DETERMINING RATIO OF PRUNABLE CHANNELS IN MOBILENET BY SPARSITY FOR ACOUSTIC SCENE CLASSIFICATION

Yiqiang Cai, Shengchen Li

School of Advanced Technology
Xi’an Jiaotong-Liverpool University
111 Ren’ai Road, Suzhou, China

ABSTRACT
MobileNet is widely used for Acoustic Scene Classification (ASC) in embedded systems. Existing works reduce the complexity of ASC algorithms by pruning some components, e.g. pruning channels in the convolutional layer. In practice, the maximum proportion of channels being pruned, which is defined as Ratio of Prunable Channels ($R_{PC}$), is often decided empirically. This paper proposes a method that determines the $R_{PC}$ by simple linear regression models related to the Sparsity of Channels ($S_C$) in the convolutional layers. In the experiment, $R_{PC}$ is examined by removing inactive channels until reaching a knee point of performance decrease. Simple methods for calculating the $S_C$ of trained models and resulted $R_{PC}$ are proposed. The experiment results demonstrate that 1) the decision of $R_{PC}$ is linearly dependent on $S_C$ and the hyper-parameters have a little impact on the relationship; 2) MobileNet shows a high sensitivity and stability on proposed method.

Index Terms— pruning, sparsity, convolutional neural network, acoustic scene classification

1. INTRODUCTION
Recently, attempts to deploy Acoustic Scene Classification (ASC) algorithms in mobile devices like microphone [1], headphone [2] and audiophone [3] are in vogue. Various deep learning models are involved, especially the MobileNet [4]. While mobile devices mostly require limited system complexity, further compression of MobileNet becomes a vitally important research topic to reduce the consumption of resources.

The state-of-the-art compression methods for neural networks comprises low rank approximation [5], weight quantization [6], neural network pruning [7] and knowledge distillation [8]. Among methods mentioned, neural network pruning is commonly used, which prunes the components of a neural network with little contribution to the final performance. Pruning methods can be broadly divided into two subcategories based on the type of components being pruned, unstructured pruning and structured pruning. Unstructured pruning [9], so called weight pruning at first, prunes the sole weight in the scope of whole network, and usually requires dedicated hardware. By contrast, structured pruning [10] avoids the problem by pruning the structured components of a neural network, like the channels in convolutional layers. Lately, Liu et al. [11] introduced the channel-level scaling factors to distinguish the important and unimportant channels, which is leveraged in our experiments.

However, the maximum percentage of channels to be pruned, so called Ratio of Prunable Channels ($R_{PC}$), is often determined empirically, which takes additional time and resources. There are some existing works [12] [13] attempting to automatically decide the $R_{PC}$ by setting up a tolerance limit of performance drop to stop the pruning process. However, setting up a fixed number lacks flexibility when being applied to different neural network architectures. Therefore, we aim to propose a new method to find an adaptive $R_{PC}$ for different network architectures.

In this paper, we focus on deciding the $R_{PC}$ directly from the statistical representation of weights in convolutional layers. In previous work [11], we observed that the Sparsity of Channels ($S_C$) in convolutional layers appeared some relevance with the pruning performance. Therefore, a method related to the scaling factors [11] is proposed to describe the $S_C$. In the experiment, we implement the architecture of MobileNet, and train sparse networks with different degree of regularization, then the $S_C$ of trained models is calculated. Next, channels are pruned empirically to discover the $R_{PC}$. Furthermore, the linear regression analysis is used to investigate the relationship between $R_{PC}$ and $S_C$. In addition, the effect of hyper-parameters of a single network architecture on the relationship is explored. Consequently, we draw a conclusion based on the experimental results.

The paper is organized as below. The Section 2 introduces the methods we use to obtain and analyze the data set of $S_C$ and $R_{PC}$. Technical details of experiments are elaborated in Section 3. Section 4 sets forth the results and analysis. Conclusion and perspective are drawn in Section 5.
2. METHODS

2.1. Network Architectures

MobileNet \cite{14} has been currently developed in many ASC tasks. It is invented for the scenario of deploying deep models in mobile devices, which achieved low complexity by using depthwise separable convolution \cite{14}.

The convolutional layers of MobileNet are commonly followed by a BN layer \cite{15}. The transformation of a BN layer can be denoted by Formulas (1) and (2).

\[
\hat{x} = \frac{x_{in} - \mu_c(B)}{\sqrt{\sigma_c^2(B) + \epsilon}},
\]

\[
x_{out} = \gamma \cdot \hat{x} + \beta
\]

where \(x_{in}\) and \(x_{out}\) respectively represent the input and output of the BN layer, \(\mu_c(B)\) and \(\sigma_c^2(B)\) separately indicate the mean and standard deviation of input on the scale of a mini-batch \(B\), \(\epsilon\) is an extremely small constant for numerical stability, \(\gamma\) and \(\beta\) are trainable parameters for scaling and shifting. Following \cite{11}, the channels with small value of \(\gamma\) are regarded as unimportant channels, which can be safely removed. In other words, the scaling factors \(\gamma\) of channels can be directly used to measure the significance of channels.

2.2. Sparsity

Training a sparse network before pruning is a common way to bring about a larger compression rate. We achieve sparsity at the level of channels by adding a L1-norm term on the scaling factors \(\gamma\) to the loss function, i.e. \(R(\gamma) = ||\gamma||_1\). As shown in Equation (3).

\[
\min_\theta \sum_{i=1}^{n} L(f(\theta; x_i), y_i) + \lambda R(\gamma)
\]

where \(L(\cdot)\) is the loss function of neural network \(f(\cdot)\), \(x_i\) and \(y_i\) respectively denote the input feature and label of the \(i\)th sample of the training set \(\{(x_i, y_i)\}_{i=1}^{n}\), \(\theta\) represent the parameters of network, \(\lambda\) is a hyper-parameter for adjusting the degree of regularization. The penalty term \(\lambda R(\gamma)\) forces the value of \(\gamma\) to near zero during training, the process is so called channel selection. However, a network may achieve different sparsities by controlling \(\lambda\) during training.

Therefore, a simple statistical method is proposed to calculate the \(S_C\) in the MobileNet. During the process of channel selection, the more number of scaling factors closing to zero indicates a sparser network. By regarding the channels with scaling factors \(\gamma\) less than 0.01 as unimportant channels, \(S_C\) is calculated through dividing the number of unimportant channels by the total number of channels. Let \(\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_n\}\), we get:

\[
S_C = \frac{|\{\gamma_i < 0.01| \gamma_i \in \Gamma\}|}{n}.
\]

2.3. Ratio of Prunable Channels

As show in Fig. 1 in real-world applications, ASC system is usually pruned to a knee point of accuracy loss, the pruning rate of which is defined as \(R_{PC}\). However, \(R_{PC}\) varies in network with different sparsities.

The knee of a curve is commonly defined as the point of maximum curvature. In this paper, we introduce a knee detection technique, Kneedle \cite{16}, to help determine the knee of pruning curves in MobileNet with different sparsities. It works as follows, for a discrete data set \(D = \{(x_i, y_i) \in \mathbb{R}|x_i, y_i \geq 0\}\), the points \((x_i, y_i)\) are firstly min-max normalized. Secondly, the data set of difference \(D_d = \{(x_{d_i}, y_{d_i})| x_{d_i} = x_i, y_{d_i} = y_i - x_i\}\) is computed. Thirdly, the local maxima of the difference curve \((x_{lmx_i}, y_{lmx_i})\) are discovered by \(x_{lmx_i} = x_{d_i}, y_{lmx_i} = y_{d_i}| y_{d_{i-1}} < y_{d_i}, y_{d_{i+1}} < y_{d_i}\). Finally, the knee will be detected as long as \((x_{d_i}, y_{d_i})\) decreases blow the threshold \(T_{lmx_i}\) before the next local maxima is reached. \(T_{lmx_i}\) is calculated by Formula (5), where \(\psi\) is a parameter to control the sensitivity.

\[
T_{lmx_i} = y_{lmx_i} - \psi \cdot \frac{\sum_{i=1}^{n-1} (x_{i+1} - x_n)}{n - 1}
\]

Fig. 1. Pruning Curve of MobileNet (Width=1). Selected examples for illustrating knee and \(R_{PC}\).

2.4. Experiment Design

The experiment is designed to reveal the linear relationship between \(S_C\) and \(R_{PC}\). It has been divided into 3 steps.

1) Training sparse networks. By controlling the hyper-parameter \(\lambda\) in Formula (1) at the training process, we are able to obtain networks with different degree of sparsity and calculating the sparsity \(S_C\) by Formula (4).

2) Pruning. Firstly, a trained model is iteratively pruned with a small fixed pruning rate of channels for each step. Then, we get a discrete data set of pruning rate and accuracy loss after evaluations. Finally, the knee is detected from the data set by using Kneedle and corresponding \(R_{PC}\) is found.

3) Linear
Data augmentation techniques play a crucial role in audio processing tasks, with the advantages of improving generalization abilities and reducing the effect of overfitting. We use Mixup [18] with $\alpha = 0.4$ and SpecAugmentation [19] with a frequency mask of 4 and a time mask of 40 in all experiments. All dropout layers in MobileNet are removed as data augmentation methods already provide some regularization.

### 3.3. Training

Four variants of MobileNet [4] with different widths are selected, separately 0.25, 0.5, 0.75 and 1. Meanwhile, another two CNN architectures are chosen for comparison, VGGNets [20] with respectively 11, 13, 16, 19 layers of depth, and ResNets [21] with respectively 11, 20, 29, 38 layers of depth. The scaling factors $\gamma$ are initialized by 0.5 to get a better performance according to [22] and [23]. All networks follow the same training setups, for 100 epochs with batch size 32, using Adam optimizer with learning rate to 0.1, momentum to 0.9, weight decay to 0.0001. In addition, the value of $\lambda$ in Equation 3 is adjusted before training, in order to get networks with various sparsities. It is worth mentioning if $\lambda$ is assigned a large value, the model will suffer a dramatic degeneration of performance, so sparse networks are trained with $\lambda$ from a tiny value to an upper limit (varies for different networks) for less than 3% accuracy loss comparing with the top performance. $S_C$ is calculated once after each sparse training as shown in the left of Table 1 following Equation (4).

### 3.4. Pruning

After training a model, as in [11], the channels are iteratively pruned for a 5% pruning rate, starting from the channels with lowest values of scaling factors $\gamma$. Meanwhile, the performance of the pruned model is evaluated after each pruning iteration, so as to obtain the discrete data set of pruning rate and accuracy loss. Then, the knee of pruning curve is detected by using Kneedle [16] and the corresponding $R_{PC}$ is revealed as shown in the middle of Table 1. Besides, we also fine-tune the model with the same optimizer as in training process for 5 epochs [9] at training set after pruning, to recover some performance loss. The fine-tuned model is also evaluated to get the $R_{PC}$ with fine-tuning as shown in the right of Table 1. It is worth mentioning that we repeat this approach for each CNN and observe the average performance decrease of all CNNs at $R_{PC}$ is approximately between 1% and 2%.

### 4. RESULTS

The analysis results are shown in Fig. 2 and Table 2. It can be easily seen that $R_{PC}$ linearly grows with the increase of $S_C$ in most cases. Generally, the slope of lines $m$ with fine-tune is smaller than that without fine-tune, which indicates $R_{PC}$ becomes less sensitive to $S_C$ when fine-tune is applied.
Table 2. Linear Regression of $S_C$ and $R_{PC}$. $m$ and $b$ are slope and bias of the linear equation $R_{PC} = m \cdot S_C + b$, can be solved by Equation (6),(7). $R^2$ indicates the goodness of fit. The row with bold font for each architecture means all data of 4 variants of a CNN is involved.

In the case of pruning models without fine-tuning, $R_{PC}$ of all 3 networks are linearly dependent on $S_C$ with $R^2$ extremely closing to 1. In addition, the hyper-parameters of network, depth and width, hardly influence the linear relationship. Therefore, $R_{PC}$ can be safely calculated by $R_{PC} = 0.9 \cdot S_C + 0.1$ for MobileNets, VGGNets and ResNets while pruning without fine-tuning. In the other case of pruning with fine-tuning, the slope of line $m = 0.7$ represents that $R_{PC}$ is relatively most sensitive to $S_C$, while VGGNets preserve the least sensitivity. Moreover, the line of MobileNets retain the highest goodness of fit with $R^2 = 0.852$. Specially, with the increase of depth of ResNet, the slope of line $m$ decreases from 0.8 to 0.3 and $R^2$ drops from 0.917 to 0.550, in other words, the hyper-parameters do more impacts to the lines of ResNets. In summary, the linear model of MobileNets behaves better than that of other two architectures when fine-tune is used.

5. CONCLUSION

This paper proposes a new method to determine the Ratio of Prunable Channels ($R_{PC}$) for MobileNet in ACS scenario by linear regressions related to the Sparsity of Channels ($S_C$). In the case of only pruning, we demonstrate that $R_{PC}$ is linearly dependent on $S_C$ regardless of architectures, and the relationship can be approximately denoted by formula $R_{PC} = 0.9 \cdot S_C + 0.1$. Meanwhile, it shows that the hyper-parameters, i.e. depth and width, have little influence on the relationship. In the other case of pruning with fine-tuning, it is recommended to use MobileNet as the linear model of MobileNet explains $R_{PC}$ with the highest sensitivity and stability. In future works, the relationship between sparsity and performance of networks is a topic worth a deeper dig.

6. REFERENCES

[1] Jianlei Zhang, Xumin Zheng, Wenfeng Shen, Dingqian Zhou, Feng Qiu, and Huiran Zhang, “A mic-based acceleration model of deep learning,” in 2016 International Conference on Audio, Language and Image Processing (ICALIP), 2016, pp. 608–614.

[2] Inderpreet Singh, Naveen Aggarwal, and Vijay Paul Singh, “Acousense: A deep learning based real-time acoustic triggering system,” in 2021 IEEE Bombay Section Signature Conference (IBSSC), 2021, pp. 1–6.

[3] V.S. Vivek, S Vidhya, and P Madhanmohan, “Acoustic scene classification in hearing aid using deep learning,” in 2020 International Conference on Communi-
cation and Signal Processing (ICCSP), 2020, pp. 0695–0699.

[4] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” arXiv preprint arXiv:1704.04861, 2017.

[5] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman, “Speeding up convolutional neural networks with low rank expansions,” arXiv preprint arXiv:1405.3866, 2014.

[6] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen, “Compressing neural networks with the hashing trick,” in International conference on machine learning. PMLR, 2015, pp. 2285–2294.

[7] Yann LeCun, John Denker, and Sara Solla, “Optimal brain damage,” Advances in neural information processing systems, vol. 2, 1989.

[8] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al., “Distilling the knowledge in a neural network,” arXiv preprint arXiv:1503.02531, vol. 2, no. 7, 2015.

[9] Song Han, Jeff Pool, John Tran, and William Dally, “Learning both weights and connections for efficient neural network,” Advances in neural information processing systems, vol. 28, 2015.

[10] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li, “Learning structured sparsity in deep neural networks,” Advances in neural information processing systems, vol. 29, 2016.

[11] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang, “Learning efficient convolutional networks through network slimming,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2736–2744.

[12] Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay P Namboodiri, “Play and prune: Adaptive filter pruning for deep model compression,” arXiv preprint arXiv:1905.04446, 2019.

[13] Guan Li, JunPeng Wang, Han-Wei Shen, Kaixin Chen, Guihua Shan, and Zhonghua Lu, “Cnnpruner: Pruning convolutional neural networks with visual analytics,” IEEE Trans. Vis. Comput. Graph., vol. 27, no. 2, pp. 1364–1373, 2021.

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

[15] Sergey Ioffe and Christian Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in International conference on machine learning. PMLR, 2015, pp. 448–456.

[16] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan, “Finding a “kneedle” in a haystack: Detecting knee points in system behavior,” in 2011 31st International Conference on Distributed Computing Systems Workshops, 2011, pp. 166–171.

[17] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen, “A multi-device dataset for urban acoustic scene classification,” in Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018), November 2018, pp. 9–13.

[18] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz, “mixup: Beyond empirical risk minimization,” arXiv preprint arXiv:1710.09412, 2017.

[19] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le, “SpecAugment: A simple data augmentation method for automatic speech recognition,” in Interspeech 2019. sep 2019, ISCA.

[20] Karen Simonyan and Andrew Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv 1409.1556, 09 2014.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

[23] Sergey Zagoruyko and Nikos Komodakis, “Wide residual networks,” arXiv preprint arXiv:1605.07146, 2016.