Reinforcement Learning based Neural Architecture Search for Audio Tagging

Haiyang Liu
Graduate School of Information, Production and Systems
Waseda University
Tokyo, Japan
haiyangliu@toki.waseda.jp

Cheng Zhang
School of Instrument Science and Engineering
Southeast University
Nanjing, China
213172942@seu.edu.cn

***Abstract*—Audio tagging aims to assign tags for an audio chunk, and it has attracted increasing attention as its potential applications seem to be evident. Deep learning technologies have been successfully applied to domestic audio tagging task. However, the performance of deep models is heavily relied on the hyper-parameters selection such as the filter size in the convolutional layers. Recently, Neural Architecture Search (NAS) has been successfully applied to design deep model architectures for specified learning task. In this paper, we explore the neural architecture search method for domestic audio tagging. We propose to use the Convolutional Recurrent Neural Network (CRNN) with Attention and Location (ATT-LOC) as the audio tagging model. Then, we apply NAS to search for the optimal number of filters and the filter size. Finally, we employ a grid search over the mixup augmentation coefficient, the input size of the spectrogram and the value of batch size to further improve the classification results. As demonstrated in our experiments, the architecture found by automatic searching achieves an equal error rate of 0.095 on DCASE 2016 task 4 dataset, outperforming the CRNN baseline of 0.10. In addition, the architecture found by NAS achieves a faster convergence rate in training than the CRNN baseline.**

***Index Terms*—audio tagging, convolutional recurrent neural network, neural architecture search, reinforcement learning**

I. Introduction

Domestic audio tagging aims to tag an audio clip recorded from home environment. The clips are typically short segments, and the tags can be one (or more) of pre-determined sound sources present in the recording, such as, "adult male speech", "broadband noise" and "video games". Domestic audio tagging has received widespread attention in plenty of applications, such as audio surveillance [1], lifelogging [2] and health activity monitoring [3]. The labeled audio data could be divided into two types: *clip level* labeled audio data and *event level* labeled audio data. The difference between these two kinds of labeling methods is that the *event level* labeled audio clips are labeled with both the corresponding tags and their occurrence time, while the *clip level* labeled audio clips are only labeled with the presence of the tags but not their occurrence time. In this paper, we focus on the *clip level* labeled audio data, which is commonly used in recent researches [4]–[6]. However, the *clip level* labeled audio data is difficult to use for the following reasons. Firstly, the tags only indicate whether the corresponding events exist in an audio clip but does not include the specific occurrence times. Secondly, different events vary in duration dramatically. For example, some events such as "speech" may last for a few minutes, while other events such as "gunshot" may only last for hundreds of milliseconds. Lastly, the overlapping of the sound events imposes more challenge for the task.

Much research has been investigated on the audio tagging task. The traditional method for audio tagging relied on the Gaussian mixture model (GMM) trained on Mel frequency cepstrum coefficients (MFCCs) [7]. Since DCASE 2016 challenge, Deep Neural Networks (DNNs) have been used to classify the audio clips [8]–[10]. Different from the GMM method which employs one VS rest policy, DNNs can distinguish the tags in shared weights simultaneously. Recently, Convolutional Neural Networks (CNNs) have been shown to outperform DNNs on this task as CNNs can capture spatial information from the input representation of the audio signal [11], [12]. With the aim of modeling the long-term temporal structure of the audio signal, Convolutional Gated Recurrent Neural Network (CGRNN) [5] is proposed, which combines the CNN and the Gated Recurrent Unit (GRU). In order to locate the occurred acoustic events in an audio clip, Attention and Location (ATT-LOC) modules are proposed [4]. The attention module is used for frame-level feature selection for each sound event, while the localization module is used to find the locations of each event. Recently, the best performing system [6] applies the mixup augmentation method on the existing ATT-LOC module. The mixup augmentation method here aims to expand the training data size by creating new samples and improve the performance of the trained model.

However, for the domestic audio tagging task, the performance of the deep models relied on the hyper-parameters selection such as the number of filters, the filter height and the filter width in the convolutional layers. As these hyper-parameters are not optimal, the CRNN model still has room for improvement. Therefore, it is of great interest to find a better configuration for the hyper-parameters, to further improve the classification results.

Recently, Neural Architecture Search (NAS) has been successfully applied to design model architectures for specified learning tasks, such as image classification and language modeling [13]–[17]. NAS aims to find the most promising

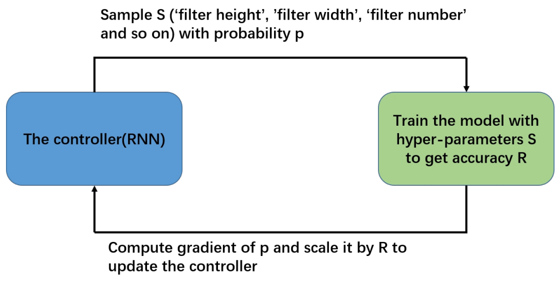
Sample S ('filter height', 'filter width', 'filter number' and so on) with probability p
The controller(RNN)
Train the model with hyper-parameters S to get accuracy R
Compute gradient of p and scale it by R to update the controller

Fig. 1. An overview of Neural Architecture search for audio tagging.

architectures on the task of desire by reinforcement learning algorithms. In NAS, a Recurrent Neural Network (RNN) is used as a controller to generate a "child neural network" (a candidate architecture which has fewer parameters than the full network for speeding up the training process). Then, the generated network could be trained on the specified dataset until convergence to get accuracy as the reward signal for the controller (see Figure 1). Based on the reward signal, the controller can be trained with some reinforcement learning algorithms. Therefore, in the next iteration, the controller can leave more opportunities to generate the architectures which can receive higher rewards. The work of NAS is established on the observation that the architecture of a neural network can be typically specified by a string (a list of hyper-parameters) whose length is variable.

In this paper, we explore the Neural Architecture Search method to search for the models for the CHIME-HOME dataset of the DCASE 2016 audio tagging challenge [18]. We first apply the CRNN model with ATT-LOC modules and the mixup augmentation method as our baseline model, which is the previous state-of-the-art method [6] for CHIME-HOME dataset. Then, with the aim of pruning the search space and improving the search efficiency of NAS, we divide the search space into two parts: hyper-parameters of the convolutional layers and hyper-parameters of the training details. After specifying the architecture of our model, we implement a grid search over the hyper-parameters for training including the batch size, the hyper-parameter of mixup approach, and the input size of the spectrogram, which can better suit the capability of the found architecture and further improve the training process. Extensive experiments are conducted to demonstrate that our approach could find several promising architectures which achieve better classification results than the previous state-of-the-art method [6]. The architecture found by NAS can not only achieve state-of-the-art performance (**0.095** EER) but also maintain a faster convergence speed compared with the baseline system.

The remainder of this paper is organized as follows. In section 2, the related work is presented. Section 3 introduces the Convolutional Recurrent Neural Network with Attention and Localization Model. The Mixup augmentation method is also given in this part. Section 4 presents the NAS approach for audio tagging. Section 5 presents the experimental settings, training details as well as the analysis of the results. The discussions of our approach are shown in Section 6.

II. Related Work

Our approach is related to two research fields, including audio tagging and neural architecture search.

A. Audio Tagging

Audio classification and detection have attracted increasing attention in recent years [19]. There are many challenges for audio detection and tagging such as DCASE 2013 [20], DCASE 2016 [18] and DCASE 2017 [21]. Many approaches described above rely on the bag of frames (BOF) model [22]. BOF is established on the assumption that tags occur in all frames. However, this assumption could not be demonstrated in practice because some audio events like "gunshot" only happen a short time in audio clips. Another audio tagging method [23], [24] transforms waveform to the time-frequency (T-F) representation, which can be regarded as an image. However, there exist some difference between images and audio clips. In an image, the objects which need to be recognized usually occupy a dominant part. However, in an audio clip, audio events only occur a short time. To solve this problem, some models with the attention mechanism [4] for audio tagging and classification have been applied to let the neural networks learn to focus more on the audio events and ignore the background noises. Recently, a mixup-based approach with a convolutional recurrent neural network and attention module [6] have been proposed to improve the performance of audio tagging.

B. Neural Architecture Search

NAS has made a great contribution to the development of automatically designing neural networks. The controller in NAS does not predict all the outputs simultaneously. Instead, it predicts hyper-parameters once a time relied on the previous predictions. This idea has some similarity with the decoding process in end-to-end Sequence to Sequence Learning [17]. A difference with sequence to sequence learning is that the metric (the accuracy of the child network) NAS optimizes is non-differentiable. In a sense, NAS is similar to the work on BLEU optimization in Neural Machine Translation [25]. A significant difference is that NAS learns directly from the reward signal without any supervised learning mechanism.

Another area which is related to NAS is the idea of learning to learn or meta-learning [26], whose goal is to use the information learned in one task to improve a future task. Besides, the idea of using a neural network to learn the gradient descent updates for another network [27] and the idea of using reinforcement learning to find update policies for another network [28] are also related to NAS.

However, a drawback for NAS is the computational expense, which limits its application range. Therefore, a great number of methods which aim to improve the efficiency of NAS have been proposed. Efficient Neural Architecture Search (ENAS) [29] is a novel method that speeds up NAS by sharing weights between architectures. This work is established on the assumption that the computational bottleneck of NAS is the training of each child model to convergence. ENAS preserves all the trained weights and optimizes the training process. The idea of sharing weights between architectures is inspired by the concept of weight inheritance in neural model evolution [30]. There are other works which focus on improving the search efficiency of NAS, such as using performance prediction [31], using iterative search method for architectures of growing complexity [32] and using a hierarchical representation of architectures [16].

III. CRNN with ATT-LOC Modules and the Mixup Augmentation Method

The baseline system in our experiments is the Convolutional Recurrent Neural Network (CRNN) with Attention and Localization (ATT-LOC) modules and the Mixup augmentation method, which have achieved state-of-the performance as claimed in previous studies. The architecture of our baseline system is shown in Figure 2. In this section, we will divide the baseline model into two parts (CRNN, ATT-LOC) and introduce the two parts in detail.

A. Convolutional Recurrent Neural Network

The Convolutional Recurrent Neural Network consists of two parts: the convolutional layers to extract the features and the Gated Recurrent Units (GRU) to model the long-term pattern along the input.

The form of the input data is log mel spectrograms [33], which is obtained by applying Short-time Fourier transform (STFT) on the audio waveform. In the process of STFT, a Hamming window with a size of 1024 is chosen. After specifying the format of the input, convolutional layers are specified to extract high-level features. Here, we denote Conv block as a convolutional operation with N filters and a kernel size of $H \times W$, a batch normalization layer, a max-pooling layer of 1×2 and a dropout layer with ratio 0.1 for preventing overfitting. Besides, Exponential linear units (ELUs) [34] are chosen for activation because they can make a contribution to faster learning and lead to better generalization performance than ReLUs [34]. The hyper-parameters of the convolutional layers in the baseline CRNN are manually specified, while we apply reinforcement learning algorithms to search for the most promising hyper-parameters. The outputs of the convolutional layers are fed into the Gated Recurrent Unit (GRU) to model the long-term pattern along the whole chunk [5]. The loss can be calculated as:

$$E = -\sum_{n=1}^{N} (P_n \log O_n + (1 - P_n) \log(1 - O_n)), \quad (1)$$

$$O = \frac{1}{T} \sum_{t=0}^{T-1} (1 + \exp(-S_t))^{-1}, \quad (2)$$

where E stands for the binary cross-entropy, N stands for the bunch size, O_n and P_n denote the estimated and reference tag vector at sample index n, respectively. The CNN linear output is defined as S_t at t_{th} frame. T represents the total number of frames in the audio chunk.

B. Attention and Localization Modules

Here, we will introduce the attention and location modules [4] in our baseline model. The diagram of the attention and localization modules is shown in Figure 3. The attention mechanism, which is an additional sigmoid layer with one node output, can effectively reduce the impact of background noise on the output and force the classifier to focus more on the frame in which the acoustic events occur. In other words, the attention module aims to predict the importance of each frame for the final labels. The predicted attention factor $Z_{att}(t)$ at the t_{th} frame indicates the importance of the current frame for the final labels:

$$Z_{att}(t) = \sigma(W_{att} * X_t + b_{att}), \quad (3)$$

where X_t is the input feature at the t_{th} frame. σ is the sigmoid function. W_{att} and b_{att} denote the weights and bias of the attention module. Then, the importance of the current frame is multiplied with the CNN output to make the classifier more deterministic:

$$Y^{`}_{cnn}(t) = Z_{att})(t) Y_{cnn}(t), \quad (4)$$

where $Y_{cnn}(t)$ denotes the result processed by the activation function of CNN. The weighted feature against the background noise is denoted by $Y^{`}_{cnn}(t)$. This weighting process can make the classifier focus more on important frames and neglect the unrelated frames. Finally, the produced attention weight is also applied to the final acoustic tag outputs at each frame. It is defined as,

$$O^{`}(t) = Z_{att}(t) O(t), \quad (5)$$

where $O(t)$ denotes the tag prediction output at the t_{th} frame and $O^{`}_t$ denotes the weighted output. The attention weight $Z_{att}(t)$ can measure the contribution degree at the t_{th} frame for the final predicted label.

At the same time, the localization module can localize the acoustic events which occurred in the whole audio chunk. It is meaningful to predict the accurate temporal locations (at frame-level) of the occurred acoustic events. The localization module is one Softmax layer without any hidden layer. The localization vector $Z_{loc}(t)$ in Figure 3 denotes the localization vector at the t_{th} frame. The working process of the localization module is quite similar to the attention factor calculation process, and more details can be found at [4].

IV. Neural Architecture Search for Audio Tagging

In the following section, we will demonstrate the fact that the convolutional layers of the baseline model are sensitive to hyper-parameters for audio tagging tasks. Then, we will describe our method which uses a recurrent neural network (RNN) as the controller to search for better convolutional

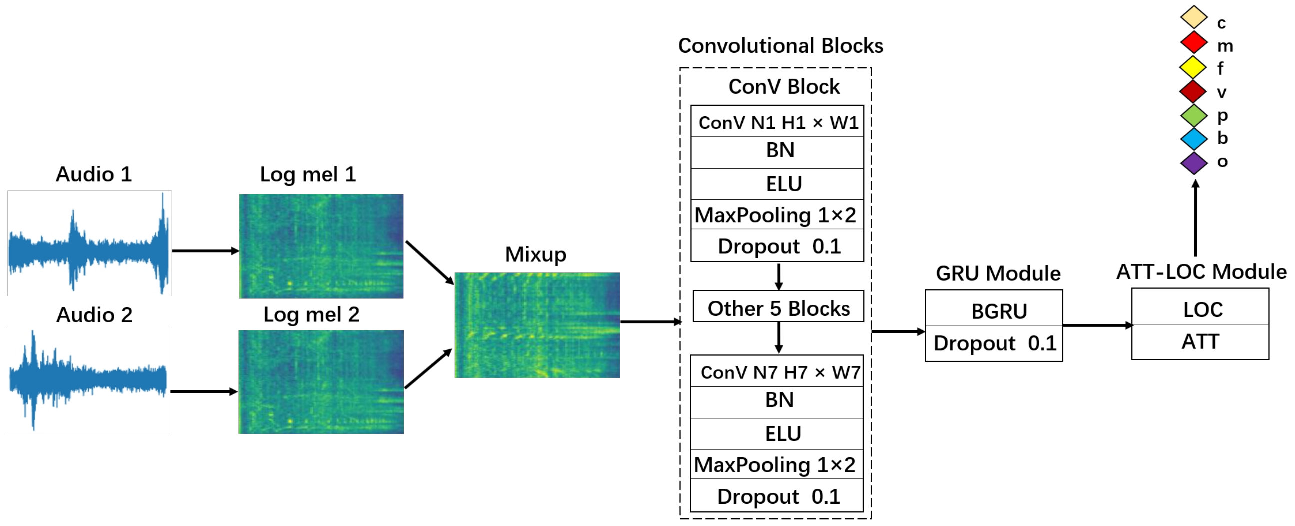

Fig. 2. System architecture for audio tagging. Data augmentation is operated on the log mel spectrogram. There are 7 Conv blocks. The number of filters and the filter size in the convolutional layers are specified from experience.

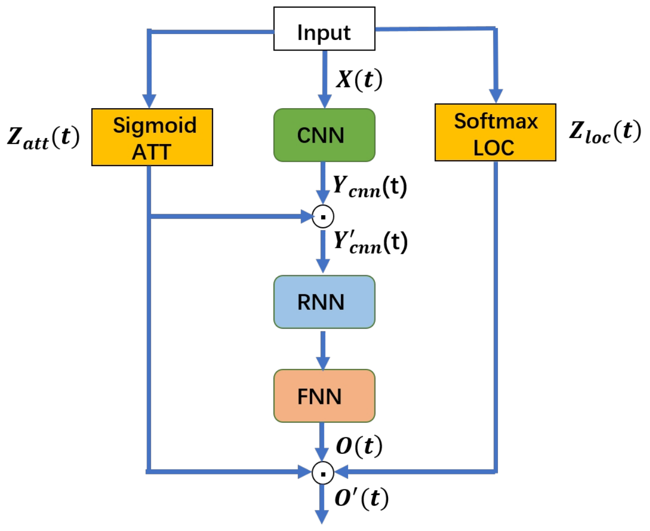

Fig. 3. The diagram of the attention and localization modules based on CRNN for audio tagging. ATT denotes the attention module. LOC represents the localization module.

architectures. Finally, we will show the details for training the controller RNN to maximize the expected accuracy of the sampled architectures by Policy Gradient [35] algorithm.

A. Sensitivity of the Convolutional Layers for Audio Tagging Tasks

For audio tagging tasks, the hyper-parameters of the convolutional layers will have an impact on the final classification results. The previous work in [6] chooses different Conv filter sizes experimentally without instructions on how to design the hyper-parameters. The ATT-LOC model has only one convolutional layer with a big kernel size of 30×1. In practice,

TABLE I
EXPERIMENTAL RESULTS OF ADJUSTING HYPER-PARAMETERS IN THE CONVOLUTIONAL LAYERS ON THE BASIS OF CRNN. N_i DENOTES THE NUMBER OF FILTERS IN THE i_{th} LAYER.

N_1	N_2	N_3	N_4	N_5	N_6	N_7	**EER**
8	16	32	64	64	64	64	0.104
8	32	64	16	8	64	64	0.141
16	8	32	64	32	64	64	0.131
32	64	32	8	32	64	64	0.115

different layers in CNN can extract the features of different levels. Therefore, the more layers the convolutional neural network has, the richer features of different levels can be extracted. Based on this point of view, the CRNN model proposed in [6] substitutes the convolutional part in ATT-LOC with seven convolutional layers which are equipped with a small kernel size of 3×3. Experiments have shown that the substituted model can further improve the classification results [6]. However, even though deeper convolutional layers can improve the classification results, merely increasing the depth will not always get acceptable results and may result in gradient dispersion or gradient explosion. Therefore, the models in audio tagging tasks are sensitive to hyper-parameters. In our experiments, we search for the promising hyper-parameters which will make the capability of the model better match the size of CHIME-HOME dataset, compared with the previous state-of-the-art model [6].

Here, we demonstrate our assumption by adjusting some hyper-parameters in the convolutional layers of CRNN [6]. In concrete, we adjust the number of filters in different convolutional layers (do not change the other hyper-parameters) and test the corresponding performance on the evaluation set. As shown in Table 1, the performance of the CRNN model fluctuates dramatically (from 0.104 to 0.141) even if we only adjust a few hyper-parameters.

B. Generate Hyper-parameters of Convolutional Layers by Policy Gradient

We use a controller to generate architectural descriptions of neural networks. The controller is implemented as a recurrent neural network. The model which we focus on is the convolutional layers of the CRNN model. Therefore, we can use the controller to generate the corresponding hyper-parameters (filter number, filter width, filter height) as a sequence of tokens. To make the search process more manageable, we first follow the experimental settings in [6] and specify the number of convolutional layers and the size of input data as constant values. In the following experiments, we will change the number of layers according to the size of input by grid search.

Once the controller RNN finishes the process of generating an architecture, a neural network with this architecture and the ATT-LOC module is built and trained. At convergence, the accuracy of the network on the evaluation set is recorded. The parameters of the controller RNN θ_c, are then optimized with the aim of maximizing the expected validation accuracy of the proposed architectures. In the next section, we will describe a policy gradient method which we use to update parameters θ_c so that the controller RNN can generate better architectures over time.

C. Training with REINFORCE

The list of tokens that the controller predicts can be viewed as a list of actions $a_{1:T}$ to design an architecture for a child network. Early stopping mechanism is used in the training process. In other words, if the validation loss has not been improved after 20 epochs, the training process will be interrupted. Then, the child network will achieve an eer on the evaluation set. We can transform it to $1 - eer$ as the reward signal R and use the reinforcement learning algorithm REINFORCE [36] to train the controller. REINFORCE does not complete the back-propagation process by the loss. Instead, it adjusts the probability of choosing a specific action by the value of rewards. The controller will give higher possibilities to the actions that get high rewards. To find the promising architecture, the controller needs to maximize its expected reward, represented by $J(\theta_c)$:

$$J(\theta_c) = E_{P(a_{1:T};\theta_c)}[R] \tag{6}$$

where θ_c represents the parameters of the controller. Here, REINFORCE directly optimizes the parameterized stochastic policy $P(a_{1:T};\theta_c)$ by performing gradient ascent on the expected reward. The expectation is implicitly taken over all possible trajectories. Denote $d(x)$ as the possibility distribution under the policy $P(a_t|a_{(t-1):1};\theta_c)$. Denote A as the action space of the current state $a_{(t-1):1}$. The function above can be rewritten as:

$$J(\theta_c) = \sum_{t=1}^{T}(d(a_{(t-1):1}) \sum_{a_t \in A} P(a_t|a_{(t-1):1};\theta_c)R) \tag{7}$$

since the reward signal R is non-differentiable, we need to use a policy gradient method iteratively update θ_c.

In this work, we use the REINFORCE rule:

$$\nabla_{\theta_c} J(\theta_c) = \nabla_{\theta_c} \sum_{t=1}^{T}(d(a_{(t-1):1}) \sum_{a_t \in A} P(a_t|a_{(t-1):1};\theta_c)R) \tag{8}$$

$$\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T}(d(a_{(t-1):1}) \sum_{a_t \in A} \nabla_{\theta_c} P(a_t|a_{(t-1):1};\theta_c)R) \tag{9}$$

Here, we suppose that the function above is differentiable when $P(a_t|a_{(t-1):1};\theta_c)$ is zero. In the case of $\nabla_{\theta_c} P(a_t|a_{(t-1):1})$ is already known, we could use the method of likelihood to ratio to define $\nabla_{\theta_c} D(a_t|a_{(t-1):1})$ as the score function. Due to the reason that the function $d()$ denotes the probability distribution under the policy $P(a_t|a_{(t-1):1};\theta_c)$ and $a_{(t-1):1}$ denotes a specific state in the exploration process, we can get $\sum_{t=1}^{T} d(a_{(t-1):1}) = 1$. Therefore, Equation (12) could be re-written as:

$$\nabla_{\theta_c} J(\theta_c) = \sum_{i=1}^{T} E_{P(a_{1:T};\theta_c)}[\nabla_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c)R] \tag{10}$$

An empirical approximation of the above quantity is:

$$\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T} \nabla_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c)R_k \tag{11}$$

where m is the number of different architectures that the controller samples in one batch and T is the number of hyper-parameters our controller has to predict to design a neural network architecture. The validation accuracy that the k_{th} neural network architecture achieves after being trained on a training dataset is R_k.The above update is an unbiased estimate for our gradient but has a very high variance. In order to reduce the variance of this estimate, we employ a baseline function:

$$\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T} \nabla_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c)(R_k - b) \tag{12}$$

In our experiments, the baseline b is an exponential moving average of the previous architecture accuracy. Based on this, the controller can judge whether the current reward is valuable compared with the previous results. First, we should initialize the controller class and build the corresponding modules. Then, we should collect the actions predicted by the controller with the corresponding rewards. Finally, we update the parameters of the controller based on the experience we store.

V. Experiments and Results

In this section, we will first introduce the experimental settings from the following two aspects: dataset and baseline models, search space and training details. Then, we will show the results of our experiments and the corresponding analyze.

TABLE II
SEVEN AUDIO EVENTS IN THE CHIME-HOME DATASET

Audio Event	Event Descriptions
'b'	Broadband noise
'c'	Child speech
'f'	Adult female speech
'm'	Adult male speech
'o'	Other identifiable sounds
'p'	Percussive sound events
'v'	TV sounds or Video games

A. Dataset and Baseline Models

We apply the neural architecture search method to the CHIME-HOME dataset of the DCASE 2016 audio tagging challenge [37]. The audio recordings are produced in a domestic environment [38]. The dataset is composed of 4-second audio chunks at a sampling rate of 16kHz and their corresponding multi-label annotations or ground truth labels [39]. As shown in Table 2, there are seven acoustic event tags, including adult male speech, child speech, adult female speech, video game / TV, percussive sounds, broadband noise, and other identifiable sounds. There are 4387 recordings in the development set and 816 recordings in the evaluation set. Besides, five-fold sets are configured in the development set. The target is to predict the presence of absence of audio tags in each audio chunk.

For the format of the input features, we choose log-mel features because they compute directly from the mel-frequency spectral coefficients for each frame of raw audio. We select CGRNN [5], CRNN with mixup [6] and ATT-LOC [4] as baseline models. CGRNN and ATT-LOC.

B. Search Space and Training Details

The NAS search process is based on the baseline model shown in Section 3. For the NAS, we only search the hyper-parameters of the convolutional layers including filter height, filter width and the number of filters. We retain the ATT-LOC modules and the mixup augmentation method. Besides, the search space is divided into two parts with the aim of pruning the search space and improving the search efficiency. The first part is the hyper-parameters of the convolutional layers in our baseline model. For each convolutional layer, the controller RNN selects a filter height from $\{1, 3, 5\}$, a filter width from $\{1, 3, 5\}$, and the number of filters from $\{8, 16, 32, 64\}$. The second part is to perform a grid search over the hyper-parameters for the training process including the coefficient of mixup, batch size, and the number of mel frequency bins. For the coefficient of the mixup augmentation method, we select a value from $\{0.1, 0.2, \cdots, 2.0\}$. For the batch size of the training process, the value needs to be selected from $\{16, 32, 64, 96, 128, 160\}$. The number of mel frequency bins is selected from $\{64, 96, 128, 160, 240\}$. To be mentioned, the existing experiments have shown that deeper networks, such as VGG [40], could not get satisfactory results. This may be caused by the fact that the size of the CHIME-HOME dataset does not match the model capability. Therefore, we do not take the skip connection approach into the model, which is designed for solving the gradient back-propagation problem on deeper neural networks.

The controller RNN for NAS is modeled by a two-layer LSTM with 32 hidden units in each layer. The controller RNN is trained with the ADAM optimizer [41] with a learning rate of 0.001. The weights of the controller are initialized uniformly between -0.08 and 0.08. During the search process, the controller samples 250 architectures, which are specified experimentally. The time for training a "child network" to convergence is 20 minutes, and the training process is implemented on 4 GeForce GTX 1080 GPUs with three days.

Once the controller RNN samples an architecture, a child model with ATT-LOC module is constructed and trained for 100 epochs. Early stopping is used to monitor the validation loss. Training is interrupted if the validation loss has not decreased after 20 epochs. The reward used for updating the controller is the opposite value of the average equal error rate (EER) for five-fold cross-validation. EER is the official evaluation method for the challenge, and we followed the 5-fold cross-validation setting with the original fold splits. To be mentioned, the EER is defined as the error rate at the ROC operating point where the false positive and false-negative rates are equal, and a lower EER represents better classification performance.

C. Results

The first part of our experiments is to search for the architecture that achieves the best validation accuracy. The second part is to run a small grid search over batch size, α hyper-parameter of mixup approach, and the value of frequency bins (input size).

We first explore NAS with the hyper-parameter α=0.0 to show the effectiveness of our approach. Several architectures discovered by NAS achieve better performance than CRNN [6] with the same experimental settings including learning rate, batch size and so on. To be mentioned, the reason for choosing CRNN for comparison is that our searching process is established on CRNN. Therefore, it is meaningful to demonstrate the effectiveness of our approach by the accuracy improvement from CRNN. A notable discovery is that the selected architectures prefer a larger number of filters at bottom layers with square kernel size. The kernel filters with the size of "1×1" are hardly used and the "ConV 32, 3×3" block is selected the most frequently. Compared with CRNN, the best value of EER is decreased from 0.13 to 0.11, which indicates the effectiveness of our method.

Then, we explore the NAS with the hyper-parameter α=1.5, which achieved the best performance in the CRNN model [6]. In the initial stage of training, the architectures found by the controller have a large variance due to the reason of randomly initialized weights and large exploration rate (to sample more architectures in the search space). The exploration rate here measures the probability for the controller to predict the architectures from experience (or randomly). If the architecture is chosen randomly, the performance of the "child network" will deviate the expected level. With the training process progresses, the exploration rate is decreased to make the predictions more deterministic. In the final 50 epochs, the exploration is set to 0, and we can find that the controller shows a stable performance. The controller finally converges to the state $NAS_{\alpha=1.5}$"**32, 3, 3, 32, 3, 3, 64, 3, 3, 16, 3, 3, 64, 3, 3, 64, 3, 3, 64, 3, 3**". To be mentioned, if we denote N_i as the number of filters, H_i as the filter height, W_i as the filter width in the i_{th} layer, the state here refers to "$N_1, H_1, W_1, \cdots, N_7, H_7, W_7$". Compared with CRNN, we can find that NAS prefers a larger number of filters at the bottom layers. Besides, in order to make the results more robust, we test the trained controller 100 times and find that the architecture NAS_{best} with state "**16, 5, 5, 32, 5, 5, 64, 3, 3, 32, 3, 3, 64, 3, 3, 64, 5, 5, 64, 3, 3**" gets the best performance with EER of **0.095**.

We select the best mixup hyper-parameters of CRNN ($\alpha = 1.5, \alpha = 2.0$) and keep all the other hyper-parameters the same to make a fair comparison. It can be seen that NAS outperforms CRNN in both $\alpha = 1.5$ and $\alpha = 2.0$. The superiority benefits mostly from the "b" and "o" classes. As shown in Figure 4 and Figure 5, under the mechanism of early stopping, CRNN needs to be trained for almost 80 epochs to reach a steady state with the validation loss of 0.325 (with final EER of 0.102). In comparison, the architecture NAS_{best}, only requires 50 epochs to early stop at the validation loss of 0.272 (with final EER of 0.095). We further demonstrate the superiority of NAS on the training speed by comparing the time for NAS and CRNN to reach the same accuracy on the evaluation set. We can see that to reach the best accuracy of CRNN (0.915), CRNN needs 68 epochs while NAS only needs 49 epochs. This indicates that NAS can not only find the architecture with a lower validation loss but also has a faster convergence speed.

In order to further improve the performance of NAS, we implement a grid search over the batch size from $\{16, 32, 64, 96, 128, 160\}$, the coefficient of mixup from $\alpha = \{0.1, 0.2, \cdots, 2.0\}$ and the frequency bins from $\{64, 96, 128, 160, 240\}$. To be mentioned, the previous experiments were implemented with batch size = 44 and frequency bins = 128. We find the best combination of these hyper-parameters (batch size = 64, α = 1.7 and frequency bins = 128), which will make $NAS_{\alpha=1.5}$ (shown as $NAS_{gridsearch}$ in Table 3) achieve the EER of 0.096. For the NAS_{best} architecture, the improvements are not so obvious, fluctuating at the value of 0.095.

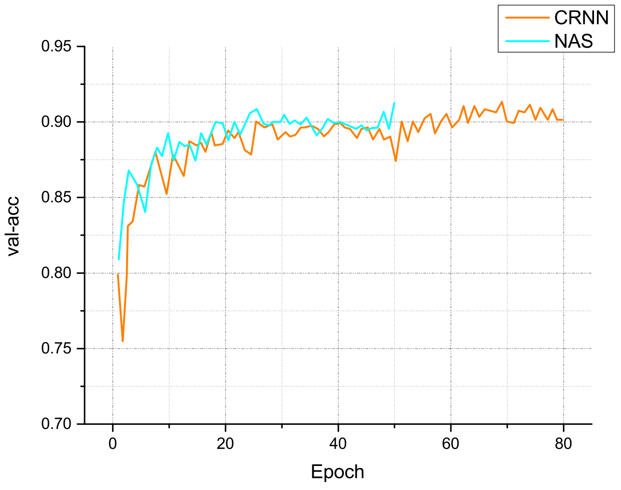

Fig. 4. The comparison on model training curves of the accuracy on the evaluation set between NAS_{best}(with final eer of 0.095) and CRNN (with final eer of 0.105).

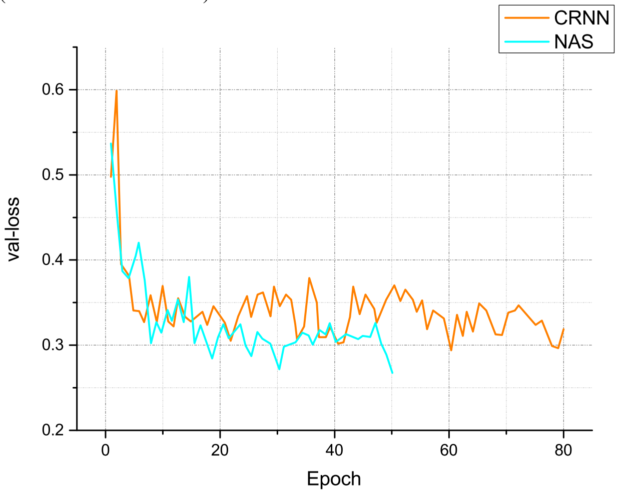

Fig. 5. The comparison on model training curves of the loss on the evaluation set between NAS_{best}(with final eer of 0.095) and CRNN (with final eer of 0.105).

VI. Conclusions

In this paper, for the audio tagging task, we explore the neural architecture search method to search the filter number and the filter size of convolutional layers on the basis of CRNN model with ATT-LOC module and mixup augmentation method. Experiments are conducted on DCASE 2016 (Task 4) task. When α hyper-parameter is set to 0.0, our method could search several promising architectures which can decrease EER from 0.13 to 0.11, compared with the previous state-of-the-art method. When α hyper-parameter is set to 1.5, the architecture found by our method can get not only a lower validation loss but also a faster convergence speed. Combined with a grid search on other influencing hyper-parameters, our method could gain the best performance with 0.095 EER, which is the state-of-the-art performance on the evaluation set of the challenge. For our future work, we aim to apply neural architecture search to larger audio datasets, such as DCASE 2019 Task1 and AudioSets.

References

[1] S. Souli and Z. Lachiri, "Audio sounds classification using scattering features and support vectors machines for medical surveillance," *Applied Acoustics*, vol. 130, pp. 270–282, 2018.

[2] M. Shah, B. Mears, C. Chakrabarti, and A. Spanias, "Lifelogging: Archival and retrieval of continuously recorded audio using wearable devices," in *Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on.* IEEE, 2012, pp. 99–102.

TABLE III
EXPERIMENTAL RESULTS OF NAS AND OTHER BASELINE MODELS ON THE EVALUATION SET OF THE DCASE 2016 DOMESTIC AUDIO TAGGING CHALLENGE (THE COEFFICIENT OF MIXUP IS SET TO 1.5, 2.0 AND 1.7 RESPECTIVELY).

Model	Data Augmentation	c	m	f	v	p	b	o	Avg
CRNN	mixup(α=1.5)	0.103	0.139	0.121	0.016	0.107	0.021	0.215	0.103
$NAS_{\alpha=1.5}$	mixup(α=1.5)	0.109	0.126	0.114	0.030	0.111	0.002	0.191	**0.098**
CRNN	mixup(α=2.0)	0.110	0.123	0.100	0.030	0.118	0.035	0.192	0.102
$NAS_{\alpha=1.5}$	mixup(α=2.0)	0.104	0.126	0.114	0.023	0.129	0.014	0.176	**0.098**
$NAS_{gridsearch}$	mixup(α=1.7)	0.109	0.126	0.114	0.025	0.118	0.005	0.176	**0.096**
NAS_{best}	mixup(α=1.5)	0.103	0.126	0.114	0.020	0.114	0.014	0.176	**0.095**

[3] P. T. Bauer, “Heart-activity monitoring with multi-axial audio detection,” Nov. 27 2007, uS Patent 7,302,290.

[4] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, “Attention and localization based on a deep convolutional recurrent model for weakly supervised audio tagging,” *arXiv preprint arXiv:1703.06052*, 2017.

[5] K. Xu, P. Roussel, T. G. Csapó, and B. Denby, “Convolutional neural network-based automatic classification of midsagittal tongue gestural targets using b-mode ultrasound images,” *The Journal of the Acoustical Society of America*, vol. 141, no. 6, pp. EL531–EL537, 2017.

[6] S. Wei, K. Xu, D. Wang, F. Liao, H. Wang, and Q. Kong, “Sample mixed-based data augmentation for domestic audio tagging,” *arXiv preprint arXiv:1808.03883*, 2018.

[7] S. Yun, S. Kim, S. Moon, J. Cho, and T. Kim, “Discriminative training of gmm parameters for audio scene classification and audio tagging,” *IEEE AASP Challenge Detect. Classification Acoust. Scenes Events*, 2016.

[8] Y. Xu, Q. Huang, W. Wang, P. J. Jackson, and M. D. Plumbley, “Fully dnn-based multi-label regression for audio tagging,” *arXiv preprint arXiv:1606.07695*, 2016.

[9] Q. Kong, I. Sobieraj, W. Wang, and M. Plumbley, “Deep neural network baseline for dcase challenge 2016,” *Proceedings of DCASE 2016*, 2016.

[10] Q. Kong, Y. Xu, W. Wang, and M. D. Plumbley, “Audio set classification with attention model: A probabilistic perspective,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 316–320.

[11] E. Cakır, T. Heittola, and T. Virtanen, “Domestic audio tagging with convolutional neural networks,” *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE 2016)*, 2016.

[12] T. Lidy and A. Schindler, “Cqt-based convolutional neural networks for audio scene classification,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, vol. 90. DCASE2016 Challenge, 2016, pp. 1032–1048.

[13] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” *arXiv preprint arXiv:1611.01578*, 2016.

[14] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” *arXiv preprint arXiv:1707.07012*, vol. 2, no. 6, 2017.

[15] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, “Efficient architecture search by network transformation.” AAAI, 2018.

[16] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, “Hierarchical representations for efficient architecture search,” *arXiv preprint arXiv:1711.00436*, 2017.

[17] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[18] http://www.cs.tut.fi/sgn/arg/dcase2016/challenge/.

[19] D. Feng, K. Xu, H. Mi, F. Liao, and Y. Zhou, “Sample dropout for audio scene classification using multi-scale dense connected convolutional neural network,” in *Pacific Rim Knowledge Acquisition Workshop*. Springer, 2018, pp. 114–123.

[20] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, “Detection and classification of acoustic scenes and events,” *IEEE Transactions on Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[21] http://www.cs.tut.fi/sgn/arg/dcase2017/challenge/.

[22] J. Ye, T. Kobayashi, M. Murakawa, and T. Higuchi, “Acoustic scene classification based on sound textures and events,” in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 1291–1294.

[23] K. Choi, G. Fazekas, and M. Sandler, “Automatic tagging using deep convolutional neural networks,” *arXiv preprint arXiv:1606.00298*, 2016.

[24] K. Xu, B. Zhu, Q. Kong, H. Mi, B. Ding, D. Wang, and H. Wang, “General audio tagging with ensembling convolutional neural networks and statistical features,” *The Journal of the Acoustical Society of America*, vol. 145, no. 6, pp. EL521–EL527, 2019.

[25] S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu, “Minimum risk training for neural machine translation,” *arXiv preprint arXiv:1512.02433*, 2015.

[26] S. Thrun and L. Pratt, *Learning to learn*. Springer Science & Business Media, 2012.

[27] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, “Learning to learn by gradient descent by gradient descent,” in *Advances in Neural Information Processing Systems*, 2016, pp. 3981–3989.

[28] K. Li and J. Malik, “Learning to optimize,” *arXiv preprint arXiv:1606.01885*, 2016.

[29] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, “Efficient neural architecture search via parameter sharing,” *arXiv preprint arXiv:1802.03268*, 2018.

[30] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin, “Large-scale evolution of image classifiers,” *arXiv preprint arXiv:1703.01041*, 2017.

[31] B. Deng, J. Yan, and D. Lin, “Peephole: Predicting network performance before training,” *arXiv preprint arXiv:1712.03351*, 2017.

[32] C. Liu, B. Zoph, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, “Progressive neural architecture search,” *arXiv preprint arXiv:1712.00559*, 2017.

[33] K. Xu, D. Feng, H. Mi, B. Zhu, D. Wang, L. Zhang, H. Cai, and S. Liu, “Mixup-based acoustic scene classification using multi-channel convolutional neural network,” in *Pacific Rim Conference on Multimedia*. Springer, 2018, pp. 14–23.

[34] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *arXiv preprint arXiv:1511.07289*, 2015.

[35] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[36] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[37] A. Mesaros, T. Heittola, and T. Virtanen, “Tut database for acoustic scene classification and sound event detection,” in *2016 24th European Signal Processing Conference (EUSIPCO)*. IEEE, 2016, pp. 1128–1132.

[38] H. Christensen, J. Barker, N. Ma, and P. D. Green, “The chime corpus: a resource and a challenge for computational hearing in multisource environments,” in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[39] P. Foster, S. Sigtia, S. Krstulovic, J. Barker, and M. D. Plumbley, “Chime-home: A dataset for sound source recognition in a domestic environment.” in *WASPAA*, 2015, pp. 1–5.

[40] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[41] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.