SIMPLE POOLING FRONT-ENDS FOR EFFICIENT AUDIO CLASSIFICATION

Xubo Liu[1], *Haohe Liu*[1], *Qiuqiang Kong*[2], *Xinhao Mei*[1], *Mark D. Plumbley*[1], *Wenwu Wang*[1]

[1]Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, UK
[2]Speech, Audio, and Music Intelligence (SAMI) Group, ByteDance, China

ABSTRACT

Recently, there has been increasing interest in building efficient audio neural networks for on-device scenarios. Most existing approaches are designed to reduce the size of audio neural networks using methods such as model pruning. In this work, we show that instead of reducing model size using complex methods, eliminating the temporal redundancy in the input audio features (e.g., mel-spectrogram) could be an effective approach for efficient audio classification. To do so, we proposed a family of simple pooling front-ends (SimPFs) which use simple non-parametric pooling operations to reduce the redundant information within the mel-spectrogram. We perform extensive experiments on four audio classification tasks to evaluate the performance of SimPFs. Experimental results show that SimPFs can achieve a reduction in more than half of the number of floating point operations (FLOPs) for off-the-shelf audio neural networks, with negligible degradation or even some improvements in audio classification performance.

Index Terms— Audio classification, audio front-ends, on-device, convolutional neural networks, deep learning

1. INTRODUCTION

Audio classification is an important research topic in the field of signal processing and machine learning. There are many applications of audio classification, such as acoustic scene classification [1], sound event detection [2] and keywords spotting [3]. Audio classification plays a key role in many real-world applications including acoustic monitoring [4], healthcare [5] and multimedia indexing [6].

Neural network methods such as convolutional neural networks (CNNs) have been used for audio classification and have achieved state-of-the-art performance [7, 8, 9]. Generally, state-of-the-art audio classification models are designed with large sizes and complicated modules, which make the audio classification networks computationally inefficient, in terms of e.g. the number of floating point operations (FLOPs) and running memory. However, in many real-world scenarios, audio classification models need to be deployed on resource-constrained platforms such as mobile devices [10].

There has been increasing interest in building efficient audio neural networks in the literature. Existing methods can generally be divided into three categories. The first is to utilize model compression techniques such as pruning [11, 12]. The second is to transfer the knowledge from large-scale pre-trained model to a small model via knowledge distillation [13, 14, 15]. The last one is to directly exploit efficient networks for audio classification, such as MobileNets [7, 16]. In summary, these methods mainly focus on reducing model size. However, the computational cost (e.g., FLOPs) of the audio neural network is not only determined by the size of the model, but also highly dependent on the size of the input features.

As existing audio neural networks usually take mel-spectrogram which may be temporally redundant. For example, the pattern of a siren audio clip is highly repetitive in the spectrogram, as shown in Figure 1. In principle, if one can remove the redundancy in the input mel-spectrogram, the computational cost can be significantly reduced. However, reducing input feature size for audio neural networks has received little attention in the literature, especially in terms of improving their computation efficiency.

In this paper, we propose a family of **sim**ple **p**ooling **f**ront-ends (SimPFs) for improving the computation efficiency of audio neural networks. SimPFs utilize simple non-parametric pooling methods (e.g., max pooling) to eliminate the temporally redundant information in the input mel-spectrogram. The simple pooling operation on an input mel-spectrogram achieves a substantial improvement in computation efficiency for audio neural networks. To evaluate the effectiveness of SimPFs, we conduct extensively experiments on four audio classification datasets including DCASE19 acoustic scene classification [17], ESC-50 environmental sound classification [18], Google SpeechCommands keywords spotting [3], and AudioSet audio tagging [19]. We demonstrate that SimPFs can reduce more than half of the computation FLOPs for off-the-shelf audio neural networks [7], with negligibly degraded or even improved classification performance. For example, on DCASE19 acoustic scene classification, SimPF can reduce the FLOPs by 75% while improving the classification accuracy approximately by 1.2%. Our proposed SimPFs are simple to implement and can be integrated into any audio neural network at a negligible computation cost. The code of our proposed method is made available at GitHub[1].

The remainder of this paper is organized as follows. The next section introduces the related work of this paper. Section 3 introduces the method SimPFs we proposed for efficient audio classification. Section 4 presents the experimental settings and the evaluation results. Conclusions and future directions are given in Section 5.

2. RELATED WORK

Our work relates to several works in the literature: efficient audio classification, feature reduction for audio classification, and audio front-ends. We will discuss each of these as follows.

2.1. Efficient audio classification

Efficient audio classification for on-device applications has attracted increasing attention in recent years. Singh et al. [11, 12, 20] proposed to use pruning method to eliminate redundancy in audio convolutional neural networks for acoustic scene classification, which can reduce approximately 25% FLOPs at 1% reduction in accuracy. Knowledge distillation methods [13, 14, 15] have been used for efficient audio

[1]https://github.com/liuxubo717/SimPFs

classification via transferring knowledge of large teacher models to small student on-device models. Efficient models such as MobileNets [21] have been proposed for visual applications to mobile devices. Kong et al. [7] have adapted MobileNet for audio tagging, demonstrating its potential to improve computational efficiency for audio classification. Unlike these methods, which focus on reducing model size, our proposed SimPF aims to reduce the size of input features.

2.2. Feature reduction for audio classification

Feature reduction methods such as principal component analysis (PCA) have been widely investigated for audio classification with classical machine learning methods such as discriminative support vector machines (SVMs) [22, 23]. The most relevant work to SimPFs in the literature is [23], where max and average pooling operations are applied to sparse acoustic features to improve the performance of SVM-based audio classification, especially in a noisy environment. In contrast to this method, SimPFs are designed to improve the efficiency of audio neural networks, whose computational cost is highly dependent on the size of the input features. In addition, the effectiveness of SimPFs is extensively evaluated on various audio classification benchmarks.

2.3. Audio Front-ends

Audio front-ends were studied as an alternative to mel-filterbanks for audio classification in the last decade. For example, trainable front-ends SincNet [24] and LEAF [25] are proposed for learning audio features from the waveform. These front-ends perform better than using traditional mel-filterbanks on various audio classification tasks. Unlike existing work on learnable front-ends, SimPFs are non-parametric and built on top of a widely-used mel-spectrogram for audio neural networks. Our motivation for designing SimPFs is not to learn a replacement for mel-spectrogram, but to eliminate temporal redundancy in mel-spectrograms. This redundancy significantly impacts on the efficiency of audio neural networks but is often ignored by audio researchers.

3. PROPOSED METHOD

Mel-spectrogram is widely used as an input feature for neural network-based audio classification. Given an audio signal x, its mel-spectrogram is a two-dimensional time-frequency representation denoted as $X \in \mathbb{R}^{F\times T}$, where T and F represent the number of time frames and the dimension of the spectral feature, respectively. An audio neural network takes X as the input and predicts the category y of the input audio:

$$\mathrm{g}(X;\theta) \mapsto y \tag{1}$$

where $\mathrm{g}(\cdot,\theta)$ stands for the model parameterized by θ. Generally, the computation cost of the neural network g is dependent on both the size of the parameter θ and the size of input X.

In this work, we propose to use simple non-parametric pooling methods to eliminate the temporal redundancy in the input mel-spectrogram. SimPFs can significantly improve the computational efficiency of audio neural networks without any bells and whistles. Formally, SimPFs take a mel-spectrogram X as input and output a compressed time-frequency representation $C \in \mathbb{R}^{F\times kT}$, where $k \in (0,1)$ is the compression coefficient in time domain and $\frac{1}{k}$ should be a positive integer. We will introduce a family of SimPFs which uses five pooling methods.

SimPF (Max) We apply a 2D max pooling with kernel size $(1, \frac{1}{k})$ over an mel-spectrogram X. The output is described as follows:

$$C(f,t) = \max_{n=0,\dots,\frac{1}{k}-1} X(f, \frac{t}{k}+n) \tag{2}$$

where $f = 0, ..., F-1$ and $t = 1, ..., kT-1$.

SimPF (Avg) Similar to SimPF (Max), we apply a 2D average pooling with kernel size $(1, \frac{1}{k})$ over an input mel-spectrogram X. Formally, the output is described as:

$$C(f,t) = k\sum_{n=0}^{\frac{1}{k}-1} X(f, \frac{t}{k}+n). \tag{3}$$

SimPF (Avg-Max) In this case, we add the outputs of *SimPF (Max)* and *SimPF (Avg)*, which is defined as:

$$C(f,t) = \max_{n=0,\dots,\frac{1}{k}-1} X(f, \frac{t}{k}+n) + k\sum_{n=0}^{\frac{1}{k}-1} X(f, \frac{t}{k}+n). \tag{4}$$

SimPF (Spectral) We adapt the spectral pooling method proposed in [26]. Concretely, the Discrete Fourier Transform (DFT) y of the input mel-spectrogram X is computed by:

$$y = \mathrm{DFT}(X) \in \mathbb{C}^{F\times T} \tag{5}$$

and the zero frequency is shifted to the center of y. Then, a bounding box of size (F, kT) crops y around its center to produce $y^{\mathrm{crop}} \in \mathbb{C}^{F\times kT}$. The output is obtained by exerting inverse DFT on y^{crop}:

$$C = \mathrm{DFT}^{\mathrm{inverse}}(y^{\mathrm{crop}}). \tag{6}$$

SimPF (Uniform) We uniformly sample one spectral frame every $\frac{1}{k}$ frames. The output of is calculated by:

$$C(f,t) = X(f, \frac{t}{k}). \tag{7}$$

We visualize the mel-spectrogram of a siren audio clip and the compressed spectrograms using different SimPFs with 50% compression factor in Figure 1. Intuitively, we can observe that even though the resolution of the spectrogram is compressed by half, the pattern of the siren remains similar in the spectrogram, which indicates high redundancy in the siren spectrogram.

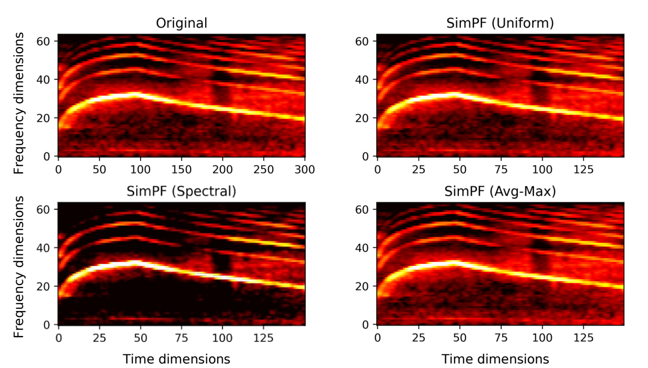

Fig. 1. Visualization of the impact of SimPFs on the mel-spectrogram of a siren audio clip with 50% compression factor.

Model (CNN10 [7])	DCASE19			ESC-50		SpeechCommands	
Baseline	0.710			0.850		0.971	
Front-end	Compression Factor k			Compression Factor k		Compression Factor k	
	50%	25%	10%	50%	25%	50%	25%
SimPF (Max)	**0.722**	**0.722**	0.706	**0.853**	0.828	0.969	0.963
SimPF (Avg)	**0.721**	**0.721**	0.701	0.846	0.831	**0.971**	0.964
SimPF (Avg-Max)	**0.724**	**0.720**	0.705	0.849	0.823	**0.971**	0.961
SimPF (Spectral)	**0.727**	**0.722**	0.709	0.845	0.821	0.970	0.964
SimPF (Uniform)	**0.718**	**0.712**	0.682	0.846	0.819	**0.971**	0.961

Table 1. CNN10 evaluation results on DCASE 2019 Task1, ESC-50, and SpeechCommands datasets. Baseline indicates the CNN10 model without SimPFs. The accuracy values where SimPFs outperform or perform the same as the CNN10 baseline are in bold.

4. EXPERIMENTS AND RESULTS

4.1. Datasets

DCASE 2019 Task 1 [27] is an acoustic scene classification task, with a development set consisting of 10-second audio clips from 10 acoustic scenes such as airport and metro station. In the development set, 9185 and 4185 audio clips are used for training and validation, respectively. We will refer to this dataset as *DCASE19*.
ESC-50 [18] consists of 2000 five-second environmental audio clips. ESC-50 is a balanced dataset with 50 sound categories, including animal sounds, natural soundscapes, human sounds (non-speech), and ambient noises. Each sound class has 40 audio clips. The dataset is pre-divided into five folds for cross-validation.
SpeechCommands [3] contains 65K speech utterances from various speakers. Each utterance is one second long and belongs to one of 30 classes corresponding to a speech command such as "Go", "Stop", "Left", and "Down". We divided the datasets by a ratio of 80:10:10 for training, validation, and testing, respectively.
AudioSet [19] is a large-scale audio dataset with 527 sound classes in total. The audio clips are sourced from YouTube videos. The training set consists of 2 063 839 audio clips. The evaluation set has 85 487 test clips. We convert all audio clips to monophonic and pad the audio clips to ten seconds with silence if they are shorter than ten seconds.

4.2. Experiment setup

Baseline systems We evaluate our proposed approach using several off-the-shelf audio classification methods proposed in [7]. As for the evaluation of ESC-50, DCASE19, and SpeechCommands dataset, we use two baseline models, CNN10 and MobileNetV2. On AudioSet, we conduct the experiment on CNN14 and MobileNetV2. CNN10 and CNN14 are both large-scale audio neural networks, and MobileNetV2 is designed with low complexity by multiply-add operations and fewer parameters. Hence, MobileNetV2 is suitable for on-device scenarios. We train all the models from scratch.
Implementation details We load the audio clips using the sampling rate as provided in the original dataset. The audio clip is converted to 64-dimensional log mel-spectrogram by the short-time Fourier transform with a window size of 1024 samples, a hop size of 320 samples, and a Hanning window. The baseline audio classification networks are optimized with the Adam optimizer with the learning rate 1×10^{-3}. The batch size is set to 32 and the number of epochs is 300, except for AudioSet where we run 15 epochs. Following [8], random SpecAugment [28] is used for data augmentation.

Models	Baseline	SimPF (50%)	SimPF (25%)
MobileNetV2	488.78M	243.80M	121.33M
CNN10	19.55G	9.76G	4.85G
CNN14	30.04G	14.97G	7.41G

Table 2. FLOPs analysis for CNN10, MobileNetV2, and CNN14 baseline models. The FLOPs are computed for one 10-second audio clip with a 44 kHz sampling rate.

Evaluation metrics Following [25], we use accuracy as the evaluation metric on ESC-50, DCASE19, and SpeechCommands datasets. As for the AudioSet dataset, we use mean average precision (mAP) to evaluate the performance of audio tagging.

4.3. Evaluation results and analysis

4.3.1. Computation cost analysis (FLOPs)

We analyze the impact on FLOPs reduction of our SimPFs on compression coefficients 50% and 25% for three baseline systems. Table 2 shows the FLOPs of the model to infer a 10-seconds audio clip with a sampling rate of 44 kHz. For our three baseline models, the compression ratio on the input spectrogram is roughly equivalent to the FLOPs reduction ratio. We refer to the spectrogram compression ratio as the FLOPs reduction ratio in the later experiment analysis.

4.3.2. DCASE19

For the CNN10 model, we evaluate the effectiveness of all our proposed SimPF with three compression factors: 50%, 25%, and 10% (as introduced in Section 3). The experimental results are shown in the left column in Table 1. Overall, *SimPF (Spectral)* performs best among all SimPFs candidates on three compression coefficient settings. Even though MobileNetV2 is smaller than CNN10, we find that *SimPF (Spectral)* can reduce the FLOPs by roughly 50% and 25% while still improving the classification accuracy by 1.7% and 1.2%, respectively. Even reducing the FLOPs by 90%, the classification accuracy only drops by 0.01%. For MobileNetV2, we evaluate the performance of two representative candidates *SimPF (Avg-Max)* and *SimPF (Spectral)* with three compression factor 50%, 25%, and 10%. We find similar trends to CNN10 model, as shown in the left column of Table 3. *SimPF (Avg-Max)* improves the accuracy by 1.2% on 25% setting and *SimPF (Spectral)* only sacrifices 0.3% accuracy on 10% setting. Experimental results on CNN10 and MobileNetV2 demonstrate the superior performance of our proposed SimPF for

Model (MobileNetV2 [7])	DCASE19			ESC-50		SpeechCommands	
Baseline	0.670			0.779		0.969	
Front-end	Compression Factor k			Compression Factor k		Compression Factor k	
	50%	25%	10%	50%	25%	50%	25%
SimPF (Avg-Max)	0.660	**0.682**	0.661	**0.784**	0.764	0.956	-
SimPF (Spectral)	**0.673**	0.668	0.667	**0.785**	0.772	0.953	-

Table 3. MobileNetV2 evaluation results on DCASE 2019 Task1, ESC-50, and SpeechCommands datasets. The accuracy values where SimPFs outperform the MobileNetV2 baseline are in bold.

Model (CNN14 [7])	AudioSet	
Baseline	0.432	
Front-end	Compression Factor k	
	50%	25%
SimPF (Avg-Max)	0.424	0.400
SimPF (Spectral)	0.426	0.397

Table 4. CNN14 evaluation results on AudioSet.

Model (MobileNetV2 [7])	AudioSet	
Baseline	0.318	
Front-end	Compression Factor k	
	50%	25%
SimPF (Avg-Max)	0.312	0.297
SimPF (Spectral)	0.314	0.293

Table 5. MobileNetV2 evaluation results on AudioSet.

acoustic scene classification, also indicating the highly redundant information in the acoustic scene data.

4.3.3. *ESC-50*

For CNN10 model, we evaluate the performance of all our proposed SimPF methods with two compression factors: 50% and 25%. The experimental results are shown in the middle column of Table 1. On the 50% setting, the best candidate *SimPF (Max)* achieves the accuracy improvement by 0.3%. On the 25% setting, the best candidate *SimPF (Avg)* reduces an accuracy by 1.9%. For MobileNetV2 model, we evaluate the performance of two representative candidates *SimPF (Avg-Max)* and *SimPF (Spectral)* with two compression factor 50%, 25%, as shown in the middle column of Table 2. *SimPF (Spectral)* performs better in these two different compression coefficient settings. Specifically, *SimPF (Spectral)* improves the classification accuracy by 0.6% on the 50% setting, and slightly decreases the accuracy by 0.7% on the 25% setting. The performance gain of SimPFs on ESC-50 is not as good as that on DCASE19 but is still decent in terms of the trade-off between the accuracy and FLOPs.

4.3.4. *SpeechCommands*

For CNN10 model, we evaluate all our proposed SimPFs with two compression factors 50% and 25%. The results are shown in the right column of Table 1. On the 50% setting, *SimPF (Avg)*, *SimPF (Avg-Max)*, *SimPF (Max)*, and *SimPF (Uniform)* achieve the equivalent performance as the baseline system. On the 25% setting, the best two candidates *SimPF (Avg)* and *SimPF (Spectral)* reduce the accuracy only by 0.7%. For the MobileNetV2 model, we evaluate the performance of two representative candidates *SimPF (Avg-Max)* and *SimPF (Spectral)* with compression factor at 50% (25% setting is not available[2]), as shown in the right column of Table 2. The best candidate *SimPF (Avg-Max)* decreases the accuracy by 1.3%. Evaluation results on SpeechCommands show that our proposed method is useful for short-utterance speech data.

4.3.5. *AudioSet*

We evaluate the performance of two representative candidates *SimPF (Avg-Max)* and *SimPF (Spectral)* with two compression factors 50% and 25%, for the CNN10 model, as shown in Table 3. On the 50% setting, *SimPF (Spectral)* only reduces the mAP by 0.8%, on the 25% setting, *SimPF (Avg-Max)* reduces the mAP by 3.2%. Similar results we obtained for the MobileNetV2 model, as shown in Table 4. On the 50% setting, *SimPF (Spectral)* only reduces the mAP by 0.4%, on the 25% setting, *SimPF (Avg-Max)* reduces the mAP by 2.1%. SimPF can roughly reduce 50% computation cost with a negligible mAP drop within 1%. Even though tagging AudioSet is a more challenging task as compared with classification for other datasets, SimPFs achieve a promising trade-off between computation cost and mAP.

5. CONCLUSION

In this paper, we have presented a family of simple pooling front-ends (SimPFs) for efficient audio classification. SimPFs utilize non-parametric pooling methods (e.g., max pooling) to eliminate the temporally redundant information in the input mel-spectrogram. SimPFs achieve a substantial improvement in computation efficiency for off-the-shelf audio neural networks with negligible degradation or considerable improvement in classification performance on four audio datasets. In future work, we will study parametric pooling audio front-ends to adaptively reduce audio spectrogram redundancy.

6. ACKNOWLEDGMENT

This work is partly supported by UK Engineering and Physical Sciences Research Council (EPSRC) Grant EP/T019751/1 “AI for Sound”, a Newton Institutional Links Award from the British Council (Grant number 623805725), British Broadcasting Corporation Research and Development (BBC R&D), a PhD scholarship from the Centre for Vision, Speech and Signal Processing (CVSSP), Faculty of Engineering and Physical Science (FEPS), University of Surrey, and a Research Scholarship from the China Scholarship Council (CSC). For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

[2]The 25% SimPF setting is not available as 25% of one-second speech clip is too short for MobileNetV2 to process.

References

[1] J. Sun, X. Liu, X. Mei, J. Zhao, M. D. Plumbley, V. Kılıç, and W. Wang, "Deep neural decision forest for acoustic scene classification," *arXiv preprint:2203.03436*, 2022.

[2] A. Mesaros, T. Heittola, T. Virtanen, and M. D. Plumbley, "Sound event detection: A tutorial," *IEEE Signal Processing Magazine*, vol. 38, no. 5, pp. 67–83, 2021.

[3] P. Warden, "Speech Commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint:1804.03209*, 2018.

[4] R. Radhakrishnan, A. Divakaran, and A. Smaragdis, "Audio analysis for surveillance applications," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2005, pp. 158–161.

[5] Y.-T. Peng, C.-Y. Lin, M.-T. Sun, and K.-C. Tsai, "Healthcare audio event classification using hidden Markov models and hierarchical hidden Markov models," in *IEEE International Conference on Multimedia*, 2009, pp. 1218–1221.

[6] S. Kiranyaz, A. F. Qureshi, and M. Gabbouj, "A generic audio classification and segmentation approach for multimedia indexing and retrieval," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 3, pp. 1062–1081, 2006.

[7] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[8] Y. Gong, Y.-A. Chung, and J. Glass, "PSLA: Improving audio tagging with pretraining, sampling, labeling, and aggregation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3292–3306, 2021.

[9] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 121–125.

[10] Y. Xiao, X. Liu, J. King, A. Singh, E. S. Chng, M. D. Plumbley, and W. Wang, "Continual learning for on-device environmental sound classification," *arXiv preprint:2207.07429*, 2022.

[11] A. Singh and M. D. Plumbley, "A passive similarity based CNN filter pruning for efficient acoustic scene classification," *arXiv preprint:2203.15751*, 2022.

[12] A. Singh, J. A. King, X. Liu, W. Wang, and M. D. Plumbley, "Low-complexity CNNs for acoustic scene classification," *arXiv preprint:2208.01555*, 2022.

[13] H. Futami, H. Inaguma, S. Ueno, M. Mimura, S. Sakai, and T. Kawahara, "Distilling the knowledge of BERT for sequence-to-sequence ASR," *arXiv preprint:2008.03822*, 2020.

[14] L. Lu, M. Guo, and S. Renals, "Knowledge distillation for small-footprint highway networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4820–4824.

[15] K. Choi, M. Kersner, J. Morton, and B. Chang, "Temporal knowledge distillation for on-device audio classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 486–490.

[16] J. Liang, T. Zhang, and G. Feng, "Channel compression: Rethinking information redundancy among channels in CNN architecture," *IEEE Access*, vol. 8, pp. 147 265–147 274, 2020.

[17] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2018, pp. 9–13.

[18] K. J. Piczak, "ESC: dataset for environmental sound classification," in *Proceedings of the 23rd ACM International Conference on Multimedia*, 2015, pp. 1015–1018.

[19] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 776–780.

[20] A. Singh, P. Rajan, and A. Bhavsar, "SVD-based redundancy removal in 1-D CNNs for acoustic scene classification," *Pattern Recognition Letters*, vol. 131, pp. 383–389, 2020.

[21] 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 preprint:1704.04861*, 2017.

[22] J. T. Geiger, B. Schuller, and G. Rigoll, "Large-scale audio feature extraction and SVM for acoustic scene classification," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2013, pp. 1–4.

[23] S. Zubair, F. Yan, and W. Wang, "Dictionary learning based sparse coefficients for audio classification with max and average pooling," *Digital Signal Processing*, vol. 23, no. 3, pp. 960–970, 2013.

[24] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with SincNet," in *IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 1021–1028.

[25] N. Zeghidour, O. Teboul, F. d. C. Quitry, and M. Tagliasacchi, "LEAF: A learnable frontend for audio classification," *arXiv:2101.08596*, 2021.

[26] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[27] T. Heittola, A. Mesaros, and T. Virtanen, "TAU Urban Acoustic Scenes 2019, Development dataset," Mar. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2589280

[28] D. S. Park, Y. Zhang, C.-C. Chiu, Y. Chen, B. Li, W. Chan, Q. V. Le, and Y. Wu, "SpecAugment on large scale datasets," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6879–6883.