SEARCHING FOR EFFICIENT NETWORK ARCHITECTURES FOR ACOUSTIC SCENE CLASSIFICATION

Technical Report

Yuzhong Wu, Tan Lee

Department of Electronic Engineering
The Chinese University of Hong Kong, Shatin, NT, Hong Kong SAR, China
yzwu@link.cuhk.edu.hk, tanlee@cuhk.edu.hk

ABSTRACT

This technical report describes our submission for Task 1B of DCASE2020 challenge. The objective of task 1B is to construct an acoustic scene classification (ASC) system with low model complexity. In our ASC system, the average-difference time-frequency features are extracted from binaural audio waveforms. A random search policy is used to find the best-performing CNN architecture while satisfying the requirement of model size. The search is limited to several predefined efficient convolutional modules based on depth-wise convolution and swish activation function to constrain the size of search space. Experimental results on development dataset shows that CNN model obtained by this search strategy has 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 task in IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE) since 2013. This technical report describes the details of our submission for Task 1B of DCASE 2020. The task requires a light-weight acoustic scene classifier to determine input audio as one of the three scene classes: indoor, outdoor and transportation. The model size of the classifier is required to be less than 500KB.

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 which makes exhaustive search impossible, many searching strategies were proposed to better explore the search space. For example, MetaQNN [1] is an NAS algorithm based on Q-learning to automatically generate high-performing CNN architectures. In [2], a recurrent neural network (RNN) is used to generate the network architecture descriptions. The RNN is trained with reinforcement learning to maximize the expected performance of the generated architectures on validation dataset. To further accelerate NAS, efficient NAS [3] with parameter sharing was proposed to discover network architectures by searching for an optimal subgraph within a predefined large computational graph. Single-path NAS [4] further reduces the search space from multi-path to single-path. On the other hand, it was reported that a random architecture selection policy may possibly have similar performance with the NAS algorithms. [5] The success of NAS shows that automatically searched model architectures could do better than manually tuned ones.

In this study, 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. We propose a simple yet effective random search policy to find high-performing light-weight model. The search space of model architectures is 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.

2. FEATURE DESIGN

2.1. Wavelet-Based Filter-Bank Features

Previous studies showed that wavelet-based filter-bank (scalogram) features performed better than log-mel features in ASC task [6, 7]. We follow the same setting as described in [6] to extract the scalogram features. Compared to log-mel features with the same number of frequency bins, the extracted scalogram features have higher frequency resolution in low frequencies.

2.2. Average-Difference Representation

Task 1B of DCASE 2020 deals with binaural audios. Compared to monaural audio, binaural audio contains spatial information about the sounds in the scene. For example, we can hear a car passing by from our left side to the right side in binaural audio, while this is not possible with monaural audio. To utilize spatial information for ASC, scalogram features S_{left} and S_{right} are computed from the left channel and right channel of the input audio. Their average S_{avg} and their difference S_{diff} are obtained as:

$$S_{avg} = S_{left} + S_{right}, \tag{1}$$

$$S_{diff} = S_{left} - S_{right}. \tag{2}$$

The input of our ASC system S_{in} is given by the concatenation of S_{avg} and S_{diff}.

3. 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 [8], which is an efficient CNN designed for mobile vision applications. DSC includes 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 found as the basic building block of MobileNet V2 [9]. 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. Besides, the module contains a shortcut connection between input and output.

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. Swish significantly improves the accuracy of neural networks as compared to ReLU for various applications [10]. Denoting the input as x, the Swish function is defined as:

$$Swish(x) = x \cdot sigmoid(\beta x), \tag{3}$$

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 either 3×3 or 5×5.

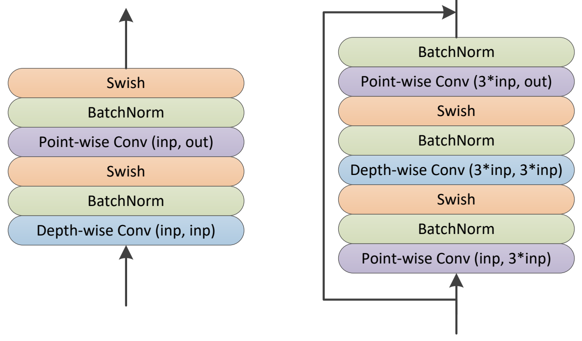
Swish
BatchNorm
Point-wise Conv (inp, out)
Swish
BatchNorm
Depth-wise Conv (inp, inp)
BatchNorm
Point-wise Conv (3*inp, out)
Swish
BatchNorm
Depth-wise Conv (3*inp, 3*inp)
Swish
BatchNorm
Point-wise Conv (inp, 3*inp)

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.

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

4. SEARCHING FOR NETWORK ARCHITECTURES

4.1. Network Architectures

The search space of network architecture is constrained to convolutional networks. The input is the average-difference scalogram feature of audio segment of size $(2, 128, 128)$. The first layer of the network is fixed as a standard 3×3 convolution layer called stem convolution layer. After the stem convolution layer is 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.

4.2. Search Space

We search for CNN models which meet the model architecture definition in Section 4.1. 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. Modules with id 1 and 2 are DSC modules described in Section 3, 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 3. 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 block is $32 \times 1.5 = 48$, after the second block is $48 \times 1.5 = 72$, and so on. When sampling a model architecture from the search space, for each configuration item, every possible choice has equal probability of being chosen.

4.3. 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

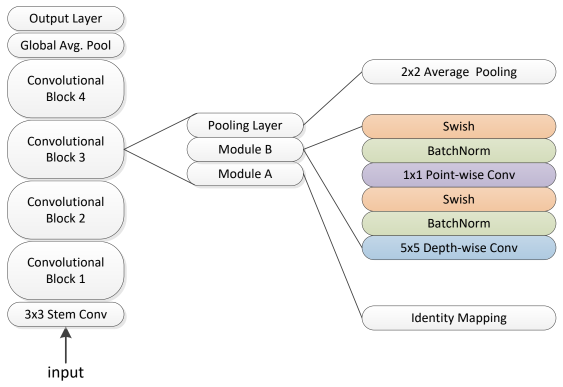

Figure 2: Illustration of a model in the search space. It has 4 convolutional blocks. The 3rd block contains a 5×5 DSC module and an average pooling layer.

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
...	

train the candidate model for only 3 epochs. The small number of training epochs are empirically set to constrain the time consumption of our search scheme, which is inspired by the early stopping strategy in NAS as described in [11]. The trained candidate model together with its accuracy on validation set and validation loss will be saved. After a considerable number of candidate models being found, we pick the model architecture with highest validation set accuracy or lowest validation loss to train from scratch with sufficient training epochs ($\gg 3$).

5. EXPERIMENTS

5.1. Data Preprocessing

The TAU Urban Acoustic Scenes 2020 3Class development dataset [12] is used for model training and testing. For each 10-second binaural audio signal in the dataset, we compute the scalogram feature for each channel. To extract the scalogram feature, STFT is applied on audio waveform with 2048 FFT points, window length of 25 ms and hop length of 10 ms. 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 $(1000, 128)$ where 1000 is the number of time frames and 128 is the number of frequency bins. Then we compute the average and difference of the scalogram features as described in 2.2. Finally, features are cut into non-overlapping segments of 128 time frames. Notice that we use the Python library Kymatio [13] to generate wavelet filters using support size of 2048, maximum scale of the filters being 1024 and number of wavelets per octave Q = 16. As a result, the total number of filters in the wavelet filter-bank is 128.

5.2. Optimization

For training the candidate model, we use initial learning rate of 0.001, and the learning rate 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 [14] ($\beta_1 = 0.9$ and $\beta_2 = 0.999$). Weight decay with coefficient 0.0015 is used for regularization purpose. Mixup [15] approach is used for data augmentation. After the final best-performing model is determined, the best model is trained from scratch for challenge submission. In this case, the number of training epochs is 60. Learning rate is multiplied with 0.5 after every 4 epochs.

5.3. Results and Discussion

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%. Besides, we observe that models having highest accuracy not necessarily have lowest validation loss given 3 training epochs. Thus, we picked two model architectures for challenge submission: model A has the highest validation set accuracy and model B has the lowest validation loss. Their architectures are shown in Table 3. After training them from scratch with 60 epochs, model A has an accuracy of 95.6% and model B has an accuracy of 95.8%. 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.

To have a grasp of how well the light-weight models perform, we also manually designed and trained an AlexNet-like model with large model size (33.8 MB). It is trained for 40 epochs and achieves an accuracy of 95.5%. Notice that the found light-weight model A has an accuracy of 95.6% (trained from scratch with 40 epochs for comparison), which is better than the AlexNet-like model. It should be noted that the model size of the model A is only 434.8 KB, which is approximately $1/80$ of the size of the AlexNet-like model.

6. SYSTEM SUBMISSION

We submit 4 systems to the DCASE2020 task1B challenge. The first system (Wu_CUHK_task1b_1) uses the model B trained on the entire development dataset with 60 epochs. The second system (Wu_CUHK_task1b_2) averages the model prediction from model A and model B. Likewise, the third system (Wu_CUHK_task1b_3) averages the model predictions from 3 independently trained model B.

The fourth system (Wu_CUHK_task1b_4) averages the model predictions from 2 models: one model B trained with average-difference scalogram features, and another model B trained with decomposed scalogram features (only the medium-duration and short-duration components are used). To obtain the decomposed scalogram features, we average the scalogram features from the left and

Table 3: Architectures of light-weight models with highest validation accuracy (model A) and lowest validation 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	10-way Softmax	10-way Softmax

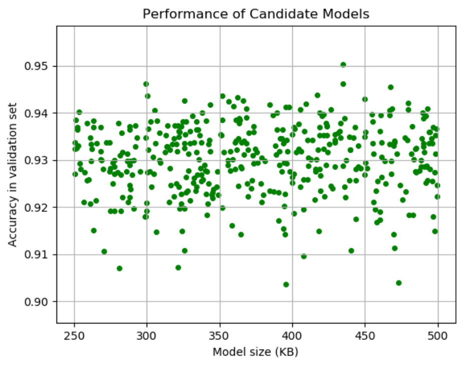

Figure 3: The candidate models' accuracy on validation set and their model sizes. Each dot represents a candidate model.

right channel. Then the decomposition method described in [6] is applied with two median filters (kernel sizes being 201 and 11). Concatenation of S_{medium} and S_{short} is used as the input features. The purpose of using different input features is to introduce more diversity in the ensemble system.

For the second, third and fourth system, to satisfy the model size requirement, the model parameters and input features are converted to 16-bit float numbers (originally they are 32-bit float). Table 4 shows the size and performance of the systems on development dataset.

Table 4: Size and accuracy of ASC systems described in Section 6. Notice that in this table the systems are trained and tested using the officially provided train/test split on development dataset, while for challenge submission we train the systems using all data in development dataset.

System Name	System Size (KB)	Acuracy
Wu_CUHK_task1b_1	299.26	95.8%
Wu_CUHK_task1b_2	367.03	96.2%
Wu_CUHK_task1b_3	448.89	96.3%
Wu_CUHK_task1b_4	299.26	96.5%

7. REFERENCES

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] Y. Wu and T. Lee, "Time-frequency feature decomposition based on sound duration for acoustic scene classification," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 716–720.

[7] H. Chen, Z. Liu, Z. Liu, P. Zhang, and Y. Yan, "Integrating the data augmentation scheme with various classifiers for acoustic scene modeling," DCASE2019 Challenge, Tech. Rep., June 2019.

[8] 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.

[9] 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.

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

[11] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating Neural Architecture Search using Performance Prediction," *arXiv e-prints*, p. arXiv:1705.10823, May 2017.

[12] 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

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

[14] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec. 2014.

[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.