Neural Architecture Search on Acoustic Scene Classification

Jixiang Li[†], Chuming Liang[†], Bo Zhang, Zhao Wang, Fei Xiang, Xiangxiang Chu

Xiaomi AI Lab

{lijixiang, liangchuming1, zhangbo11, wangzhao3, xiangfei, chuxiangxiang}@xiaomi.com

Abstract

Convolutional neural networks are widely adopted in Acoustic Scene Classification (ASC) tasks, but they generally carry a heavy computational burden. In this work, we propose a high-performance yet lightweight baseline network inspired by MobileNetV2, which replaces square convolutional kernels with unidirectional ones to extract features alternately in temporal and frequency dimensions. Furthermore, we explore a dynamic architecture space built on the basis of the proposed baseline with the recent Neural Architecture Search (NAS) paradigm, which first train a supernet that incorporates all candidate architectures and then apply a well-known evolutionary algorithm NSGA-II to discover more efficient networks with higher accuracy and lower computational cost from the supernet. Experimental results demonstrate that our searched network is competent in ASC tasks, which achieves 90.3% F1-score on the DCASE2018 task 5 evaluation set, marking a new state-of-the-art performance while saving 25% of FLOPs compared to our baseline network.

Index Terms: acoustic scene classification, neural architecture search, deep learning, NSGA-II

1. Introduction

Acoustic Scene Classification (ASC) is an important task in the field of audio understanding and analysis, which classifies an audio stream into one of the predefined acoustic scenes. Recently, ASC has drawn increased attention from academy to industry, due to its great potential in many applications like context aware devices [1], acoustic monitoring [2], and assistive technologies [3]. In the early stages, many traditional machine learning methods such as GMM [4], HMM [1] and SVM [5] have been applied. Now with the great success of deep learning in computer vision and the availability of larger audio datasets, methods based on DNN [6], CNN [7] and RNN [8] have gradually been dominant in ASC. Among them, CNN-based methods have obtained the most state-of-the-art results because of their excellent capability of learning high-level features by exploiting time-frequency pattern of a signal [9]. For example, Inoue [10] ensembled four CNN models that championed in DCASE 2018 task 5 [11]. Hershey [12] compared different CNN architectures (VGG [13], Xception [14] and ResNet [15] etc.) in an audio classification task and all of them showed promising results. However, in vision tasks, most architectures above have been outperformed by advanced networks such as MobileNetV2 [16] in terms of the number of parameters and computational cost.

To automatically get higher-performance networks than those designed by human experts, neural architecture search (NAS) has been adopted in many vision and NLP tasks, including our previous work on image super-resolution and image classification [17, 18, 19]. A few have applied NAS in speech like keyword spotting [20, 21]. We are thus motivated to validate NAS applicability in ASC task. Early NAS methods based on reinforcement learning [22] and evolutionary algorithms [23] consume immense GPU resources to evaluate thousands of neural networks. Recent gradient-based methods [24] require only a small amount of GPUs, but it is less robust and has weak reproducibility. On the contrary, the one-shot [25] paradigm is more promising mainly for three reasons: robustness to generate powerful architectures, moderate consumption of GPU resources, and convenience for solving multi-objective problems (MOPs, e.g. searching for a model with a trade-off between accuracy and computational cost). Generally, a supernet that comprises of all candidate architectures is constructed and trained fully on a target dataset. Next, by inheriting weights directly from the supernet, candidates can have a good performance, from which we can judge competitive ones from the rest. Beyond [25], single-path one-shot [26] and FairNAS [27] also exhibit improved training stability and more accurate ranking.

In this paper, we propose a lightweight network of high-performance inspired by MobileNetV2's inverted bottleneck block. On this basis, we also search for better architectures with the one-shot NAS doctrine. Many previous works in vision used CNNs as a feature extractor, mostly with square k×k kernels since the information has the same nature in the spatial resolution. For ASC's mel-spectrogram, however, horizontal and vertical information has different implications, namely, temporal relation and frequency distribution. Square kernels are thus less appropriate. Based on the above consideration, we adapt MobileNetV2 with unidirectional (1×k and k×1) kernels to separately handle each dimension's information. Experiments in section 3.3 demonstrate the proposed unidirectional convolution network has outstanding performance in DCASE 2018 Task 5 [11]. Further, we leverage fair supernet training strategy [27] and NSGA-II algorithm [28] to search for network architectures with higher accuracy and less computation. In section 3.4 we illustrate the details of the searched network that achieve state-of-the-art results on DCASE 2018 task 5.

2. Acoustic Scene Classification with Neural Architecture Search

2.1. Proposed Baseline Network

In view of MobileNetV2's recent success in speech applications [29, 30], we incorporate the MobileNetV2's bottleneck blocks [16] as the core components in our baseline network. Our baseline mainly consists of three parts, namely a feature extractor (FE) module, a two-level gated recurrent unit (GRU) layer and two fully-connected (FC) layers. Specifically, the FE module comprises one stem convolution layer, 20 bottleneck blocks with 1×1-depthwise-1×1 convolution structure and one global frequency-dimensional convolution layer. We use unidirectional (1×k and k×1) kernels to replace original 3×3 kernels
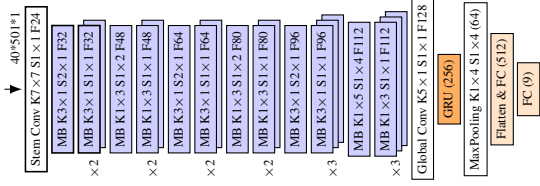

Figure 1: *The proposed baseline adapted from MobileNetV2 bottleneck (MB) for ASC. Note K and S refer to kernel size and stride. F means filters. The expansion rate of MB is 6.*

in depthwise convolution of [16]. The configuration of this architecture is shown in Figure 1. This FE module design is based on the following considerations. First, the residual property [15] of bottleneck block helps backpropagation and prevents vanishing gradients. Second, unidirectional kernels are utilized to extract features more attentively in temporal dimension and frequency dimensions. Besides, they reduce the number of parameters to prevent overfitting. Last, extracting features alternately in temporal and frequency dimensions is conducive to information flow and fusion.

2.2. Neural Architecture Search

Based on our baseline network, we utilize NAS methodology to search for architectures with better performance. Concretely, we let our search targets be the kernel sizes and expansion rates in each inverted bottleneck block, which sums up to an enormous search space. Our supernet thus consists of this searchable feature extractor along with the original GRU and FCs. We then train the supernet on the target dataset to obtain an evaluator for a good ranking among candidate models. Next, we use NSGA-II [28], an advanced evolutionary algorithm that features multi-objective optimization, to seek promising architectures with higher accuracy and less computation. Final competent architectures are selected from the Pareto-front obtained by NSGA-II and are trained from scratch. The overall NAS pipeline is illustrated in Figure 2.
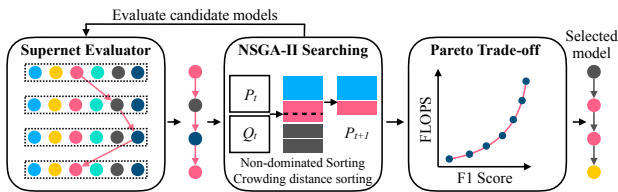

Figure 2: *The NAS pipeline consists of supernet training [27] and NSGA-II Searching [28].*

2.2.1. Search Space

We now describe the details of the search space for the FE module, where a certain number of bottleneck blocks with different settings can be chosen in each layer. We allow an expansion rate in $\{3, 6\}$ for the first 1×1 convolution and a kernel size in $\{3, 5, 7\}$ for the depthwise convolution in each block as shown in Figure 3. For layer 17, we make an exception due to its large downsampling rate, where its expansion rate ranges in $\{3, 6, 8\}$ and its kernel size in $\{5, 7\}$. The output filters of each block and the downsampling positions are fixed as in the baseline. In total, we have 20 searchable layers in the FE module, each with 6 choices of MB blocks. Hence, our search space contains $6^{20} \approx 10^{15}$ possible architectures. It is very hard to search for better architectures from such an enormous search space simply by trial-and-error. For convenience's sake, we refer the c-th

choice block B in layer l as $B_l^{(c)}$, where $l \in \{1, 2, ..., 20\}$ and $c \in \{1, 2, ..., 6\}$.
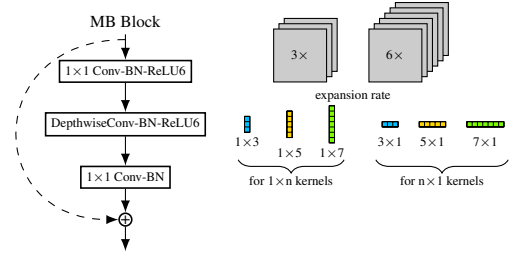

Figure 3: *The searchable MobileNetV2's bottleneck blocks.*

2.2.2. Supernet Training Strategy

We construct a supernet with the above searchable FE module, the original GRU and FCs together. To train the supernet, we use the same fairness strategy proposed in our previous work [27]. Specifically, given a mini-batch of training data, instead of training the supernet as a whole, we *uniformly sample without replacement* to have 6 models with no shared blocks for separate training, i.e. in the *first* step we **randomly** select one choice from the 6 choices in each search layer to construct a model and the gradients of all parameters in this model are calculated by back-propagation. In the same way, in the *second* step we **randomly** pick one choice from the **remaining unselected** choices in each layer to get another model and calculate its gradients. The rest may be deduced by analogy. After the *sixth* step, **all** choices in each layer have been selected and their gradients are obtained. Finally, we update the trainable parameters of the supernet according to the corresponding gradients altogether. The training algorithm is detailed in Algorithm 1. Through this method, each choice has the same opportunity to update itself using every mini-batch of data, which stabilizes the training process.

The trained supernet is then used to evaluate candidate models. In particular, we sample a candidate model from the supernet with its trained weights, and evaluate it on the validation set. In this way, we can quickly get the approximate performance of each candidate model in the supernet without extra training. As proved in our previous work [27], the performance ranking among models obtained through this fair supernet training strategy is highly consistent with those trained from scratch.

2.2.3. Search Strategy

There are many ways to search, such as Random Search (RS), reinforcement learning (RL), and evolutionary algorithms (EA). Here we utilize an efficient evolutionary algorithm NSGA-II [28] to search for promising architectures in the enormous search space. We adapt NSGA-II to our needs and only describe the differences in this section. Please refer to its original paper for the rest details (non-dominated sorting, crowding distance, tournament selection, and Pareto-front, etc.). In this paper, an architecture is regarded as an *individual* and encoded uniquely by one *chromosome*. According to the illustration in Section 2.2.1, we define the chromosome of an architecture as a list containing 20 elements corresponding to 20 searchable layers. Each *gene* in the chromosome is an integer ranges from 1 to 6 corresponding to 6 choices. Since we aim to search for architectures with higher accuracy and lower computation, we set the accuracy metric and computational cost as two *objectives*.

Algorithm 1 Supernet Training Strategy
___
Input: training data loader D_{train}, the number L of search layers, the number C of choice blocks each layer, the number I of training epochs, choice set $\Phi_l = \{1, 2, 3, ..., C\}$ for each search layer $l \in \{1, 2, ..., 20\}$
Output: the supernet with trained parameters
for $i = 1$ **to** I **do**
  for data d, ground-truth g in D_{train} **do**
    clear gradients of all supernet parameters
    for $c = 1$ **to** C **do**
      initialize $\Psi_l = \Phi_l$ for each search layer if c == 1
      for $l = 1$ **to** L **do**
        randomly select an element e from Ψ_l
        delete e from Ψ_l
        get the choice $B_l^{(e)}$ as B_l
      end for
      construct $model_c$ by $(B_1, B_2, ..., B_L)$
      calculate gradients ∇_c of $model_c$ parameters by d, g
    end for
    update all trainable parameters of the supernet by gradients $\nabla_1, \nabla_2, ..., \nabla_C$
  end for
end for
___

Algorithm 2 Search Strategy
___
Input: population size P, max iteration I, trained supernet weights W, mutation m, crossover c, tournament selection *ts* [28], objective1 *obj1* for accuracy on validation set, objective2 *obj2* for FLOPs, non-dominated sorting and crowding distance sorting *nds-cds* [28]
Output: the best population list
initialize best population list $bp = \varnothing$
for $i = 1$ **to** I **do**
  chromosome list $cl = \varnothing$, population list $p = \varnothing$
  update exploitation ratio α by Eq. 1
  $cl \Leftarrow \alpha \times P$ chromosomes by creating randomly
  $cl \Leftarrow (1-\alpha) \times P$ chromosomes by $m(c(bp, ts))$
  for each chromosome c **in** cl **do**
    construct *model* by c and W
    accuracy = *obj1*(*model*)
    flops = *obj2*(*model*)
    restore $(c, accuracy, flops)$ into p
  end for
  $p \Leftarrow bp \cup p$
  $bp \Leftarrow$ Top$P$ Pareto-optimal individuals by *nds-cds* on p
end for
___

A *population* of P=64 architectures are chosen for evolution which iterates for I=70 of generations. For crossover, we get two *champions* (the architectures who defeat each opponent's architecture on *objectives*) using tournament selection, then select two gene spots and swap genes at corresponding spots on these two champion chromosomes. As for mutation, we select one to four gene spots randomly and change gene values on these spots. In the early stage of the search, we set exploration ratio 100% to explore enormous search space to find promising architectures by creating chromosomes randomly. As the search progressing, exploitation ratio α, i.e. the number of individuals created by crossover and mutation / P, increases to 80% gradually with the exploration ratio (1-α) decreasing to 20%, which pays more attention to searching for better ones near the searched promising architectures. The exploitation ratio α is defined as

$$\begin{cases} 0, & i < 15, \\ (i - 15)/68.75, & 15 \leq i \leq 70 \end{cases} \quad (1)$$

where i refers to the evolutionary iteration. The whole search algorithm is shown in Algorithm 2.

3. Experiments and Analysis

3.1. Dataset and Data Augmentation

We evaluate our models on the task 5 dataset of DCASE 2018 Challenge [11] containing audios of 9 domestic activities. The whole dataset is divided into the development set (72984 segments) and the evaluation set (72972 segments). Each segment has four acoustic channels. We extract 40 log-mel band energies for each channel signal with a frame size of 40ms and hop size of 20ms, that gives us 40×501 data matrix for each sample.

For data augmentation, we adopt the shuffling and mixing as in [10] offline because of the highly imbalanced class distribution of the dataset. In particular, we increase the number of segments of minority classes (cooking, dishwashing, eating, other, social activity and vacuum cleaning) to 14K and the rest

to 28K. Besides, Cutout [31] is used online when training to improve the regularization of models.

3.2. Training and Evaluation Setup

We train our models using Adam optimizer [32] with a weight decay of 1e-6 on 2 GPUs. We warm up the learning rate from 0 to 0.003 in the first three epochs and keep this learning rate in subsequent epochs. An exponential moving average with decay 0.9986 is used. The batch size is 192.

We evaluate our models on the entire evaluation set in two ways: **single**-mode and **ensemble**-mode. For single-mode, we just train one single model on augmented development set for 10 epochs, then evaluate it using the last checkpoint weights. For ensemble-mode, we use the official cross-validation folds [11] and train four models on these four folds, respectively. As for scoring, we calculate macro-F1 score [11] of each channel signal and then average them for the final score.

3.3. Experiments on Baseline Network

As an outperforming feature extractor, VGG [13] architecture is widely used by previous works like [33] [34] [35] that achieve good performance, but its size and depth also bring the problem of high computational cost. As a comparison, we replace the FE module with original VGG16 while the experimental settings remain the same as our baseline to explore the relationship between the model capacity and the ability to extract features. In this section, we train our models in single-mode.

Table 1: *Performance comparisons with different feature extractor and input size.*

Model	Input (H, W)	Params (M)	FLOPs (G)	F1 Score (%)
[35]-ensemble	(64, 1250)	26.65	17.98	89.1
BaselineVGG	**(40, 501)**	17.24	5.99	89.4
Baseline	**(40, 501)**	**3.31**	**2.03**	**89.8**

We can see from Table 1 that our baseline model is 0.4%

F1 higher than the comparison model which just replaces the FE with original VGG16, while reducing 4G FLOPs and 14M parameters. It suggests that capacity and depth of model have no absolute relationship with the ability to extract features and demonstrates our proposed architecture with unidirectional kernels alternately in temporal dimension and frequency dimensions is efficient for the ASC task. Moreover, our baseline, as a single model with smaller input size, also outperforms [35] which based on VGG architecture and ensembled by four models. The experiments in this section show that our proposed FE architecture can be used as a light-weight and high-performance feature extractor.

3.4. Experiments on NAS

3.4.1. Search Result

In this section, we divide each category of the development set into a training set and a validation set in a 70%:30% ratio and augment the training set based on the strategy in section 3.1. We train our supernet for 10 epochs on the augmented training set using section 2.2.2 training strategy, then evaluate more than 4.4K candidate models on the validation set by the section 2.2.3 method. As a comparison, we also evaluate the same amount of candidates by RS instead of NSGA-II.
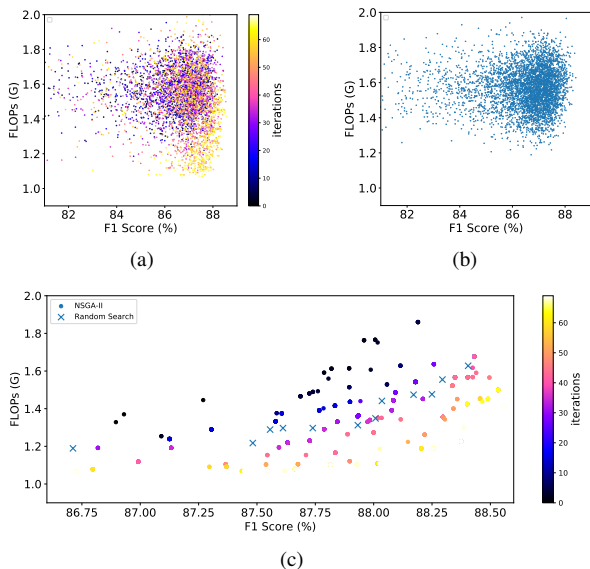

Figure 4: *(a) Models explored by NSGA-II (b) Random Search (c) Pareto-front of both methods.*

From Figure 4 (a) we can see that in the early stage some promising models and lots of mediocre models are explored because of the large exploration rate. As the evolution iterates, the exploitation rate increases gradually, so that models near the searched promising models will be searched by crossover and mutation. In the later period of the evolution, due to the large exploitation ratio, more attention is invested in crossover and mutation so that more promising models are found. The Figure 4 (b) clearly shows that the search distribution of RS only almost coincides with the **early** search area of NSGA-II, suggesting that RS is relatively weak on problems with multi-objective constraints. The Figure 4 (c) shows the Pareto optimal models searched by NSGA-II and RS, which demonstrates that NSGA-II is more powerful.
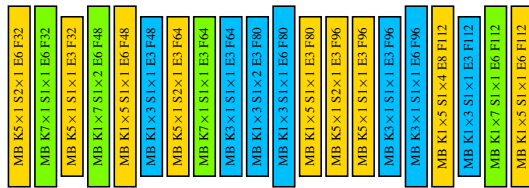

Figure 5: *The searched part of the feature extractor module. Note E denotes the expansion rate.*

3.4.2. Performance on DCASE 2018 Task 5

Table 2: *Class-wise performance comparison. Note **B-s** denotes the single baseline model, **N-s** denotes the single NASC-net model, **N-e** denotes the ensembled NASC-net model.*

Class	[11][1]	[10][1]	[35]	B-s	N-s	N-e
Absence	88.7	**94.0**	92.7	93.5	93.1	93.9
Cooking	94.9	94.5	93.8	95.7	**96.7**	96.1
Dishwashing	78.5	87.6	86.6	88.5	**89.2**	88.9
Eating	81.7	**88.7**	88.0	88.0	88.5	**88.7**
Other	40.2	57.3	58.8	58.6	59.3	**60.1**
Social activity	96.5	97.2	**97.9**	97.4	97.0	97.8
Vacuum clean	95.9	97.2	95.3	97.1	97.4	**97.7**
Watching TV	99.9	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Working	81.5	89.4	88.4	89.3	88.9	**89.7**
F1 score	84.2	89.5	89.1	89.8	90.0	**90.3**

Due to space limitation, we only select one searched model which has 1.53G FLOPs and 3.01M parameters (saving 25% of FLOPs and 9% of parameters compared to the baseline) on the Pareto-front of the 70-th iteration, though we can give lots of promising models after only one search process. We name this model **NASC-net** and the searched part of this model is illustrated in Figure 5. NASC-net looks very weird in architectural perspective and it's almost impossible for human experts to design such an architecture, so we select it as a typical searched model and train it from scratch in single and ensemble mode, respectively. The performances on the evaluation set of DCASE 2018 task 5 is shown in Table 2. NASC-net outperforms the baseline model in terms of macro F1 score, which is also more light-weight than those expert-designed. Besides, an ensemble of NASC-net uplifts F1-score further to 90.3% at a $4\times$ cost.

4. Conclusions

In this paper, we present a novel and efficient network for ASC tasks where its feature extractor is inspired by MobileNetV2. We show that our proposed network can achieve both high performance and low computation. Besides, on the basis of the proposed network, we apply neural architecture search to achieve a more sophisticated architecture using the fairness supernet training strategy and NSGA-II algorithm. Our searched network obtains a new state of the art on the DCASE2018 Task5 dataset with much lower computation. We can conclude that NAS is applicable in the field of ASC and potentially in other acoustics domains.

---

[1]Entire Evaluation Set of DCASE2018 task5 was divided into unknown microphone set (0.4286=3/7) and known microphone set (0.5714=4/7) for each category. Based on the ratio of the two sets, we can calculate the F1-score for entire Evaluation Set based on the official published leaderboard, e.g. the F1-score of category "Absence" in [11] is 88.7%=87.7%×0.4286+89.4%×0.5714.

5. References

[1] A. J. Eronen, V. T. Peltonen, J. T. Tuomi, A. P. Klapuri, S. Fager-lund, T. Sorsa, G. Lorho, and J. Huopaniemi, "Audio-based context recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 321–329, 2005.

[2] S. Ntalampiras, I. Potamitis, and N. Fakotakis, "On acoustic surveillance of hazardous situations," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 165–168.

[3] M. Bugalho, J. Portelo, I. Trancoso, T. Pellegrini, and A. Abad, "Detecting audio events for semantic video search," in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[4] 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.

[5] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: Outcome of the dcase 2016 challenge," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 26, no. 2, pp. 379–393, 2018.

[6] S. Mun, S. Shon, W. Kim, D. K. Han, and H. Ko, "Deep neural network based learning and transferring mid-level audio features for acoustic scene classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 796–800.

[7] Y. Han and K. Lee, "Acoustic scene classification using convolutional neural network and multiple-width frequency-delta data augmentation," *arXiv preprint arXiv:1607.02383*, 2016.

[8] T. H. Vu and J.-C. Wang, "Acoustic scene and event recognition using recurrent neural networks," *Detection and Classification of Acoustic Scenes and Events*, vol. 2016, 2016.

[9] A. Mesaros, T. Heittola, and T. Virtanen, "Acoustic scene classification: An overview of dcase 2017 challenge entries," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2018, pp. 411–415.

[10] T. Inoue, P. Vinayavekhin, S. Wang, D. Wood, N. Greco, and R. Tachibana, "Domestic activities classification based on cnn using shuffling and mixing data augmentation," *DCASE 2018 Challenge*, 2018.

[11] G. Dekkers, L. Vuegen, T. van Waterschoot, B. Vanrumste, and P. Karsmakers, "DCASE 2018 Challenge - Task 5: Monitoring of domestic activities based on multi-channel acoustics," KU Leuven, Tech. Rep., 2018. [Online]. Available: https://arxiv.org/abs/1807.11246

[12] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "Cnn architectures for large-scale audio classification," in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2017, pp. 131–135.

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

[14] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[17] X. Chu, B. Zhang, R. Xu, and H. Ma, "Multi-Objective Reinforced Evolution in Mobile Neural Architecture Search," *arXiv preprint. arXiv:1901.01074*, 2019.

[18] X. Chu, B. Zhang, H. Ma, R. Xu, J. Li, and Q. Li, "Fast, accurate and lightweight super-resolution with neural architecture search," *arXiv preprint arXiv:1901.07261*, 2019.

[19] X. Chu, B. Zhang, and R. Xu, "Moga: Searching beyond mobilenetv3," in *International Conference on Acoustics, Speech, and Signal Processing*, 2020. [Online]. Available: https://arxiv.org/pdf/1908.01314.pdf

[20] T. Véniat, O. Schwander, and L. Denoyer, "Stochastic Adaptive Neural Architecture Search for Keyword Spotting," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2842–2846.

[21] H. Mazzawi, X. Gonzalvo, A. Kracun, P. Sridhar, N. Subrahmanya, I. L. Moreno, H. J. Park, and P. Violette, "Improving Keyword Spotting and Language Identification via Neural Architecture Search at Scale," in *Interspeech*, 2019, pp. 1278–1282.

[22] B. Zoph, V. Vasudevan, J. Shlens, and Q. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.

[23] E. Real, A. Aggarwal, Y. Huang, and Q. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 4780–4789.

[24] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *International Conference on Learning Representations*, 2019.

[25] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and Simplifying One-Shot Architecture Search," in *International Conference on Machine Learning*, 2018, pp. 549–558.

[26] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single Path One-Shot Neural Architecture Search with Uniform Sampling," *arXiv preprint. arXiv:1904.00420*, 2019.

[27] X. Chu, B. Zhang, R. Xu, and J. Li, "FairNAS: Rethinking Evaluation Fairness of Weight Sharing Neural Architecture Search," *arXiv preprint. arXiv:1907.01845*, 2019.

[28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[29] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, pp. 1256–1266, 2019.

[30] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, and Y. Zhang, "Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions," *arXiv preprint arXiv:1910.10261*, 2019.

[31] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[32] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2014.

[33] Y. Han, J. Park, and K. Lee, "Convolutional neural networks with binaural representations and background subtraction for acoustic scene classification," *DCASE 2017 Challenge*, 2017.

[34] R. Tanabe, T. Endo, Y. Nikaido, T. Ichige, P. Nguyen, Y. Kawaguchi, and K. Hamada, "Multichannel acoustic scene classification by blind dereverberation, blind source separation, data augmentation, and model ensembling," in *DCASE 2018 Challenge*, 2018.

[35] W. Wang, W. Wang, M. Sun, and C. Wang, "Acoustic scene analysis with multi-head attention networks," *arXiv preprint arXiv:1909.08961*, 2019.