AutoSpeech: Neural Architecture Search for Speaker Recognition

Shaojin Ding[1]*, Tianlong Chen[2]*, Xinyu Gong[1,2], Weiwei Zha[3], Zhangyang Wang[2]

[1]Department of Computer Science and Engineering, Texas A&M University, USA
[2]Department of Electrical and Computer Engineering, The University of Texas at Austin, USA
[3]School of Software Engineering, University of Science and Technology of China, China

shjd@tamu.edu, {tianlong.chen, xinyu.gong, atlaswang}@utexas.edu,
Sa517009@mail.ustc.edu.cn

Abstract

Speaker recognition systems based on Convolutional Neural Networks (CNNs) are often built with off-the-shelf backbones such as VGG-Net or ResNet. However, these backbones were originally proposed for image classification, and therefore may not be naturally fit for speaker recognition. Due to the prohibitive complexity of manually exploring the design space, we propose the first neural architecture search approach for the speaker recognition tasks, named as **AutoSpeech**. Our algorithm first identifies the optimal operation combination in a neural cell and then derives a CNN model by stacking the neural cell for multiple times. The final speaker recognition model can be obtained by training the derived CNN model through the standard scheme. To evaluate the proposed approach, we conduct experiments on both speaker identification and speaker verification tasks using the VoxCeleb1 dataset. Results demonstrate that the derived CNN architectures from the proposed approach significantly outperform current speaker recognition systems based on VGG-M, ResNet-18, and ResNet-34 backbones, while enjoying lower model complexity.

Index Terms: speaker recognition, neural architecture search

1. Introduction

Speaker recognition aims to retrieve the identity of the speaker given his/her utterances. According to the content similarity of the utterances, speaker recognition can be categorized into text-dependent and text-independent, the latter being more general and realistic for practical applications. Additionally, according to different application settings, speaker recognition usually fall into one of the two categories: speaker identification (SID) and speaker verification (SV). SID identifies the speaker of an utterance from a known speaker set, and SV determines if the speaker of an utterance matches its given enrollment.

End-to-end speaker recognition systems [1, 2, 3, 4, 5, 6] have emerged in recent years and achieved the state-of-the-art performance. These models usually admit a *three-stage pipeline*: (1) A deep neural network, typically Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), as a feature extractor to generate frame-wise speaker embedding; (2) a temporal aggregation layer, to produce fixed length speaker embedding (i.e., d-vector); (3) A loss function to optimize the entire network. In testing stage, the network along with the temporal aggregation layer are first used to produce d-vectors for the testing utterances, and then a similarity metric (e.g., cosine similarity) follows to generate the final same/different speaker decision based on the d-vectors.

Most CNN/RNN-based speaker recognition works focus on improving the effectiveness of the speaker embedding through advanced training objectives and temporal aggregation strategies [7, 8, 3, 9, 5]. By contrast, network architecture design has received relatively less attention. Existing studies usually use off-the-shelf backbones that have been shown to be successful in standard tasks such as image classification (e.g., VGG-Net [10], ResNet [11]). However, these backbones are not designed for and may not be optimal for speaker recognition. Meanwhile, in other speech processing tasks such as speech recognition and speech synthesis, the improved design of network architecture has already yielded evident performance gains [12, 13]. In view of those, we conjecture that **enhancing network architecture design matters for improving speaker recognition performance too**. Yet as the major hurdle for improving the backbone of any new task, manually exploring the gigantic design space of deep networks is notoriously tedious and ad-hoc.

This paper proposes an automated approach to identify the optimal CNN architecture for text-independent speaker recognition, named as **AutoSpeech**. It is inspired by recent advances on neural architecture search (NAS) [14, 15, 16, 17, 18], which has proven success in designing deep networks that outperform hand-designed best performers in various tasks [19, 20, 21, 22]. We evaluated the effectiveness of AutoSpeech on both SID and SV tasks using VoxCeleb1 dataset [23]. Our derived CNN architectures significantly outperforms several networks used in state-of-the-art speaker recognition systems (based on VGG-M, ResNet-18, and ResNet-34), at lower model complexities.

2. Literature Review

2.1. Speaker Recognition

Earlier speaker recognition systems used spectral features (e.g., MFCCs) as the speaker embedding [24, 25], but these systems have been superseded by systems based on "i-vectors" [26, 27, 28]. An i-vector takes speech from a speaker and uses it to adapt a speaker-independent Gaussian Mixture Model (GMM, referred to as a Universal Background Model). The means of the adapted GMM are concatenated to form a supervector, which is then reduced in dimensionality using joint factor analysis. More recently, end-to-end speaker recognition [1, 2, 3, 4, 5] systems have been shown to surpass the i-vector based systems and achieve state-of-the-art performance. Variani *et al.* first proposed a neural network based speaker recognition system. In their work, they used maxout fully-connected network to produce d-vectors, and then they used cosine similarity to make the final decision. Following this, advanced network architectures such as CNNs [3, 5, 9, 8, 29] and RNNs [5, 2] are used for feature extracts. Additionally, advanced training objectives [2, 8, 29] and temporal aggregation strategies [7, 3, 9, 5] are also proposed to improve speaker recognition performance.
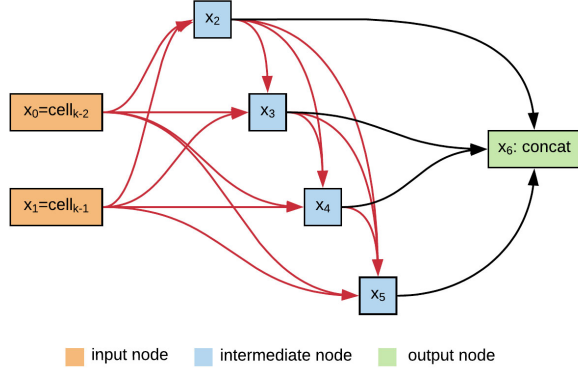
---

* Equal contribution.

Figure 1: *Illustration of a neural cell. During the search process, the intermediate nodes (x_2 to x_5) are densely connected (red arrows), and the optimal combination of the operations on these edges are found at the end of the searching process. During architecture derivation, only the two operations with highest softmax probabilities (excepting zero operation) are retained for each intermediate node.*

2.2. Neural Architecture Search

The goal of neural architecture search is to search for an optimal network architecture for a given task. To accomplish this, a search space will be constructed firstly, which may vary across different tasks and objectives. Towards accurate image classification, many previous work [14, 15, 16] adopt a cell-based search space with complicated inner connection. To improve its efficiency, [30, 31, 32] adopts a sequential-based search space, benefiting a lot in reducing latency and FLOPs inherently. More recently, neural architecture search has also been applied to other tasks (e.g., image segmentation [20, 33], generative adversarial network [21], and phone recognition [19]), where task-specific search spaces are usually designed. To conduct architecture search over the defined search space, various optimization methods are proposed. Barret and Quoc first use the reinforcement learning (RL) to search for architecture[17], where a RNN serves as an architecture sampler. However, it is quite time-consuming (2,000 GPU days). To address this limitation, Hieu *et al.* propose to use weight sharing (ENAS[16]) to accelerate RL-based search method. Furthermore, Liu *et al.* present a gradient-based method [14] to accelerate the searching process.

3. Methods

In this section, we introduce how to use NAS to automatically find the optimal CNN architecture for speaker recognition. Following previous NAS methods [14, 17], we use a *two-stage pipeline*. First, we search for the optimal architecture of a neural cell that consists of several different types of operations. Second, we derive a CNN model using the searched cell, and then train the CNN model through the standard speaker recognition training scheme. We first define the search space of a neural cell in Section 3.1. Then, we describe the NAS algorithm in Section 3.2. Finally, we introduce how to derive a CNN model using searched neural cells and how to train it, in Section 3.3.

3.1. Neural cell and search space

During the NAS process, the network is constructed by stacking multiple neural cells. We define a neural cell to have $N = 7$

Algorithm 1 Neural architecture search algorithm

Inputs: Training set D_{train}, validation set D_{val}.
Output: Searched neural cell architecture.

---

while not converge **do**
  - Update weight parameters ω by $\nabla_\omega L_{train}(\omega, \alpha)$ on D_{train}.
  - Update architecture parameters α by $\nabla_\alpha L_{val}(\omega, \alpha)$ on D_{val}.
end while
- Derive the neural cell architecture from α by retaining the two operations with highest softmax probabilities for each node.

---

nodes, as illustrated in Figure 1. A cell can be viewed as a directed acyclic graph, where a node x_i corresponds to a tensor and a directed edge (i, j) corresponds to an operation $o_{ij}(\cdot)$. Following [14, 17, 15, 34], we set each cell to have two input nodes, four intermediate nodes, and one output node. The input nodes are set equal to the outputs of previous two cells, respectively (e.g., The first node x_0 in the k-th cell is equal to the output of the (k-2)-th cell, and the second node x_1 in the k-th cell is equal to the output of the (k-1)-th cell). The four intermediate nodes (x_2 to x_5) are densely connected (red arrows in Figure 1, 14 edges in total), and each of them is computed as the summation based on all of its predecessors:

$$x_j := \sum_{i<j} o_{ij}(x_i) \qquad (1)$$

A best combination of the operations on these edges are found at the end of the searching process. Finally, the output node x_7 is defined as the concatenation of all intermediate nodes.

The search space O defines all possible candidate operations in a neural cell. We include the following 8 operations that are prevalent in modern CNNs in our search space, and the resulting search space size is 8^{14} (14 edges, 8 candidate operations for each edge).

- 3×3 separable convolution
- 3×3 average pooling
- 5×5 separable convolution
- 3×3 max pooling
- 3×3 dilated convolution
- skip connection
- 5×5 dilated convolution
- no connection (zero)

We stack the neural cell 8 times to form the backbone CNN during the NAS process. Additionally, we define two types of neural cells: *normal cells* (cells that keep the spatial resolution of the feature tensor) and *reduction cells* (cells that divide the spatial resolution by 2 and multiply the number of filters by 2). Following previous studies [35, 14, 15], we set the cells whose position is located at the $1/3$ and $2/3$ of the total depth to reduction cells, and the other cells are normal cells. All the normal cells share the same architecture, and all the reduction cells share the same architecture, respectively. The output of the last cell is then fed to an average pooling layer, followed by a fully-connected layer that outputs the softmax probability.

3.2. Neural architecture search

To search for the optimal operation combinations in a neural cell, we define two sets of parameters: (1) a set of *architecture* parameters α that controls the choice of operations and (2) a set of *weight* parameters ω of all operations in O. We use architecture parameters $\alpha_{ij} \in \mathbb{R}^{|O|}$ to relax the categorical choice

of a particular operation o_{ij} on edge (i, j) to a softmax over all possible operations in the search space:

$$\hat{o}_{ij}(x_i) := \sum_{o \in O} \frac{\exp(\alpha_{ij}^o)}{\sum_{o' \in O} \exp(\alpha_{ij}^{o'})} o(x_i) \qquad (2)$$

As a result, the search space becomes continuous, and NAS can be realized by optimizing the architecture parameters. In addition, since we have two types of neural cells (normal cells and reduction cells), the architecture parameters becomes $\alpha = (\alpha_{normal}, \alpha_{reduction})$, where α_{normal} are shared among all normal cells and $\alpha_{reduction}$ shared among all reduction cells.

We use a differentiable NAS algorithm [14] to learn α and ω jointly through back-propagation. We denote the training loss as $L_{train}(\omega, \alpha)$ and the validation loss as $L_{val}(\omega, \alpha)$. The NAS process can be viewed as a bi-level optimization problem [36] (eq. 3), which aims to find an optimal α that minimizes $L_{validation}(\omega, \alpha)$, where the optimal ω is determined by minimizing $L_{train}(\omega, \alpha)$.

$$\min_{\alpha} L_{val}(\omega, \alpha) \qquad (3)$$
$$\text{s.t. } \omega = \arg\min_{\omega} L_{train}(\omega, \alpha)$$

In our task, we use cross-entropy loss for both L_{train} and L_{val}:

$$L := -\sum_{k=1}^{K} \mathbb{I}(y = k) \log p_k \qquad (4)$$

where $\mathbb{I}(\cdot)$ is the indicator function, K is the number of speakers, y is the ground-truth speaker, and p_k is the softmax probability of speaker k.

In practice, this bi-level optimization problem can be solved iteratively, as shown in Algorithm 1. During each iteration, we perform two steps: weight parameters update and architecture parameters update. In the weight parameters update step, we update ω by minimizing the the cross-entropy loss on training set L_{train}, while fixing α. In the architecture parameter update step, we update α by minimizing the cross-entropy loss on validation set L_{val}, while fixing ω.

The algorithm terminates when the choice of operations in the neural cell converges, which is empirically measured by the entropy of the architecture parameters α as:

$$E = \sum_{i,j} \sum_{o \in O} \alpha_{ij}^o \log \alpha_{ij}^o \qquad (5)$$

A smaller the entropy indicates a higher confidence of choosing a particular operation among all possible operations. As a result, we consider the architecture of the neural cell being converged when the entropy does not decrease.

3.3. Architecture derivation

Once the search process is converged, we can derive a neural cell architecture from the learned architecture parameters. For each node x_j, we retain *two operations with highest softmax probabilities* (excepting zero operation) from all its predecessors $x_i, \forall i < j$, following [14]. The softmax probability of an operation o between nodes (i, j) is defined as:

$$p_{ij}^o := \frac{\exp(\alpha_{ij}^o)}{\sum_{o' \in O} \exp(\alpha_{ij}^{o'})} \qquad (6)$$

With the derived neural cell, we construct a CNN by stacking multiple neural cells. We explore different number of neural cells and different number of channels in our experiments, as we will describe in Section 4.3. Similar to the CNN architecture in NAS process, we still set the cells whose position is located at the 1/3 and 2/3 of the total depth to reduction cells, and the other cells are normal cells.

To obtain the final speaker recognition model, we train the CNN model from random initialized weights (denoted as *training from scratch*) on training set by minimizing the cross-entropy loss in eq. 4. We use the output of the average pooling layer of the trained CNN as the speaker embedding.

4. Experiments

4.1. Datasets

We conducted experiments on VoxCeleb1 [23] dataset consisting of 153,516 utterances produced by 1,251 speakers. VoxCeleb1 has an identification split and a verification split for SID and SV, respectively. The identification split divides the entire dataset to a training set of 138,316 utterances, a development set of 6,904 utterances, and a test set of 8,251 utterances. All the three sets were produced by the same 1,251 speakers. The verification split divides the dataset into a training set of 148,642 utterances produced by 1,211 speakers and a test set of 4,874 utterances produced by 40 speakers. The test speakers do not overlap with training. The verification split also provided 37,720 trial pairs for SV evaluation. During the NAS process, we followed the identification split. These subsets were used as introduced in Section 3.2. During training from scratch, we train different models for SID and SV tasks following identification split and verification split, respectively.

4.2. Implementation details

For each utterance, we extracted a 257-dim spectrogram with a 25ms window and 10ms shift. We performed mean and variance normalization on each frequency bin of the spectrogram. To form a mini-batch, we randomly selected 3-second segments from utterances. As a result, the input size is 257×300.

We implemented the proposed method based on PyTorch [38] and NVIDIA TITAN RTX GPU[1]. During the NAS process, the network with 8 neural cells was training using the algorithm in Section 3.2 for 50 epochs with a batch size of 16. We set the initial channels to 16, due to the limited GPU memory. We used Adam Optimizer to optimize both the architecture parameters α and weight parameters ω. We set the initial learning rate of the optimizer for α to 10^{-3}, and we set that of the optimizer for ω to 10^{-2}. Both learning rate were annealed down to zero following a cosine schedule [39]. We set the weight decay of both optimizers to 3×10^{-4}. The entire searching process takes around 5 days on a single GPU.

During training from scratch, the model was trained for 300 epochs with a batch size of 128. We explored different initial channels, as we will describe in Section 4.3. We used Adam Optimizer with initial learning rate of 10^{-2}, which was annealed down to zero following a cosine schedule. We set the weight decay of the optimizer to 3×10^{-4}. The entire training process takes around 1 day on a single GPU.

4.3. Experimental Setup

We explored the effect of using different number of neural cells and different number of channels during training from scratch. First, we derived a relatively light-weight model by setting the

---

[1]Codes available at https://github.com/TAMU-VITA/AutoSpeech

Table 1: *Speaker identification and speaker verification performance on VoxCeleb1 dataset. Dimensions indicate the dimensionality of the speaker embedding. N denotes the number of neural cells, and C denotes the number of initial channels.*

Method	Top-1(%)	Top-5(%)	EER (%)	Dimensions	Parameters
VGG-M [23]	80.50	92.1	10.20	1,024	67 million
ResNet-18 [8, 29]	79.48	90.97	12.30	512	12 million
ResNet-34 [37, 3, 9]	81.34	94.49	11.99	512	22 million
Proposed ($N = 8, C = 64$)	84.45	94.74	9.13	1,024	5 million
Proposed ($N = 30, C = 64$)	83.45	94.21	9.01	1,024	18 million
Proposed ($N = 8, C = 128$)	**87.66**	**96.01**	**8.95**	2,048	18 million

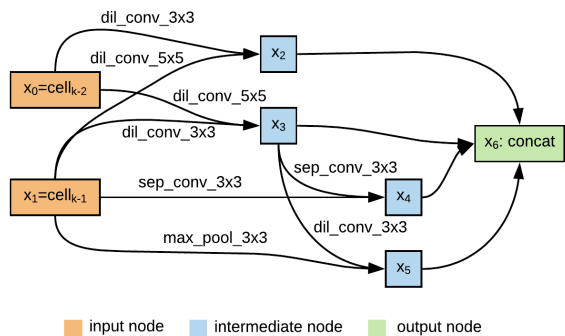Figure 2: *Architecture of the searched normal cell.*

Figure 3: *Architecture of the searched reduction cell.*

number of neural cells to $N = 8$ and the number of initial channels $C = 64$. Following this, we tried to construct two larger models by growing the number of neural cells and the number of channels, respectively. For a fair comparison between the two large models, we kept the total number of parameters the same in the two models. Consequently, we have a second model with $N = 30, C = 64$, and a third model with $N = 8, C = 128$. The embedding dimensions and the number of parameters of these models are shown in Table 1.

When evaluating the derived architecture for speaker identification, we followed the identification split of VoxCeleb1 dataset. We trained the derived CNN on training set and evaluated it on testing set. To obtain utterance-level speaker embeddings, we divided each test utterance into 3-second segments with 1.5-second overlap. Following this, we extracted the speaker embeddings on these segments, and used the average of these embeddings as the utterance-level speaker embedding. We computed Top-1 and Top-5 accuracies on these utterance-level speaker embeddings to measure the performance.

When evaluating the derived architecture for speaker verification, we followed the verification split of VoxCeleb1 dataset. We trained the derived CNN on training set and evaluated it on the trial pairs. We obtain utterance-level speaker embeddings as described above, and we used cosine similarity as the similarity metric. We used Equal Error Rate (EER) to measure the performance, which is commonly used for speaker verification systems [23].

4.4. Results and Design Insights

We visualize the architectures of the searched cells in Figures 2 and 3. We report the results on speaker identification and speaker verification, as shown in Table 1.

We first analyze the effect of the number of neural cells and the number of channels in the proposed method. We found that the model with 8 neural cells and 128 initial channels achieved the best performance among the proposed models on
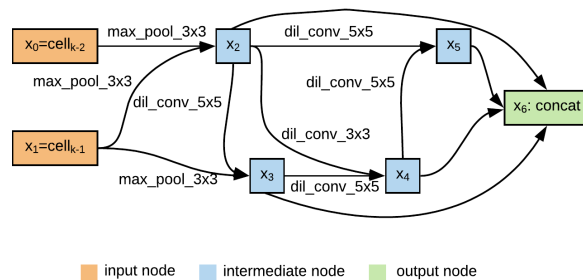
both identification (Top-1: 87.66%, Top-5: 96.01%) and verification (EER: 8.95%). Additionally, the two large models both outperform the light-weight one. When comparing between the two large models, we found that the model with ($N = 8, C = 128$) achieves better performance that the model with ($N = 30, C = 64$). This result suggests that increasing the number of channels is more effective than increasing the number of neural cells, when constructing a larger model from the searched neural cell.

We next compared the proposed method against three CNN architectures that were commonly used for speaker recognition: VGG-M [23], ResNet-18 [8, 29], and RestNet-34 [37, 3, 9]. As there are no open-source implementations from previous works on ResNet-18 and ResNet-34, we implemented these two architecture under similar training settings as the proposed method for a fair comparison. As shown in Table 1, all our proposed models outperform the baseline CNN architectures. Specifically, our light-weight model ($N = 8, C = 64$) outperforms ResNet-34, the best baseline architecture, by 3.11% Top-1 and 2.86% EER, with only 5 million parameters (ResNet-34: 22 million parameters). Our best model ($N = 8, C = 128$) outperforms ResNet-34 by 6.32% Top-1 and 3.04% EER, with only 18 million parameters. These results showed that the searched CNN architectures significantly improved both speaker identification and speaker verification performance, while achieving higher parameter efficiency and enjoying lighter weights.

5. Conclusions

This paper proposed an automatic approach to find the optimal CNN architecture for speaker recognition. The proposed approach has two stages. The first stage searches for the optimal architecture of a neural cell consisting of several different types of operations. The second stage derives a CNN model by stacking the searched neural cell several times and then trains the CNN model through the standard speaker recognition training process. Evaluation results on both speaker identification and speaker verification demonstrate significantly improved performance, while having lower model complexity.

6. References

[1] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4052–4056.

[2] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4879–4883.

[3] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5791–5795.

[4] S. O. Sadjadi, S. Ganapathy, and J. W. Pelecanos, "The IBM 2016 speaker recognition system," *arXiv preprint arXiv:1602.07291*, 2016.

[5] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: An end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, 2017.

[6] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[7] Z. Gao, Y. Song, I. McLoughlin, P. Li, Y. Jiang, and L. Dai, "Improving aggregation and loss function for better embedding learning in end-to-end speaker verification system," *Proceedings of Interspeech 2019*, pp. 361–365, 2019.

[8] M. Hajibabaei and D. Dai, "Unified hypersphere embedding for speaker recognition," *arXiv preprint arXiv:1807.08312*, 2018.

[9] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," *arXiv preprint arXiv:1804.05160*, 2018.

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

[11] 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.

[12] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[13] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[14] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[15] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the AAAI conference on Artificial Intelligence*, 2019.

[16] 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.

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

[18] R. Negrinho and G. Gordon, "Deeparchitect: Automatically designing and training deep architectures," *arXiv preprint arXiv:1704.08792*, 2017.

[19] A. Baruwa, M. Abisiga, I. Gbadegesin, and A. Fakunle, "Leveraging end-to-end speech recognition with neural architecture search," *arXiv preprint arXiv:1912.05946*, 2019.

[20] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 82–92.

[21] X. Gong, S. Chang, Y. Jiang, and Z. Wang, "AutoGAN: Neural architecture search for generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3224–3234.

[22] R. Quan, X. Dong, Y. Wu, L. Zhu, and Y. Yang, "Auto-ReID: Searching for a part-aware convnet for person re-identification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3750–3759.

[23] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: A large-scale speaker identification dataset," *arXiv preprint arXiv:1706.08612*, 2017.

[24] J. Martinez, H. Perez, E. Escamilla, and M. M. Suzuki, "Speaker recognition using mel frequency cepstral coefficients (mfcc) and vector quantization (vq) techniques," in *International Conference on Electrical Communications and Computers*. IEEE, 2012.

[25] V. Tiwari, "Mfcc and its applications in speaker recognition," *International journal on emerging technologies*, 2010.

[26] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.

[27] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *Twelfth annual conference of the international speech communication association*, 2011.

[28] P. Kenny, "Bayesian speaker verification with heavy-tailed priors." in *Odyssey*, 2010, p. 14.

[29] G. Bhattacharya, J. Alam, and P. Kenny, "Deep speaker recognition: Modular or monolithic?" in *Proceedings of Interspeech*, 2019.

[30] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[31] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 734–10 742.

[32] X. Dai, P. Zhang, B. Wu, H. Yin, F. Sun, Y. Wang, M. Dukhan, Y. Hu, Y. Wu, Y. Jia *et al.*, "Chamnet: Towards efficient network design through platform-aware model adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 398–11 407.

[33] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang, "Fasterseg: Searching for faster real-time semantic segmentation," *arXiv preprint arXiv:1912.10917*, 2019.

[34] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.

[35] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[36] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals of operations research*, 2007.

[37] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep speaker recognition," *Proceedings of Interspeech 2018*, 2018.

[38] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[39] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.