathbb{E}[\text{FLOPs}_i]$ is the expected FLOPs of the i^{th} mixed operation, $F(\cdot)$ is the FLOPs prediction function, and $F(o_j^i)$ is the predicted latency of o_j^i. With Eq. (6), the estimated FLOPs can be differentiated with respect to the architecture parameter and the gradient of $\mathbb{E}[\text{FLOPs}_i]$ can be expressed as $\partial\mathbb{E}[\text{FLOPs}_i]/\partial p_j^i = F(o_j^i)$. The expected FLOPs for the entire network with a sequence of mixed operations can be expressed as the sum of the estimated FLOPs of the mixed operations:

$$\mathbb{E}[\text{FLOPs}] = \sum_i \mathbb{E}[\text{FLOPs}_i]. \tag{7}$$

Thus, the expected latency of the network is added to a negative SI-SDR loss function by multiplying by a scaling factor $\lambda_2(>0)$, that controls the trade-off between accuracy and latency. The final loss function is given as

$$\mathcal{L} = \mathcal{L}_{-\text{SI-SDR}} + \lambda_1 \|w\|_2^2 + \lambda_2\mathbb{E}[\text{FLOPs}], \tag{8}$$

where $\lambda_1 \|w\|_2^2$ is the weight decay term.

D. REINFORCE-BASED APPROACH

With the architecture search settings mentioned in the previous subsections, REINFORCE [58] can be applied to train the architecture parameters. The main objective of updating the binarized parameters is to determine the optimal binary gates that maximize a certain reward, denoted as $R(\cdot)$. We adopt the reward function presented in [59] by changing the accuracy term to the SI-SDR and latency to the FLOPs of a model. The reward function obtained by applying the existing reward function is as follows:

$$R(\mathcal{N}_g) = \text{SI-SDR}(\mathcal{N}_g) \times \left[\frac{\text{FLOPs}(\mathcal{N}_g)}{T_{\text{ref}}}\right]^w, \tag{9}$$

where $\mathcal{N}_g$ denotes a currently sampled network with sampled binary gates, T_{ref} is the reference target FLOPs of a network that functions as a hard constraint, and w is a weight factor that adjusts the trade-off between SI-SDR and FLOPs. Thus, the reward function maximizes SI-SDR under a hard constraint. In this study, We set T_{ref} as the estimated total FLOPs of Conv-TasNet and $w = -0.07$.

Based on REINFORCE, we accomplish the following updates for the binarized parameters in a mixed operation network:

$$\begin{aligned} J(\alpha) &= \mathbb{E}_{g\sim\alpha}[R(\mathcal{N}_g)] = \sum_i p_i R(\mathcal{N}(e = o_i)), \\ \nabla_\alpha J(\alpha) &= \sum_i R(\mathcal{N}(e = o_i))\nabla_\alpha p_i, \\ &= \sum_i R(\mathcal{N}(e = o_i)) p_i \nabla_\alpha \log(p_i), \\ &= \mathbb{E}_{g\sim\alpha}[R(\mathcal{N}_g)\nabla_\alpha \log(p(g))], \\ &\approx \frac{1}{M}\sum_{i=1}^{M} R(\mathcal{N}_{g^i})\nabla_\alpha \log(p(g^i)). \end{aligned} \tag{10}$$

where g^i denotes the i^{th} sampled binary gate, $p(g^i)$ is the probability of sampling (policy) according to Eq. (3), and $\mathcal{N}_{g^i}$ is a compact network based on the binary gates g_i. M is the hyperparameter of the REINFORCE-based NAS, the number of different architectures sampled in one mini-batch. In this study, M is set to 4, meaning that we sample 4 models for a mini-batch from the current policy and evaluate the estimated reward of the policy from the average reward of the four models. Because Eq. (10) does not require $\mathcal{N}_{g^i}$ to be differentiable with respect to g, it can handle non-differentiable objectives.

V. AUXILIARY LOSS

In this section, we introduce auxiliary loss, which alleviates the architecture parameter update imbalance between mixed operations during NAS and improves the separation performance of the model. As described in Section IV-A, the architecture parameters are assigned to each mixed operation when performing the gradient descent-based search. The output of a mixed operation is the weighted sum of the candidate operation outputs. The weights are the values obtained by applying the softmax function to the architecture parameter. As NAS progresses, the weights are modified by updating the architecture parameters, and the weight of the most appropriate operation for the corresponding position among the candidate operations increases. The weight of each mixed operation can be a confidence level for a specific candidate operation and as the weight increases, the entropy of the weight distribution decreases. The entropy is the level of randomness or uncertainty of a probability distribution, and the entropy of a discrete probability distribution is defined as

$$\text{H} = -\sum_{i=1}^{N} p_i \ln(p_i), \tag{11}$$

where H is the entropy of the probability distribution, and p_i denotes the probability of corresponding candidate operations in a mixed operation. p_i is the weight of the weighted sum of a mixed operation in DARTS. N denotes the number of candidate operations. We monitor the entropy reduction of each weight distribution of the mixed operation for each NAS epoch to observe the progress of NAS. Through entropy decrease monitoring, an architecture parameter update imbalance is observed. The entropies of the mixed operations located at the front barely decrease, while those located at the last position reduce rapidly and excessively. This phenomenon indicates that the architecture parameter updates differ depending on the location of the mixed operation. We presumed that as the separation module is given a large depth, the layers in the front cannot provide a discriminative feature because the gradients are ineffectively back-propagated through all layers. Consequently, in earlier mixed operations, because each candidate operation cannot provide a discriminating feature, determining the preference among the candidates is impossible; thus, candidates are randomly selected.

To alleviate the parameter update imbalance between mixed operations placed in different locations and prevent random decisions in mixed operations of specific locations, we apply auxiliary loss, which is the method used to train deep models [54] or improve their ability to capture long-term dependencies [53]. Auxiliary loss is obtained by estimating the mask for each repeat and aggregating the losses. Figure 5 illustrates the acquisition of the auxiliary loss designed for the Conv-TasNet architecture. We denote the result of the element-wise sum of the output skip-connections of the layers

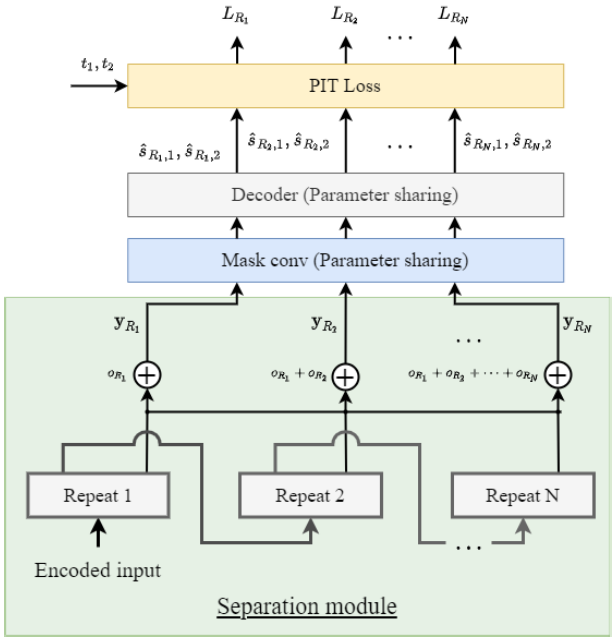


FIGURE 5. Schematic of auxiliary loss. Auxiliary loss is obtained by estimating the mask for each repeat and aggregating the losses.

of each repeat as y_{R_n} and the loss from each repeat as L_{R_n}. With L_{R_n}, we compute as much losses as the number of repeats by reusing the existing mask network and decoder, without creating additional model. The losses, L_{R_n}, are aggregated with the exponentially weighted moving average. We assume that the mask estimation from the earlier 1-D convolutional block repeats have relatively higher losses than the losses of the later repeats. Thus, we use the weighted average to control the influence of the front repeats instead of regular averaging. The moving average is employed instead of simple averaging to maintain the weighting pattern, even if the number of repeats is adjusted. The auxiliary loss L_{aux} is expressed as follows:

$$L_{\text{aux}} = L_{R_t} = \begin{cases} L_{R_t}, & t = 1 \\ \alpha L_{R_t} + (1-\alpha) L_{R_{t-1}} & t > 1 \end{cases}, \tag{12}$$

where coefficient α is a smoothing constant between 0 and 1, indicating the degree of weight reduction; if α is large, losses from earlier repeat decay are faster. The best performance is observed when α is 0.4. Auxiliary loss is introduced only in model training, and during inference, it estimates a mask only at the last repeat.

VI. EXPERIMENT AND RESULTS

A. DATASET

We evaluated our system on a two-speaker speech separation task using the widely used WSJ0-2mix and WSJ0-3mix datasets [5]. The datasets consisted of 30 h of training, 10 h of validation, and 5 h of evaluation splits generated from the Wall Street Journal (WSJ0) `si_tr_s`, `si_dt_05`, and `si_et_05` sets. Speech mixtures were generated by randomly mixing speech utterances from two and three active speakers at random signal-to-noise ratios (SNRs) between -5 dB and 5 dB. All waveforms were resampled at 8 kHz.

B. EXPERIMENTAL DETAILS

As mentioned, the Conv-TasNet structure consists of an autoencoder (encoder and decoder) and a separator. Because we applied NAS only to the separator, we determined several model configurations that needed to be fixed, including the settings of the autoencoder. The hyperparameters of the model were set by referring to the Conv-TasNet hyperparameters with the best performance reported in [17]. The encoder consisted of a 1-D CNN layer with an activation function. For 1-D CNN encoder configurations, we set the kernel size to $L = 16$, and converted this length to seconds, giving 2 ms ($\frac{L}{fs} = \frac{16}{8000} = 0.002s$); the stride size was 50% of the kernel size. Thus, the stride size was 8 ($\frac{L}{2} = 8$). The number of filters in the encoder was 512. The nonlinear activation function for the encoder was a rectified linear unit (ReLU). The decoder was a 1-D transposed CNN and the kernel and stride sizes of the decoder were set to be identical to those of the encoder. The separator comprised a bottleneck CNN, separation module, and mask CNN. The bottleneck CNN was a point-wise CNN with B channels and we set B as 128 in this study. The mask CNN was composed of a 1×1 CNN

TABLE 2. Comparison with backbone networks on the WSJ0-2mix and WSJ0-3mix datasets.

Model	# repeats (R)	Model size	WSJ0-2mix		WSJ0-3mix	
			SI-SDRi (dB)	SDRi (dB)	SI-SDRi (dB)	SDRi (dB)
Conv-TasNet* [17]	3	5.1M	16.0 (15.3*)	16.3 (15.6*)	12.3 (12.7*)	12.6 (13.1*)
	4	6.7M	16.3	16.6	12.7	13.1
TDCN++ [25]	4	6.7M	16.9	17.2	**14.2**	**14.5**
NAS-TasNet (GD)	3	4.5M	16.7	17.0	13.3	13.7
	4	6.3M	**17.7**	**18.0**	14.1	14.5
NAS-TasNet (RL)	3	**4.4M**	17.0	17.3	13.5	13.8
	4	5.0M	17.5	17.7	14.0	14.3

*: Method with reported results. The maximum SI-SDRi and SDRi performances reported in the corresponding paper were assigned.

and mask activation function. For the mask CNN, we set the channel counts to be the same as $B = 128$ and used the ReLU for the mask activation function. For the separation module, we defined two hyperparameters: maximum number of blocks in each repeat ($X_{\max}$) and maximum number of repeats ($R_{\max}$). We set $X_{\max}$ as 8, that is, we searched for the appropriate number of layers among a maximum of 8 layers for each repeat. For $R_{\max}$, we conducted experiments with $R_{\max} = 3$ and $R_{\max} = 4$.

The weight parameters were updated using the Adam optimizer throughout the process. The initial learning rate was set to 0.001 and the learning rate was halved if the validation score did not improve in 3 consecutive epochs. In addition, gradient clipping with a maximum L_2 norm of 5 was applied. NAS was performed in three stages: warm-up, search, and evaluation. In the warm-up phase, we froze the architecture parameters to avoid updating them. Subsequently, we randomly sampled a model in each step and trained the weight parameters of the sampled candidate operations. This phase aimed to converge the weight parameters of the candidate operations to render the performance between the candidate operations more discriminative. Thirty training epochs were performed during the warm-up phase. In the search phase, we alternatively trained the weight and architecture parameters. The Adam optimizer was used to train the architecture parameters, with a learning rate of 0.006 for the gradient-based algorithm and 0.01 for the REINFORCE-based algorithm. The number of search epochs was 40, which is sufficient for the model to converge. Finally, in the evaluation phase, the model induced from the search phase was trained. The auxiliary loss method was used to train the network during the evaluation stage. Early stopping was used, and the training was terminated if performance improvement did not improve for 7 consecutive epochs. Additionally, to test the generalization performance of the retrieved model for various tasks, the model induced from the architecture search from WSJ0-2mix was trained on WSJ0-3mix and evaluated. During the entire procedure, we used a mini-batch size of 8.

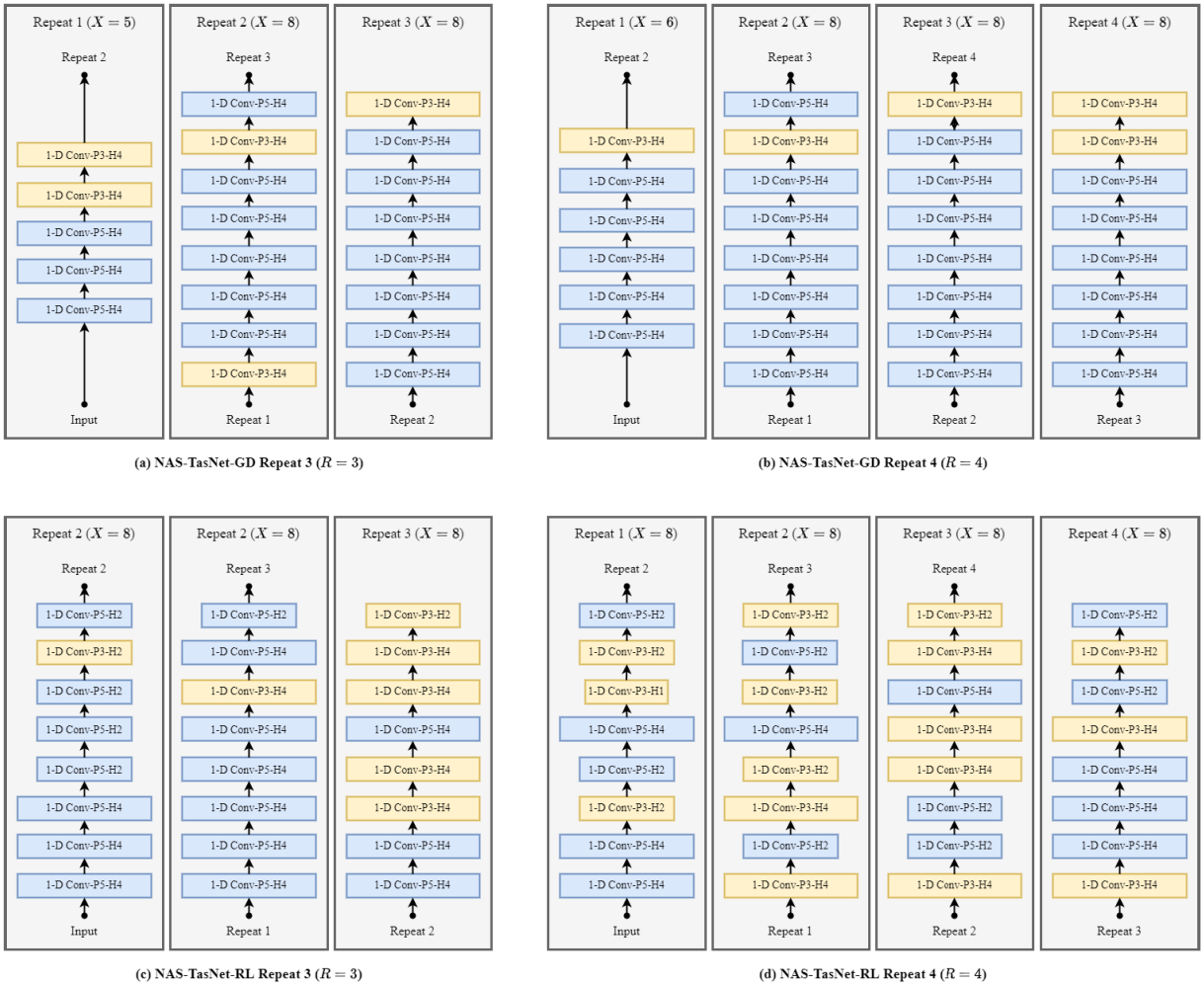
C. EXPERIMENTAL RESULTS OF ARCHITECTURE SEARCH

We report the degree of improvement in the signal fidelity measured by the improvements in the signal-to-distortion ratio (SDRi) [60] and SI-SNR (SI-SNRi), as follows:

$$\text{SI-SDRi}(s_i, \hat{s}_i, \text{x}) = \text{SI-SDR}(s_i, \hat{s}_i) - \text{SI-SDR}(s_i, \text{x}). \quad (13)$$

Eq. (5) defines SI-SDR, where SI-SDRi indicates the SI-SDR gain over the original mixture. Table 2 shows the results comparing the searched model with the proposed system and backbone network. First, we implemented the models presented in [17], [25] to generate a comparison to verify the performance improvement of the proposed model. We trained the reconstructed model from scratch and conducted tests to verify its performance. For Conv-TasNet, the test result was higher than the performance reported in [17]; however, this was an expected result, as we observed a similar result here [61]. By comparing Conv-TasNet with $R = 3$ and $R = 4$, we determined that performance improved as the number of repeats increased. For the TDCN++ configuration [25], because definitive hyperparameters were not provided, we set all hyperparameters, except the number of repeats (R), to the same value as in Conv-TasNet. Because TDCN++ is an improved form of Conv-TasNet, it showed better performance, as expected. NAS-TasNets refer to models searched through NAS, and NAS-TasNet-GD and NAS-TasNet-RL resulting from NAS, are based on gradient descent and reinforcement learning-based algorithms, respectively. Figure 6 shows the final compact separation modules derived from NAS. In the case of NAS-TasNet-GD, the model size was adjusted using zero layers, whereas for NAS-TasNet-RL, no zero layer was used; however, the model size was restrained by controlling the channel expansion ratios of each layer. In addition, candidate operations with a higher number of parameters were preferred as the front layers for each repeat.

We inferred that NAS effectively determined an excellent performance model with few parameters. In particular, the model searched by reinforcement learning had a 2M smaller



(a) NAS-TasNet-GD Repeat 3 ($R = 3$)

(b) NAS-TasNet-GD Repeat 4 ($R = 4$)

(c) NAS-TasNet-RL Repeat 3 ($R = 3$)

(d) NAS-TasNet-RL Repeat 4 ($R = 4$)

FIGURE 6. NAS-TasNets: models searched by NAS with gradient descent and reinforcement learning-based methods. We refer to these models as NAS-TasNet-GD (gradient descent) and NAS-TasNet-RL (reinforcement learning).

model size, but showed better performance on the WSJ0-2mix dataset. Also, we trained with the WSJ0-3mix dataset and compared the results to examine the generalization performance of the searched model. By comparing models with the same number of 1-D CNN block repeats, the searched model demonstrated similar performance to the backbone model and indicated that the searched models did not overfit the WSJ0-2mix dataset. The warm-up to evaluation phase required approximately two days with 4 RTX 2080 Ti GPUs. That is, the estimated efficiency of the NAS was 8 GPU-days.

D. AUXILIARY LOSS EFFECT

As mentioned in Section V, auxiliary loss alleviates the problem of architecture parameter update imbalance between mixture operations and prevents the operation of some layers from being randomly selected. To investigate the effect of auxiliary loss, we compared the average entropy reduction for each repeated mixed operation with and without auxiliary loss. For the experiment, a mixed operation network was composed of four mixed operation repeats and each repeat was composed of 8 mixed operations. A search stage of 80 epochs was performed and the average entropy of each repeat was computed for every epoch.

Figure 7(a) depicts the decrease in the average entropy for each repeated mixed operation when auxiliary loss was not applied. The most significant decrease in entropy was observed in repeat 4 located at the end of the network, whereas the least entropy reduction was observed in repeats 1 and 2 located at the front of the network. The results indicated that the operations at the front of the network were selected almost randomly and the operations at the end of the network converged early, such that various operations could not be attempted. Figure 7(b) shows the experiment using the auxiliary loss method. The average entropies of all mixed operation repeats decreased to a similar level, indicating that the architecture parameter updates in the entire mixed operation network were balanced. The randomness of repeats 1 and 2 can be regarded as significantly reduced by utilizing the auxiliary loss method. Figure 7(c) compares the average entropy reductions for the entire separation module during NAS, with and without auxiliary loss. We observed that with or without auxiliary loss, the entropy reductions for

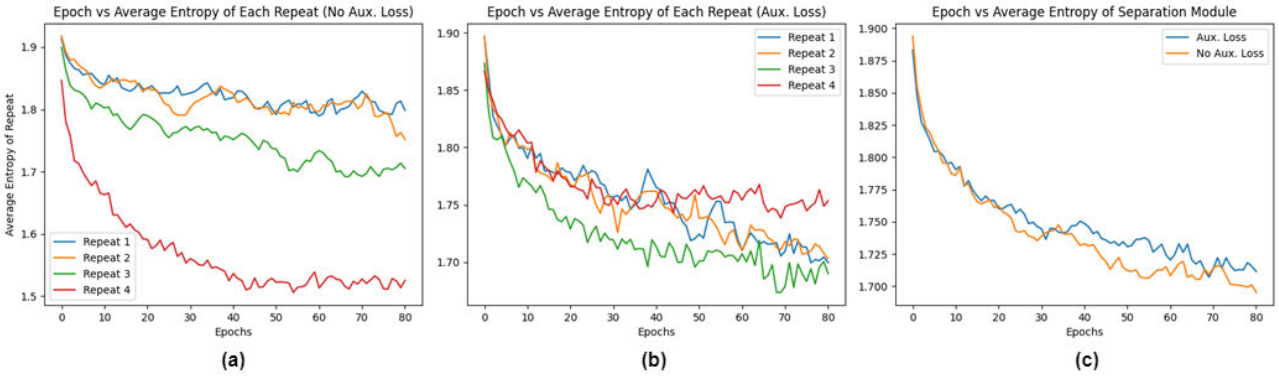


FIGURE 7. Visualizations of the observed average entropy during the search stage composed of four mixed operation repeats ($R_{\max} = 4$) and each mixed operation repeat consisted of 8 mixed operations ($X_{\max} = 8$). In (a) and (b), each average entropy for each mixed operation repeat is depicted by different lines. (a) shows the average entropy decrease during the search stage when the auxiliary loss was not applied; an imbalance in the average entropy for each repeat was observed as the search proceeded. (b) is when the auxiliary loss was used. The deviation of the average entropy between each repeat was less than in (a). (c) is a visualization of the comparison of the mean entropy reduction for the entire mixed operation network during the search stage with and without auxiliary loss, in both cases decreasing to a similar level.

TABLE 3. Separation performance comparison of the explored model and the backbone (TDCN++) model with and without the auxiliary loss method training.

Model	Model Size	Train with Aux. Loss	WSJ0-2mix		WSJ0-3mix	
			SI-SDRi (dB)	SDRi (dB)	SI-SDRi (dB)	SDRi (dB)
TDCN++ [25]	6.7M	✗	16.91	17.16	14.16	14.48
		✓	**17.78**	**18.01**	**14.19**	**14.53**
NAS-TasNet-GD	6.3M	✗	17.38	17.62	14.14	14.46
		✓	17.72	17.96	14.12	14.47
NAS-TasNet-RL	5.0M	✗	17.11	17.35	13.37	13.70
		✓	17.50	17.74	14.00	14.34

the entire separation module were similar. To summarize the analysis of all plots in Figure 7, the auxiliary loss method can be concluded to reduce the entropy deviation between each repeated mixed operation, and the method prevents mixed operations in specific locations from randomly determining an operation.

A separation accuracy improvement experiment according to the application of auxiliary loss was also conducted (see Table 3). First, as the auxiliary loss method could be applied to regular separation model training, we compared the separation accuracy with and without auxiliary loss applied to TDCN++ with the same model configuration. A significant performance improvement was observed on the WSJ0-2mix dataset when the separation model was trained with the auxiliary loss and a slight performance improvement was observed for the WSJ0-3mix. Next, NAS-TasNet-GD and NAS-TasNet-RL were trained without auxiliary loss training, and evaluated and compared with the previous results. The auxiliary loss method also improved the performance of the explored networks. In particular, for NAS-TasNet-RL trained with the WSJ0-3mix, a significant performance difference was observed between the two cases. Subsequently, we compared NAS-TasNet and TDCN++ without auxiliary loss training. NAS-TasNets exhibited a better performance with relatively small model sizes. For the WSJ0-3mix dataset, we observed similar separation performance except for NAS-TasNet-RL. Finally, when comparing the performance of each model subjected to auxiliary loss training, the performance of TDCN++ was the best, with a slight difference. We inferred that the auxiliary loss method designed for Conv-TasNet architecture could significantly improve separation performance without generating additional parameters.

VII. CONCLUSION

In this study, we attempted to extend the neural architecture search to a speech separation model. First, we used an end-to-end mask estimation-based speech separation model (Conv-TasNet) as the backbone model and applied NAS to the separation module of the network. To apply NAS, the search space for the separation module was defined, the network was represented as a DAG, and the edges of the DAG were configured as mixed operations to construct an over-parameterized network (mixed operation network) for NAS. Then, the network was explored by applying the gradient descent algorithm and reinforcement learning

algorithm-based search strategies to the constructed network for NAS. In this process, the binary gate, a GPU memory-saving algorithm, was applied to overcome the limitation of Conv-TasNet that uses excessive memory for training.

Next, when NAS was simply applied, we observed that operations in certain locations in the network were selected almost randomly. This phenomenon was derived from the architecture parameter update imbalance and an auxiliary loss method appropriate for Conv-TasNet was devised to alleviate it. We concluded that the auxiliary loss eased the parameter update imbalance and assisted the separation model training to improve the separation performance.

Finally, the derived search results, NAS-TasNet-GD and NAS-TasNet-RL, were evaluated by training from scratch on the WSJ0-2mix and WSJ0-3mix datasets. The explored model outperformed or its performance was similar to the existing performance, with fewer parameters than the baseline method.

REFERENCES

[1] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 26, no. 10, pp. 1702–1726, Oct. 2018.

[2] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "An experimental study on speech enhancement based on deep neural networks," *IEEE Signal Process. Lett.*, vol. 21, no. 1, pp. 65–68, Jan. 2014.

[3] Y. Xu, J. Du, L.-R. R. Dai, and C.-H. H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE Trans. Audio. Speech. Language Process.*, vol. 23, no. 1, pp. 7–19, May 2015.

[4] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder," in *Proc. Interspeech*, Aug. 2013, pp. 436–440.

[5] J. R. Hershey, Z. Chen, J. L. Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 31–35.

[6] Y. Isik, J. L. Roux, Z. Chen, S. Watanabe, and J. R. Hershey, "Single-channel multi-speaker separation using deep clustering," in *Proc. Interspeech*, Sep. 2016, pp. 545–549.

[7] D. Yu, M. Kolbaek, Z.-H. Tan, and J. Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 241–245.

[8] M. Kolbaek, D. Yu, Z.-H. Tan, and J. Jensen, "Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 25, no. 10, pp. 1901–1913, Oct. 2017.

[9] Z. Chen, Y. Luo, and N. Mesgarani, "Deep attractor network for single-microphone speaker separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 246–250.

[10] Y. Luo, Z. Chen, and N. Mesgarani, "Speaker-independent speech separation with deep attractor network," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 26, no. 4, pp. 787–796, Apr. 2018.

[11] Z.-Q. Wang, J. L. Roux, and J. R. Hershey, "Alternative objective functions for deep clustering," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 686–690.

[12] C. Li, L. Zhu, S. Xu, P. Gao, and B. Xu, "CBLDNN-based speaker-independent speech separation via generative adversarial training," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 711–715.

[13] S. Venkataramani, J. Casebeer, and P. Smaragdis, "End-to-end source separation with adaptive front-ends," in *Proc. 52nd Asilomar Conf. Signals, Syst., Comput.*, Oct. 2018, pp. 684–688.

[14] D. Stoller, S. Ewert, and S. Dixon, "Wave-U-Net: A multi-scale neural network for end-to-end audio source separation," in *Proc. Int. Soc. Music Inf. Retr. Conf. (ISMIR)*, 2018, pp. 334–340.

[15] S.-W. Fu, T.-W. Wang, Y. Tsao, X. Lu, and H. Kawai, "End-to-end waveform utterance enhancement for direct evaluation metrics optimization by fully convolutional neural networks," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 26, no. 9, pp. 1570–1584, Sep. 2018.

[16] Y. Luo and N. Mesgarani, "TaSNet: Time-domain audio separation network for real-time, single-channel speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 696–700.

[17] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 27, no. 8, pp. 1256–1266, Aug. 2019.

[18] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2016, pp. 47–54.

[19] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 156–165.

[20] A. V. D. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," in *Proc. ISCA Speech Synth. Workshop*, 2016, p. 125.

[21] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-path RNN: Efficient long sequence modeling for time-domain single-channel speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 46–50.

[22] E. Tzinis, Z. Wang, and P. Smaragdis, "Sudo RM-RF: Efficient networks for universal audio source separation," in *Proc. IEEE 30th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Sep. 2020, pp. 1–6.

[23] J. Chen, Q. Mao, and D. Liu, "Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation," in *Proc. Interspeech*, Oct. 2020, pp. 2642–2646.

[24] C. Subakan, M. Ravanelli, S. Cornell, M. Bronzi, and J. Zhong, "Attention is all you need in speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 21–25.

[25] I. Kavalerov, S. Wisdom, H. Erdogan, B. Patton, K. Wilson, J. L. Roux, and J. R. Hershey, "Universal sound separation," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust. (WASPAA)*, Oct. 2019, pp. 175–179.

[26] E. Tzinis, S. Wisdom, J. R. Hershey, A. Jansen, and D. P. W. Ellis, "Improving universal sound separation using sound classification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 96–100.

[27] D. Samuel, A. Ganeshan, and J. Naradowsky, "Meta-learning extractors for music source separation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 816–820.

[28] E. Tzinis, S. Venkataramani, Z. Wang, C. Subakan, and P. Smaragdis, "Two-step sound source separation: Training on learned latent targets," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 31–35.

[29] M. W. Y. Lam, J. Wang, D. Su, and D. Yu, "Mixup-breakdown: A consistency training method for improving generalization of speech separation models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 6374–6378.

[30] N. Zeghidour and D. Grangier, "Wavesplit: End-to-end speech separation by speaker clustering," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 29, pp. 2840–2849, 2021.

[31] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 2902–2911.

[32] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4780–4789.

[33] H. Zhu, Z. An, C. Yang, K. Xu, E. Zhao, and Y. Xu, "EENA: Efficient evolution of neural architecture," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1891–1899.

[34] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–16.

[35] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.

[36] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2423–2432.

[37] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 2787–2794.

[38] H. Pham, M. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 4095–4104.

[39] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–12.

[40] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1294–1303.

[41] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1761–1770.

[42] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 19–34.

[43] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–13.

[44] Y. Weng, T. Zhou, Y. Li, and X. Qiu, "NAS-UNet: Neural architecture search for medical image segmentation," *IEEE Access*, vol. 7, pp. 44247–44257, 2019.

[45] A. Kwasigroch, M. Grochowski, and A. Mikolajczyk, "Neural architecture search for skin lesion classification," *IEEE Access*, vol. 8, pp. 9061–9071, 2020.

[46] X. Yan, Y. Jiang, W. Shi, and C. Zhuo, "MS-NAS: Multi-scale neural architecture search for medical image segmentation," in *Proc. Med. Image Comput. Comput. Assist. Intervent (MICCAI)*, vol. 12261, 2020, pp. 388–397.

[47] Q. Yu, D. Yang, H. Roth, Y. Bai, Y. Zhang, A. L. Yuille, and D. Xu, "C2FNAS: Coarse-to-fine neural architecture search for 3D medical image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4125–4134.

[48] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 82–92.

[49] Z. Xu, S. Zuo, E. Y. Lam, B. Lee, and N. Chen, "AutoSegNet: An automated neural network for image segmentation," *IEEE Access*, vol. 8, pp. 92452–92461, 2020.

[50] X. Zhang, H. Xu, H. Mo, J. Tan, C. Yang, L. Wang, and W. Ren, "DCNAS: Densely connected neural architecture search for semantic image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13956–13967.

[51] Y.-C. Chen, J.-Y. Hsu, C.-K. Lee, and H.-Y. Lee, "DARTS-ASR: Differentiable architecture search for multilingual speech recognition and adaptation," in *Proc. Interspeech*, Oct. 2020, pp. 1803–1807.

[52] S. Hu, X. Xie, M. Cui, J. Deng, S. Liu, J. Yu, M. Geng, X. Liu, and H. Meng, "Neural architecture search for LF-MMI trained time delay neural networks," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 30, pp. 1093–1107, 2022.

[53] T. H. Trinh, A. M. Dai, M.-T. Luong, and Q. V. Le, "Learning longer-term dependencies in RNNs with auxiliary losses," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 4965–4974.

[54] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[55] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 1997–2017, 2019.

[56] J. L. Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR—Half-baked or well done?" in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 626–630.

[57] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 6105–6114.

[58] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, 1992.

[59] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2820–2828.

[60] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *IEEE Trans. Audio, Speech, Language Process.*, vol. 14, no. 4, pp. 1462–1469, Jul. 2006.

[61] M. Pariente, S. Cornell, J. Cosentino, S. Sivasankaran, E. Tzinis, J. Heitkaemper, M. Olvera, F.-R. Stöter, M. Hu, J. M. Martín-Doñas, D. Ditter, A. Frank, A. Deleforge, and E. Vincent, "Asteroid: The PyTorch-based audio source separation toolkit for researchers," in *Proc. Interspeech*, Oct. 2020, pp. 2637–2641.

JOO-HYUN LEE received the B.S. degree in electrical engineering from Konkuk University, Seoul, South Korea, in 2018. He is currently pursuing the M.S. degree in electronic engineering with Hanyang University, Seoul. His research interests include acoustic signal processing and deep/machine learning.

JOON-HYUK CHANG (Senior Member, IEEE) received the B.S. degree in electronics engineering from Kyungpook National University, Daegu, South Korea, in 1998, and the M.S. and Ph.D. degrees in electrical engineering from Seoul National University, Seoul, South Korea, in 2000 and 2004, respectively. From 2000 to 2005, he was with Netdus Corp., Seoul, as CTO. From 2004 to 2005, he was a Postdoctoral Researcher with the University of California at Santa Barbara, Santa Barbara, CA, USA, where he was involved in adaptive signal processing and audio coding. In 2005, he joined the Korea Institute of Science and Technology, Seoul, as a Research Scientist, where he was involved in speech recognition. From 2005 to 2011, he was an Assistant Professor with the School of Electronic Engineering, Inha University, Incheon, South Korea. He is currently a Full Professor with the School of Electronic Engineering, Hanyang University, Seoul. His research interests include speech recognition, deep/machine learning, artificial intelligence (AI), speech processing, acoustic signal processing, and bio-medical signal processing. He was a recipient of the IEEE/IEEK IT Young Engineer Award, in 2011. He is currently serving on the Editorial Board of the *Digital Signal Processing* journal (Elsevier).

JAE-MO YANG received the M.S. and Ph.D. degrees in electrical and electronic engineering from Yonsei University, Seoul, South Korea, in 2009 and 2014, respectively. He was an Intern at Microsoft Research Asia, Beijing, China, from 2010 to 2011, and Microsoft Research, Redmond, WA, USA, in 2011. He is currently a Principal Researcher at the Advanced Audio Laboratory, Samsung Electronics, South Korea. His research interests include speech/audio signal processing, speech enhancement, microphone arrays and machine learning, and specifically deep learning.

HAN-GIL MOON received the Ph.D. degree in acoustics and audio engineering from Seoul National University. In 2005, he joined the Digital Media and Communications Research and Development Center. In 2010, he took a short sabbatical to complete a Postdoctoral Researcher at Cambridge University, before enrolling on the Mobile Team, in 2016. He is currently the VP of technology (Master) and the Head of the Advanced Audio Laboratory, Samsung's Mobile Communications Business, Samsung Electronics. He plays a leading role in developing audio solutions for Galaxy smartphones, smartwatches, and earbuds. His current research interests include wireless audio, speech enhancement, hearing aids, and AI audio signal processing.