SEARCHING FOR EFFICIENT NETWORK ARCHITECTURES FOR ACOUSTIC SCENE CLASSIFICATION

Yuzhong Wu, Tan Lee

The Chinese University of Hong Kong
Department of Electronic Engineering, Shatin, N.T., Hong Kong S.A.R., China
yzwu@link.cuhk.edu.hk, tanlee@cuhk.edu.hk

ABSTRACT

Acoustic scene classification (ASC) is the task of classifying recorded audio signal into one of the predefined acoustic environment classes. While previous studies reported ASC systems with high accuracy, the computation cost and system complexity may not be optimal for practical mobile applications. Inspired by the success of neural architecture search (NAS) and the efficacy of MobileNets in vision applications, we propose a simple yet effective random search policy to obtain high accuracy ASC models under strict model size constraint. The search policy allows automatic discovery of the best trade-off between model depth and width, and statistical analysis of model design can be carried out using the evaluation results of randomly sampled architectures. To enable fast search, the search space is limited to several predefined efficient convolutional modules based on depth-wise convolution and swish activation function. Experimental results show that the CNN model found by this search policy gives higher accuracy compared to an AlexNet-like CNN benchmark.

Index Terms— Acoustic scene classification, convolutional neural network, neural architecture search, depthwise convolution, swish

1. INTRODUCTION

Acoustic scene classification (ASC) is the task of classifying recorded audio signal into one of predefined acoustic environment classes. It has been one of the major tasks in IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) since 2013. While the main research focus has been on constructing high-accuracy ASC systems with deeper and larger networks, there is inevitable concern on the computation cost and system complexity for practical mobile applications.

To reduce the model complexity and computation cost, many network architectures have been proposed in computer vision applications. MobileNet [1] is one of the pioneering light-weight network architectures with low latency. Depthwise separable convolutions (DSC) were proposed to build light-weight deep neural networks. MobileNet V2 [2] further improves the model efficiency by utilizing the inverted residual structure (with linear bottleneck). ShuffleNet [3] is another architecture which reduces computation cost via point-wise group convolution and channel shuffle. ShuffleNet V2 [4] introduces a simple operator called channel split to further improve the efficiency. Although these light-weight models achieve high performance in vision applications, the architectures may not be optimal when dealing with audio input. Fine-tuning of these architectures is needed to better suit the task of ASC.

Neural Architecture Search (NAS) is a technique for automating the design of deep neural networks (DNN). NAS has been widely investigated and applied to computer vision. Because the search space of DNN architectures could be tremendously large, making exhaustive search impossible, many search strategies were investigated to better explore the search space. For example, MetaQNN [5] is an NAS algorithm based on Q-learning to automatically generate high-performing CNN architectures. In [6], a recurrent neural network (RNN) is used to generate the network architecture descriptions. The RNN is trained by reinforcement learning to maximize the expected performance of the generated architectures on validation dataset. To further accelerate NAS, efficient NAS [7] with parameter sharing was proposed to discover network architectures by searching for an optimal subgraph within a predefined large computational graph. Single-path NAS [8] reduces the search space from multi-path to single-path. On the other hand, it was reported that a random architecture selection policy might possibly have similar performance with the NAS algorithms. [9] The success of NAS suggests that automatically searched model architectures could do better than manually tuned ones.

In this paper, we focus on finding efficient classifiers for ASC of 3 high-level classes (indoor, outdoor and transportation). Inspired by the efficacy of MobileNets and the success of NAS, we propose a simple yet effective random search policy to find high-performing ASC models under a strict model size constraint (< 500 KB). The search policy allows automatic discovery of the best trade-off between model depth and width, and statistical analysis of model design can be carried out using the evaluation results of randomly sampled architectures. In our system, scalogram features are extracted from binaural acoustic scene signals. The average-difference representation of scalogram features is used as the input feature of the ASC system. The search space of model architectures is empirically constrained to 2 - 4 convolutional blocks, with each block having 1-2 convolutional modules and one pooling layer. The convolutional modules include depth-wise separable convolution and inverted residual, with the ReLU activation function replaced by Swish. Experimental results on development dataset shows that CNN model obtained by this search strategy has higher performance compared to the AlexNet-like CNN benchmark. Analysis on sampled architectures shows that CNNs with constant number of filters in each convolutional layer perform better than CNNs with increasing number of filters in this ASC task.

2. EFFICIENT CONVOLUTIONAL MODULES

To develop an ASC system with low model complexity, we use depth-wise and point-wise convolution to construct the classifier

model. The depth-wise separable convolution (DSC) is the core module of MobileNet V1 [1], which is an efficient CNN designed for mobile vision applications. DSC reduces the number of parameters by factorizing the standard convolution operation into a depth-wise convolution and a point-wise convolution. The depth-wise convolution applies a single filter to each input channel. The point-wise convolution is a 1×1 convolution to combine the output of depth-wise convolution.

Inverted residual with linear bottleneck is another efficient module that is the basic building block of MobileNet V2 [2]. It takes a low-dimensional input, expands the feature to high-dimension, and then apply filtering by depth-wise convolution. Subsequently it projects the feature back to low-dimensional representation using point-wise convolution. After the final point-wise convolution, no non-linear activation function is applied, and this explains the name of linear bottleneck. The module contains a residual connection between input and output. This improves gradient flow through the network and enables effective training of deeper networks. The residual connection is described as "inverted" because it exist between narrow parts of the network, which is opposite to the original formulation of residual connection [10]. The inverted design is considerably more memory efficient.

We consider both DSC and inverted residual (with linear bottleneck) in ASC system design. Instead of using the original formulation, we replace the ReLU activation function with Swish. Similar to ReLU, Swish function is bounded below and unbounded above. Besides, Swish function has a smoother output curve and preserves more information from negative-valued input compared to ReLU. These properties may provide benefit in terms of convergence. It was reported that Swish performs better than ReLU for various applications [11]. Denoting the input as x, the Swish function is defined as:

$$Swish(x) = x \cdot sigmoid(\beta x), \quad (1)$$

where β can be a constant or trainable parameter. For simplicity we set $\beta = 1$.

Figure 1 shows the two types of convolution modules used in our experiments. For a convolution layer, the input variable "inp" refers to the number of input channels and "out" refers to the number of output channels of this module. The inverted residual module has an expansion ratio of 3, which is smaller than the typical ratio 6 used in the original MobileNet V2. The stride of convolution layers is 1 and the kernel size of depth-wise convolution is typically 3×3.

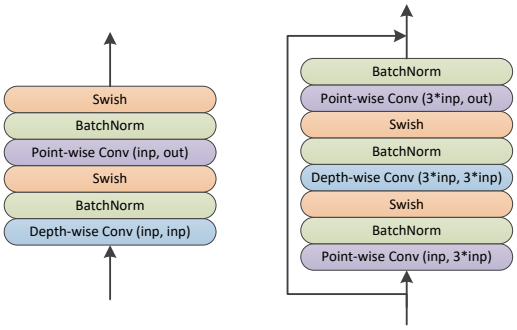

Figure 1: The two types of convolutional modules used for network architecture sampling. Left: Depth-wise separable convolution module; right: inverted residual with linear bottleneck with expansion ratio 3. The swish activation function is used instead of ReLU.

3. SEARCHING FOR NETWORK ARCHITECTURES

3.1. Network Architectures

The search space of network architecture is constrained to CNNs with classic single-path design, which is a stacking of convolution modules and pooling operations with a fully connected output layer in the end. Specifically, the first layer of the network is fixed as a standard 3×3 convolution layer called stem convolution layer. After the stem convolution layer there are 2 - 4 convolutional blocks. We define a convolutional block as a stacking of convolutional module(s) and a pooling layer. A block may contain one or two convolutional modules(s) and one pooling layer. Then a global average pooling layer follows. The output layer is a fully connected layer with the output dimension of 3, representing the output probabilities of the 3 acoustic scene classes. Notice that probabilities for an audio file are the average of its segments' probabilities. Figure 2 shows an example of the model in the search space.

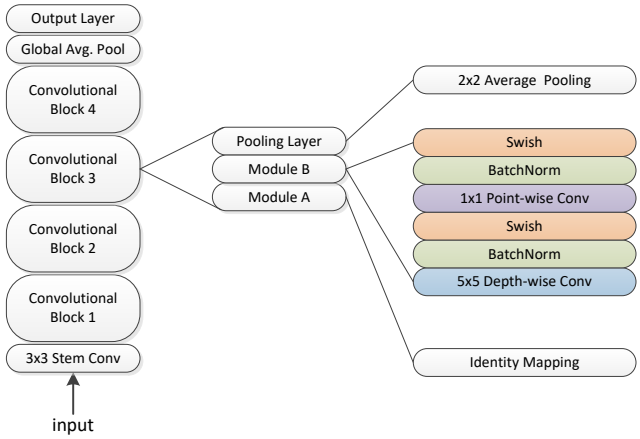

Figure 2: Illustration of a model in the search space. It has 4 convolutional blocks. The 3rd block contains a 5×5 DSC module and an average pooling layer.

3.2. Search Space

We search for CNN models which meet the model architecture definition in Section 3.1. To reduce the search space, we limit the types of modules in convolutional blocks to 5. The modules are shown as in Table 1. Identity mapping means the input is identical to the output, which is a dummy module used to increase the diversity of sampled network's depth. Modules with id 1 and 2 are DSC modules described in Section 2, with kernel size being 3×3 and 5×5. Likewise, modules with id 3 and 4 are inverted residual (with linear bottleneck) modules described in Section 2. The different choices of kernel size allow sampling of CNN architectures with different receptive field, which is a factor affecting the classification accuracy [12]. For the pooling layer, either 2×2 average pooling or 2×2 max pooling can be selected.

The details of our search space are shown in Table 2. The "Stem conv. output filters" specify the number of output filters in the stem convolution layer. "Growth ratio of filter number" controls the increment of filter number after each convolutional block. For example, given the number of output filters of stem convolution layer as 32, a growth ratio of 1.5 means the number of channels after the first

Table 1: List of modules that can be selected to build a model architecture.

id	Module description
0	Identity Mapping
1	3×3 DSC module
2	5×5 DSC module
3	3×3 Inverted Residual module
4	5×5 Inverted Residual module

Table 2: The search space of CNN architectures.

Model Configuration	Possible Choice
Number of blocks	2,3,4
Stem conv. output filters	4 - 128
Growth ratio of filter number	1.0,1.25,1.50,1.75,2.0
Block 1 module A id	0,1,2,3,4
Block 1 module B id	1,2,3,4
Block 1 pooling layer	Avg. Pool, Max Pool
Block 2 module A id	0,1,2,3,4
Block 2 module B id	1,2,3,4
Block 2 pooling layer	Avg. Pool, Max Pool
...	

block is $32 \times 1.5 = 48$, after the second block is $48 \times 1.5 = 72$, and so on. Allowing different growth ratios increases the diversity of network shape in sampled architectures. When sampling a model architecture from the search space, for each configuration item, every possible choice has equal probability of being chosen.

3.3. Search Scheme

With the search space defined, we use the following search scheme to find the high-performing architectures. First, a candidate model architecture is randomly sampled from the search space. Then its model size is checked. If the model size is too small (e.g., smaller than 250 KB) or too large (larger than 500 KB), we discard this candidate architecture. If the model size requirement is satisfied, we train the candidate model for only 3 epochs. The small number of training epochs are empirically set for fast evaluation of candidate architectures. This is based on the assumption that a better network architecture consistently outperforms a worse architecture given the same number of training epochs. The trained candidate model together with its accuracy on test set and test loss will be saved. After a considerable number of candidate models being sampled, we select the model architecture with highest test set accuracy or lowest test loss. It is trained from scratch with sufficient training epochs (60 epochs in this study) for final evaluation.

4. EXPERIMENTS

4.1. Dataset

The TAU Urban Acoustic Scenes 2020 3Class development dataset [13] is used for model training and testing. The audios are labeled with three high level acoustic scenes: indoor, outdoor and transportation. All the audio data is recorded with a single device in binaural 48000Hz 24-bit format. Each audio signal is 10-second long and there is in total 40 hours of audios in the dataset. We use the officially provided train/test split for our experiments: the training set contains 25.5 hours of audios and the test set contains 11.6 hours of audios. The ratio of the amount of indoor, outdoor and transportation data is around $3 : 4 : 3$.

4.2. Data Preprocessing

For each 10-second binaural audio signal in the dataset, we compute the wavelet-based filter-bank (scalogram) feature for each channel. STFT is applied on audio waveform with 2048 FFT points, window length of 25 ms and hop length of 10 ms. Python library Kymatio [14] is used to generate wavelet filters using support size of 2048, maximum scale of the filters being 1024 and number of wavelets per octave being 16. The wavelet filter-bank is applied on the logarithm magnitude of the STFT result to obtain the scalogram features. The resulted scalogram feature has the shape of $(1000, 128)$, where 1000 is the number of time frames and 128 is the number of frequency bins. The average and difference of the scalogram features from two channels are concatenated and cut into non-overlapping segments of 128 time frames. The resulted feature segment of shape $(2, 128, 128)$ is used as CNN input.

4.3. Optimization

For training the candidate model, we use initial learning rate (LR) of 0.001, and the LR is multiplied with 0.1 after each epoch. The number of training epochs is 3. The model is trained with binary cross-entropy loss with Adam optimizer ($\beta_1 = 0.9$ and $\beta_2 = 0.999$). Weight decay with coefficient 0.0015 is used for regularization purpose. Mixup [15] is used for data augmentation. After the final best-performing model is determined, the best model is trained from scratch. In this case, the number of training epochs is 60. LR is multiplied with 0.5 after every 4 epochs.

4.4. Results and Discussion

We randomly sampled 510 distinct candidate architectures with model size in the range of 250 KB - 500 KB. Training (for 3 epochs) and testing a candidate model take about 20 minutes using single GPU and thus in total it takes about 170 hours to evaluate 510 candidate architectures. Figure 3 shows the accuracies of candidate models (trained for 3 epochs) which satisfy the model size requirement. It can be seen that the accuracies of most candidate models are in the range of 92% - 94%. The difference in model size seems to have very mild influence on the model accuracy. Besides, we observe that models having highest accuracy not necessarily have lowest test loss given 3 training epochs. Thus, we picked two representative model architectures: model A has the highest test set accuracy and model B has the lowest test loss. Their architectures are shown in Table 3. Notice that model B has a constant number of filters in each convolution layer (growth ratio of filter number being 1.0). It is quite counter-intuitive because for typical CNNs, the number of filters will increase after each convolutional block.

Figure 4 shows the accuracy distribution of candidate models with different growth ratio of filter number. According to the figure, we calculate the mean accuracy of models with growth ratio being 2.0, 1.75, 1.5, 1.25 and 1.0, and the mean accuracy is 92.5%, 92.9%, 92.9%, 93.2% and 93.3% respectively (the mean model sizes of different growth ratios are similar, which are around 374 ± 8 KB). Thus, models with low growth ratio of filter number exhibit higher accuracy than high growth ratio. The reason could

Table 3: Architectures of light-weight models with highest test accuracy (model A) and lowest test loss (model B) given 3 training epochs. "IRwLB" means the inverted residual with linear bottleneck module. "DSC" means the depthwise separable convolution module. For each layer, the number inside "()" is the number of output filters.

	Model A	Model B
1	3×3 Stem Conv. (76)	3×3 Stem Conv. (64)
2	3×3 IRwLB (76)	3×3 IRwLB (64)
3	3×3 DSC (133)	2×2 AvgPool
4	2×2 AvgPool	3×3 DSC (64)
5	5×5 DSC (133)	2×2 AvgPool
6	5×5 DSC (232)	5×5 DSC (64)
7	2×2 MaxPool	3×3 DSC (64)
8		2×2 AvgPool
9		3×3 IRwLB (64)
10		3×3 DSC (64)
11		2×2 AvgPool
12	GlobalAvgPool	GlobalAvgPool
13	Fully Connected	Fully Connected
14	3-way Sigmoid	3-way Sigmoid

be because we only have 3 output classes, and thus it is sufficient to classify them using embedding features with small dimension.

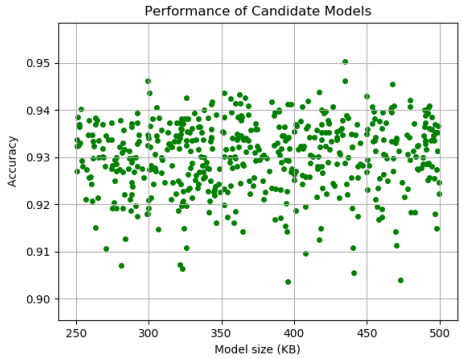

Figure 3: The candidate models' accuracy (trained for 3 epochs) and their model sizes. Each dot represents a candidate model.

To have a grasp of how well the light-weight models perform, we manually designed and trained a CNN modified from AlexNet [16] with large model size (33.8 MB). Besides, we evaluate the performance of several light-weight models which are designed for vision applications. To apply them on the ASC task, we change the number of input channels to 2 and the number of output classes to 3, other parts of the model architectures remain unchanged.

Table 4 shows the performance of models with various model size. It can be seen that MobileNets and ShuffleNet V2 preserve most of the performance comparing to the large model "AlexNet (Modified)". Meanwhile, our model A and model B perform even better than the "AlexNet (Modified)" with model size less than $1/10$ of ShuffleNet V2. The model size can be further reduced by converting the model parameters to float-16 format (originally they are in float-32) without loss of accuracy.

In the Task 1B of DCASE 2020 challenge, our ASC system using a single model B achieves an accuracy of 94.2% in the evaluation dataset [17], which ranks 6th out of the 30 teams. The order of official system ranking (on evaluation dataset) of our submitted systems is the same as the order of system ranking on the test set of development dataset, which indicates that there should be little over-fitting to the development test set when its information is used to pick the high-performing models.

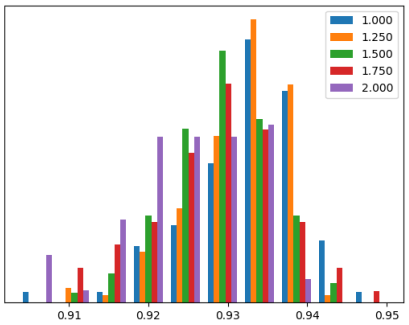

Figure 4: Normalized histograms of accuracy of candidate models with different growth ratio of filter number. The x-axis represents the candidate model accuracy on development dataset. The y-axis represents the normalized count of models lying in the accuracy range.

Table 4: Size and accuracy of ASC models. The models are trained and tested using the officially provided train/test split on development dataset.

Model	Model Size (MB)	Accuarcy
AlexNet (Modified)	33.8	95.5%
MobileNet V1	12.2	94.8%
MobileNet V2	8.5	94.4%
ShuffleNet V2	4.8	94.0%
Model A (float-32)	0.42	95.6%
Model A (float-16)	0.21	95.6%
Model B (float-32)	0.29	95.8%
Model B (float-16)	**0.15**	$\mathbf{95.8\%}$

5. CONCLUSIONS

In this study, a simple yet effective random search policy is proposed to find high-performing light-weight models for ASC. On the TAU Urban Acoustic Scenes 2020 3Class development dataset used for Task 1B of DCASE 2020 challenge, our best-performing model achieves an accuracy of 95.8% with model size being only 150 KB. Analysis on sampled architectures shows that CNNs with constant number of filters in each convolutional layer perform better than CNNs with increasing number of filters. We show that the simple random search scheme can work well in finding high-performing models, and it may serve as a baseline for further study on NAS algorithms in ASC task.

6. REFERENCES

[1] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv e-prints*, p. arXiv:1704.04861, Apr. 2017.

[2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *arXiv e-prints*, p. arXiv:1801.04381, Jan. 2018.

[3] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," *arXiv e-prints*, p. arXiv:1707.01083, July 2017.

[4] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," *arXiv e-prints*, p. arXiv:1807.11164, July 2018.

[5] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing Neural Network Architectures using Reinforcement Learning," *arXiv e-prints*, p. arXiv:1611.02167, Nov. 2016.

[6] B. Zoph and Q. V. Le, "Neural Architecture Search with Reinforcement Learning," *arXiv e-prints*, p. arXiv:1611.01578, Nov. 2016.

[7] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient Neural Architecture Search via Parameter Sharing," *arXiv e-prints*, p. arXiv:1802.03268, Feb. 2018.

[8] D. Stamoulis, R. Ding, D. Wang, D. Lymberopoulos, B. Priyantha, J. Liu, and D. Marculescu, "Single-Path NAS: Designing Hardware-Efficient ConvNets in less than 4 Hours," *arXiv e-prints*, p. arXiv:1904.02877, Apr. 2019.

[9] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the Search Phase of Neural Architecture Search," *arXiv e-prints*, p. arXiv:1902.08142, Feb. 2019.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv e-prints*, p. arXiv:1512.03385, Dec. 2015.

[11] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," *arXiv e-prints*, p. arXiv:1710.05941, Oct. 2017.

[12] K. Koutini, H. Eghbal-zadeh, and G. Widmer, "Receptive-field-regularized CNN variants for acoustic scene classification," *arXiv e-prints*, p. arXiv:1909.02859, Sept. 2019.

[13] T. Heittola, A. Mesaros, and T. Virtanen. (2020, Feb.) TAU Urban Acoustic Scenes 2020 3Class, Development dataset. [Online]. Available: https://doi.org/10.5281/zenodo.3670185

[14] M. Andreux, T. Angles, G. Exarchakis, *et al.*, "Kymatio: Scattering Transforms in Python," *arXiv e-prints*, p. arXiv:1812.11214, Dec 2018.

[15] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond Empirical Risk Minimization," *arXiv e-prints*, p. arXiv:1710.09412, Oct 2017.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[17] (2020) TAU Urban Acoustic Scenes 2020 3Class, Evaluation dataset. [Online]. Available: https://doi.org/10.5281/zenodo.3685835