EFFICIENT SIMILARITY-BASED PASSIVE FILTER PRUNING FOR COMPRESSING CNNS

Arshdeep Singh, Mark D. Plumbley

Centre for Vision, Speech and Signal Processing (CVSSP)
University of Surrey, UK
Email: {arshdeep.singh, m.plumbley}@surrey.ac.uk

ABSTRACT

Convolution neural networks (CNNs) have shown great success in various applications. However, the computational complexity and memory storage of CNNs is a bottleneck for their deployment on resource-constrained devices. Recent efforts towards reducing the computation cost and the memory overhead of CNNs involve similarity-based passive filter pruning methods. Similarity-based passive filter pruning methods compute a pairwise similarity matrix for the filters and eliminate a few similar filters to obtain a small pruned CNN. However, the computational complexity of computing the pairwise similarity matrix is high, particularly when a convolutional layer has many filters. To reduce the computational complexity in obtaining the pairwise similarity matrix, we propose to use an efficient method where the complete pairwise similarity matrix is approximated from only a few of its columns by using a Nyström approximation method. The proposed efficient similarity-based passive filter pruning method is 3 times faster and gives same accuracy at the same reduction in computations for CNNs compared to that of the similarity-based pruning method that computes a complete pairwise similarity matrix. Apart from this, the proposed efficient similarity-based pruning method performs similarly or better than the existing norm-based pruning methods. The efficacy of the proposed pruning method is evaluated on CNNs such as DCASE 2021 Task 1A baseline network and a VGGish network designed for acoustic scene classification.

Index Terms— Acoustic scene classification, pruning, VGGish, DCASE.

1. INTRODUCTION

Compressing convolutional neural networks (CNNs) is crucial to reduce their computational complexity and memory storage for efficient deployment on resource-constrained devices [1], despite state-of-the-art performances of CNNs in various applications [2]. Typically, CNNs have redundant parameters such as weights or filters, which yield only extra computations and storage without contributing much to the performance of the underlying task [3, 4]. For example, Singh et al. [5, 6] found that 73% of the filters in SoundNet that do not provide discriminative information across different acoustic scene classes, and eliminating such filters gives similar performance compared to that of using all filters in SoundNet. Thus, the compression of CNNs has recently drawn significant attention from the research community.

Recent efforts towards compressing CNNs involve filter pruning methods [7, 8, 9] that eliminate some of the filters in CNNs based on their importance. The importance of the CNN filters is measured in an active or in a passive manner. Active filter pruning methods involve a dataset. For example, some methods [7, 10, 11, 12, 13] use feature maps which are outputs produced by the filters corresponding to a set of examples, and apply metrics such as entropy or the average percentage of zeros on the feature maps to quantify the filter importance. On the other hand, passive filter pruning methods [14, 15] use only parameters of the filters, such as an absolute sum of the weights in the filters, to quantify the filter importance. The passive filter pruning methods do not involve a dataset to measure filter importance and therefore are easier to apply compared to active filter pruning methods. After eliminating filters from the CNNs, the pruned network is fine-tuned to regain some of the performance lost due to the filter elimination.

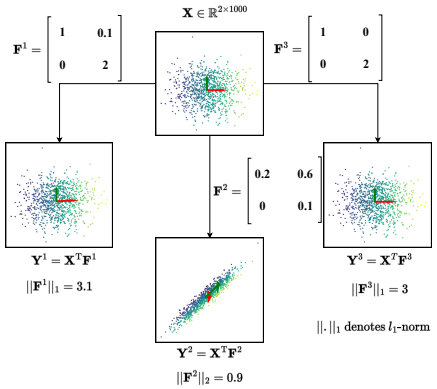


Fig. 1. An illustration of output produced in a convolution layer by three CNN filters, $\mathbf{F}^1$, $\mathbf{F}^2$ and $\mathbf{F}^3$, with a convolution operation on randomly generated data points, $\mathbf{X} \in \mathbb{R}^{2\times1000}$.

Previously, passive filter pruning methods used norm-based metrics such as l_1-norm [14], which is a sum of the absolute values of each weight in the filter, or l_2-distance of the filters from a geometric median of all filters [15] to quantify the importance of the filters. These norm-based methods use a "smaller-norm-less-important" criterion to eliminate filters. For example, a filter having a relatively high l_1-norm is considered more important than others. However, while selecting relatively high-norm filters as important, norm-based methods may ignore the redundancy among the high-norm filters. To illustrate this, we show outputs produced by three filters in Figure 1. Filters $\mathbf{F}^1$ and $\mathbf{F}^3$ have similar l_1-norm and produce similar outputs. However, selecting two important filters out of the three filters shown in Figure 1, the norm-based method selects filters $\mathbf{F}^1$ and $\mathbf{F}^3$ as important due to their relatively high norm, despite producing similar outputs, while it eliminates filter $\mathbf{F}^2$ that produces significantly different output than the other filters. Thus the diversity learned in the network may be ignored.

To capture diversity in the network, similarity-based methods

are employed that eliminate similar filters with an assumption that the similar filters produce similar or redundant outputs. For example, Kim et al. [16] perform clustering on filters and selects a filter from each cluster as important and eliminates the other filters. Singh et al. [17] measure similarity between filters by computing a pair-wise cosine distance for all filters and then eliminating a filter from a pair of similar filters. Such similarity-based methods give better performance compared to norm-based methods. However, similarity-based pruning methods involve a similarity matrix that takes $\mathcal{O}(n^2d)$ computations to compute for n filters having d parameters. Due to this, the computational complexity is high, particularly when there is a large number of filters in the convolutional layer.

In this work, we propose passive filter pruning method for CNNs to reduce their computational complexity and memory storage by using a Nyström approximation [18] to approximate the similarity matrix using only a few columns of the complete similarity matrix. We evaluate the proposed pruning framework on acoustic scene classification using two CNNs, DCASE 2021 Task 1A baseline network [1] and VGGish network [19].

The rest of this paper is organised as follows. Section 2 explains efficient similarity-based passive filter pruning method. Experimental setup is included in Section 3. Section 4 presents results and analysis. Finally, conclusion is included in Section 5.

2. EFFICIENT SIMILARITY-BASED PASSIVE FILTER PRUNING METHOD

Consider a set of n filters, $\mathbf{F}^l, 1 \leq l \leq n$ each of size $(w \times h \times c)$ with w is a width, h is a height and c is the number of channels, in a convolution layer of a CNN. Each filter is transformed to a 2D matrix of size $(d \times c)$ without loss of generality with $d = wh$. Next, we compute a Rank-1 approximation of the filter by performing singular value decomposition (SVD) on the transformed 2D filter. Next, a column $\in \mathbb{R}^d$ with unit norm from the Rank-1 approximation of $\mathbf{F}^l$ is chosen as a representative of the corresponding filter. Let $\mathbf{R} \in \mathbb{R}^{d\times n}$ denotes the filter representative matrix which is constructed by stacking the filter representatives of the n filters.

Given $\mathbf{R}$, we identify a small set of important filters out of total n filters in a given convolutional layer based on the similarity between the filters using the following two steps:

(Step 1) Approximating distance matrix: In the first step, we approximate the pairwise cosine distance matrix $\mathbf{Z} = \mathbf{1} - \mathbf{S}$, where $\mathbf{S} = \mathbf{R}^{\mathrm{T}}\mathbf{R} \in \mathbb{R}^{n\times n}$ denotes a pairwise similarity matrix for n filters.

We take a few columns of $\mathbf{S}$ to approximate the rest of the entries of $\mathbf{S}$ by using a Nyström approximation method [18]. Without loss of generality, the matrix $\mathbf{S}$ can be written as follows:

$$\mathbf{S} = \begin{bmatrix} \mathbf{W} & \mathbf{A}^{\mathrm{T}} \\ \mathbf{A} & \mathbf{B} \end{bmatrix} \text{ and } \mathbf{C} = \begin{bmatrix} \mathbf{W} \\ \mathbf{A} \end{bmatrix} \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{m\times m}$, $\mathbf{C} \in \mathbb{R}^{n\times m}$, $\mathbf{A} \in \mathbb{R}^{(n-m)\times m}$, and $m << n$.

The Nyström method approximates $\mathbf{S}$ by taking $\mathbf{C}$, m columns from $\mathbf{S}$, generating a rank-k approximation $\tilde{\mathbf{S}}$ of $\mathbf{S}$ given by,

$$\tilde{\mathbf{S}} = \mathbf{C}\mathbf{W}_k^{+}\mathbf{C}^{\mathrm{T}}, \tag{2}$$

where $\mathbf{W}_k$ is the best rank-k approximation of $\mathbf{W}$ for the Frobenius norm with $k \leq \text{rank}(\mathbf{W})$ and $\mathbf{W}_k^{+} = \sum_{j=1}^{k} \sigma_j^{-1}\mathbf{U}^j\mathbf{U}^{j^{\mathrm{T}}}$ denotes the pseudo-inverse of $\mathbf{W}_k$. $\mathbf{W}_k^{+}$ is obtained by performing SVD on $\mathbf{W} = \mathbf{U}\Sigma\mathbf{U}^{\mathrm{T}}$, where $\mathbf{U}$ is an orthonormal matrix, $\mathbf{U}^j$ is an j^{th} column of $\mathbf{U}$ and $\Sigma = \text{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ is a real diagonal matrix with $\sigma_1 \geq \sigma_2, \ldots, \sigma_m \geq 0$. The computational complexity needed to obtain $\tilde{\mathbf{S}}$ is $\mathcal{O}(m^3 + nmk)$. After obtaining $\tilde{\mathbf{S}}$, we compute $\tilde{\mathbf{Z}} = \mathbf{1} - \tilde{\mathbf{S}}$, as an approximation of $\mathbf{Z}$.

(Step 2) Obtaining important filters: Given $\tilde{\mathbf{Z}}$, we identify the closet filter corresponding to each filter. A filter from the closest filter pairs is then considered redundant and eliminated from the underlying convolution layer.

A summary of the overall framework is given in Algorithm 1.

Algorithm 1: Efficient similarity-based pruning algorithm to identify important filters in a convolution layer.

Data: Pair-wise similarity matrix of n filters with m filters, $\mathbf{C} = [\mathbf{W} \quad \mathbf{A}]^{\mathrm{T}} \in \mathbb{R}^{n\times m}$, $m << n$.
Result: Indices of important filters (Imp_list).
(Step 1): Obtaining distance matrix via approximating $\mathbf{S}$
$\mathbf{W} = \mathbf{U}\Sigma\mathbf{U}^{\mathrm{T}}$,
$\mathbf{W}_k^{+} = \sum_{j=1}^{k} \sigma_j^{-1}\mathbf{U}^j\mathbf{U}^{j^{\mathrm{T}}}$,
$\tilde{\mathbf{S}} = \mathbf{C}\mathbf{W}_k^{+}\mathbf{C}^{\mathrm{T}}$,
$\tilde{\mathbf{Z}} = \mathbf{1} - \tilde{\mathbf{S}}$. %Distance matrix
(Step 2): Identify important filter indices
Q= [ ], Imp_list =[ ], Red_list = [ ]
for $l \leq n$ **do**
  [q, D] = argmin{ $\tilde{\mathbf{Z}}[\, l, : - \{l\}]$ } %Identify the closet filter with index q to l^{th} filter with their distance D
  Q.append((l, q), D)
end
Q_sort = Sort(Q) %Sort Q based on the distance D
for $i \leq len(\text{Q})$ **do**
  id_imp = Q_sort[i][0] %important filter index
  id_red = Q_sort[i][1] %redundant filter index
  if id_imp $\notin$ Red_list **then**
    Imp_list.append(id_imp)
    Red_list.append(id_red)
  end
end

Obtaining pruned network and performing fine-tuning: After obtaining the important filters across different convolution layers using Algorithm 1, we retain the set of important filters and eliminate the other filters from the unpruned CNN to obtain a pruned network. Eliminating a filter from a given convolutional layer also removes the corresponding feature map produced by the filter and the associated channel of the filter in the following convolutional layer. Therefore, the computations in the next convolutional layer are also reduced in the pruned network.

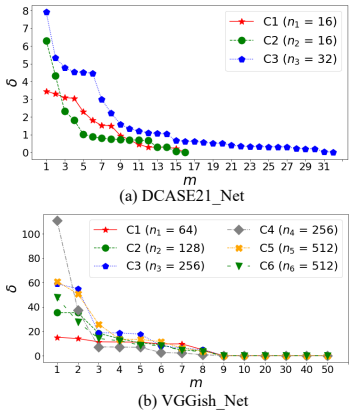
After removing filters, we perform fine-tuning which involves re-training of the pruned network to regain some of the lost performance due to the removal of the connection from the unpruned CNN. The codes for the proposed efficient pruning framework can be found at the link[1].

3. EXPERIMENTAL SETUP

We evaluate the proposed pruning framework on CNNs designed for acoustic scene classification (ASC). An overview of the unpruned CNNs is given below,

(a) DCASE21_Net: DCASE21_Net is a publicly available pre-trained network designed for DCASE 2021 Task 1A that is trained

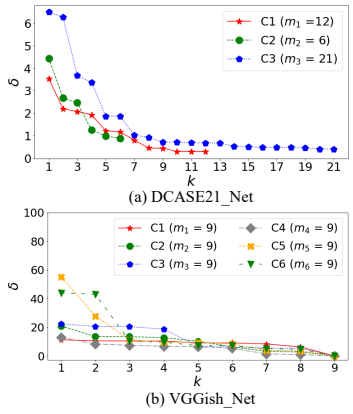
[1] https://github.com/Arshdeep-Singh-Boparai/Efficient_similarity_Pruning_Algo.git



(a) DCASE21_Net

(b) VGGish_Net

Fig. 2. Approximation error (δ) when m columns are selected out of n columns from the similarity matrix for different convolutional layers in (a) DCASE21_Net and (b) VGGish_Net . Here, the similarity matrix is computed using rank-k approximation with $k = m$.



(a) DCASE21_Net

(b) VGGish_Net

Fig. 3. Approximation error (δ) when the similarity matrix is generated with rank-k approximation by varying k using fixed number of columns (m) across various convolutional layers for (a) DCASE21_Net having m_1 =12 for C1, m_2 = 6 for C2 and m_3 =21 for C3 and (b) VGGish_Net where m = 9 for all C1 to C6 layers.

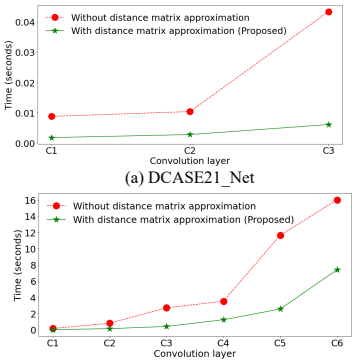
using TAU Urban Acoustic Scenes 2020 Mobile development dataset (we denote "DCASE-20") to classify 10 different acoustic scenes [1]. The input to the network is a log-mel spectrogram of size (40 × 500) corresponding to a 10s audio clip. DCASE21_Net is trained using the Adam optimizer with cross-entropy loss function for 200 epochs. The trained network has 46,246 parameters and requires approximately 287M multiply-accumulate operations (MACs) during inference corresponding to 10-second-length audio clip, and gives 48.58% accuracy on the DCASE-20 development validation dataset. DCASE21_Net consists of three convolutional layers (termed as C1 to C3) and one fully connected layer. C1 has $n_1 = 16$, C2 has $n_2 = 16$ and C3 has $n_3 = 32$ filters.

(b) VGGish_Net: This is built using a publicly available pre-trained VGGish network [19] followed by a dense and a classification layer. We train VGGish_Net on the TUT Urban Acoustic Scenes 2018 development ("DCASE-18") training dataset [20] to classify 10 different acoustic scenes using Adam optimizer with cross-entropy loss function for 200 epochs. The input to the VGGish_Net is a log-mel spectrogram of size (96 × 64) computed corresponding to a 960ms audio segment from a whole 10s audio scene. The VGGish_Net has approximately 55.361M parameters and requires 903M MACs during inference corresponding to an audio clip of 960ms and gives 64.69% accuracy on 10s audio scene for DCASE-18 development validation dataset. VGGish_Net has six convolution layers (termed as C1 to C6). The number of filters in each convolutional layers are {64, 128, 256, 256, 512, 512} respectively.

For i^{th} convolutional layer, we approximate the distance matrix $\tilde{\mathbf{Z}}_{m_i,k_i}$ using first 1 to m_i columns of the similarity matrix $\mathbf{S}_i$ and approximating the similarity matrix by rank-k_i approximation where $1 \leq k_i \leq m_i$. To measure the effectiveness of the approximation, we compute an approximation error $\delta_i = ||\mathbf{Z}_i - \tilde{\mathbf{Z}}_{m_i,k_i}||_2$ at different values of m_i and k_i.

To obtain the pruned network, we identify a set of important filters by computing $\tilde{\mathbf{Z}}_{m_i,k_i}$ at m_i and k_i, where $\delta_i < 1$. Fine-tuning of the pruned network is performed with similar conditions such as loss function, optimizer as used for training the unpruned network except for 100 epochs.

Performance metrics: We analyse a total time required to obtain the set of important filters for all convolutional layer. The total pruning time is computed after running the pruning algorithm for 10K times



(a) DCASE21_Net

(b) VGGish_Net

Fig. 4. Total pruning time to obtain important set of filters using Algorithm 1 for various convolutional layers, with and without approximating the distance matrix.

and an average of the total pruning time is reported. To measure the performance of the pruned network, we compute accuracy, the number of MACs per inference and the number of parameters. The accuracy of the pruned network is computed after fine-tuning the pruned network independently for 5 times and we report the average accuracy.

Other methods for comparison: We compare the proposed Algorithm 1 with existing norm-based pruning methods such as an l_1-norm [14] method and a geometric median (GM) method [15], feature map based active filter pruning methods such as HRank [7] and Energy-aware [11], and a similarity-based pruning method [17] that first computes complete cosine distance matrix, and then uses Step 2 of the Algorithm 1 to compute important set of filters for a given convolutional layer.

4. RESULTS AND ANALYSIS

Figure 2 shows the approximation error when m_i columns are selected out of n_i columns from the similarity matrix and the similarity matrix is approximated using rank-k_i approximation with $k_i = m_i$ for different convolutional layers in DCASE21_Net and VGGish_Net. We observe that selecting few columns from the similarity

Table 1. Comparison of accuracy, multiply-accumulate operations (MACs), parameters, pruning time to compute pruned network and memory required to store feature maps or filters to perform pruning. For HRank [7] and Energy-aware [11] pruning methods, we randomly selected 500 examples from the underlying dataset to generate feature maps and perform pruning.

Network	Pruning Method	Data & feature maps used in Pruning	Pruning Time (Seconds)	Feature map/filter storage	Accuracy (%)	Parameters	MACs
DCASE21_Net	No pruning (Baseline)	-	-		48.58	46246	286M
on	HRank [7]	✓	23	1.26GB	47.24	24056	139M
DCASE-20	Energy-aware [11]	✓	21	1.26GB	47.00	—"—	—"—
(Input: $40 \times 500 \times 1$)	l_1-norm [14]	×	0.0072	0.15MB	44.42	—"—	—"—
	GM [15]	×	0.010	0.15MB	45.84	—"—	—"—
	Similarity-based [17]	×	0.063	0.15MB	45.54	—"—	—"—
	Proposed (Efficient similarity-based)	×	0.011	0.15MB	45.54	—"—	—"—
VGGish_Net	No pruning (Baseline)	-	-	-	64.69	55M	903M
on	HRank [7]	✓	53	1.6GB	63.22	42.89M	595M
DCASE-18	Energy-aware [11]	✓	51	1.6GB	62.55	—"—	—"—
(Input: $64 \times 96 \times 1$)	l_1-norm [14]	×	0.21	17MB	60.02	—"—	—"—
	GM [15]	×	0.42	17MB	59.71	—"—	—"—
	Similarity-based [17]	×	34.80	17MB	62.00	—"—	—"—
	Proposed (Efficient similarity-based)	×	11.70	17MB	62.00	—"—	—"—

matrix are sufficient to approximate the complete similarity matrix with $\delta_i < 1$. Also, the distance matrix ($\tilde{\mathbf{Z}}_{m_i,m_i}$) approximated by choosing m_i columns of the similarity matrix that gives $\delta_i < 1$ results in a same set of important filters as obtained using the complete distance matrix ($\mathbf{Z}_i$).

For DCASE21_Net as shown in Figure 2(a), we find that choosing $m_1 = 12$ out of $n_1 = 16$ columns for C1, $m_2 = 6$ out of $n_2 = 16$ columns for C2 and $m_3 = 21$ out of $n_3 = 32$ columns for C3 gives $\tilde{\mathbf{Z}}_{m_i,m_i} \approx \mathbf{Z}_i$. For VGGish_Net as shown in Figure 2(b), we find that choosing $m_i = 9$ out of n_i columns, where $n_i \in \{64, 128, 256, 256, 512, 512\}$ for C1 to C6 layers respectively, gives $\tilde{\mathbf{Z}}_{m_i,m_i} \approx \mathbf{Z}_i$.

Next, Figure 3 shows the approximation error as k_i varies from 1 to m_i across different convolutional layers of (a) DCASE21_Net, where $m_1 = 12$, $m_2 = 6$ and $m_3 = 21$ for C1, C2 and C3 layers respectively, and (b) VGGish_Net, where $m_i = 9$ across various convolutional layers. We obtain the same set of important filters using $\tilde{\mathbf{Z}}_{m_i,k_i}$ as that of $\mathbf{Z}_i$ with $\delta_i < 1$ when $k_1 = 9$ for C1, $k_2 = 6$ for C2 and $k_3 = 13$ for C3 layer in DCASE21_Net, and $k_i = 9$ for $1 \leq i \leq 6$ convolutional layers of VGGish_Net.

Figure 4 compares the total pruning time computed for each convolutional layer, when the distance matrix ($\tilde{\mathbf{Z}}_{m_i,k_i}$ with $\delta_i < 1$) is approximated using Algorithm 1: Step 1, and when the complete pairwise distance matrix ($\mathbf{Z}_i$) is computed without any approximation for (a) DCASE21_Net and (b) VGGish_Net.

The total pruning time is reduced by approximating the distance matrix compared to computing the complete pairwise distance matrix for various convolution layers. When the number of filters is large, for example, the C6 layer in VGGish_Net has 512 filters, the total pruning time reduces significantly with the distance matrix approximation to that of computing the complete distance matrix. On the other hand, when the number of filters is smaller, for example C1 layer of VGGish_Net has 64 filters or all convolutional layers in DCASE21_Net has $n_i \leq 32$, the total pruning time reduces marginally by approximating the distance matrix compared to that of computing the complete distance matrix.

Table 1 compares the performance metrics with the other methods. For DCASE21_Net, the pruned network obtained using the proposed pruning method reduces both the MACs and the parameters by approximately 50% at 3 percentage points drop in accuracy compared to the unpruned network. The total pruning time for l_1-norm method [14] is the smallest among other methods. However, the accuracy obtained using the l_1-norm method is 1 percentage points lesser than that of the proposed pruning method. The accuracy and the total pruning time for the geometrical median (GM) pruning method [15] is marginally better than that of the proposed pruning method. In contrast to the similarity-based pruning method [17], the proposed efficient similarity-based pruning method takes less total pruning time and gives similar accuracy.

For VGGish_Net, the pruned network obtained using the proposed pruning method reduces the MACs by 34%, and the parameters are reduced by 23% at 2.7 percentage points drop in the accuracy compared to that of the unpruned network. Even though the l_1-norm and the GM pruning methods take significantly less computations than the proposed pruning method, the proposed pruning method improves the accuracy of the pruned network by 2 percentage points compared to that of the l_1-norm and the GM pruning methods. In contrast to the similarity-based pruning method, the proposed efficient similarity-based pruning method is 3 times faster and gives the same accuracy.

In comparison to the active filter pruning methods [7, 11], the proposed pruning method requires significantly less memory storage and at least 5 times less computational time in obtaining pruned network at accuracy within 2 percentage points of accuracy as obtained using the active filter pruning methods.

5. CONCLUSION

This paper presents an efficient similarity-based passive filter pruning framework to reduce computational complexity and memory storage in CNNs. We show that using only a few columns of the similarity matrix is sufficient to approximate similarity matrix and is 3 times faster than computing the complete pairwise similarity matrix with no loss in accuracy. The proposed pruning method yields a pruned network that performs similarly or better than the existing norm-based pruning methods.

In future, we would like to improve the performance of the pruned network obtained using the proposed pruning framework to achieve a similar performance as that of the unpruned network by using better distance measures such as graph-based similarity between the filters. Also, reducing the number of fine-tuning epochs (e.g. < 100) to recover some of the performance lost due to filter elimination is a future goal to reduce overall computations.

6. ACKNOWLEDGEMENTS

This work was partly supported by Engineering and Physical Sciences Research Council (EPSRC) Grant EP/T019751/1 "AI for Sound (AI4S)". For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

7. REFERENCES

[1] Irene Martín-Morató, Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen, "Low-complexity acoustic scene classification for multi-device audio: Analysis of DCASE 2021 Challenge systems," *DCASE Workshop*, 2021.

[2] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al., "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[3] Misha Denil, Babak Shakibi, Laurent Dinh, M.A. Ranzato, and Nando De Freitas, "Predicting parameters in deep learning," *Advances in Neural Information Processing Systems*, pp. 2148–2156, 2013.

[4] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir, "On the computational efficiency of training neural networks," *Advances in Neural Information Processing Systems*, pp. 855–863, 2014.

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

[6] Arshdeep Singh, Padmanabhan Rajan, and Arnav Bhavsar, "Deep hidden analysis: A statistical framework to prune feature maps," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 820–824, 2019.

[7] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao, "HRank: Filter pruning using high-rank feature map," *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 1529–1538, 2020.

[8] Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu, and Weiyao Lin, "ThiNet: Pruning CNN filters for a thinner net," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2525–2538, 2018.

[9] Wolfgang Roth, Günther Schindler, Matthias Zöhrer, Lukas Pfeifenberger, Robert Peharz, Sebastian Tschiatschek, Holger Fröning, Franz Pernkopf, and Zoubin Ghahramani, "Resource-efficient neural networks for embedded systems," *arXiv preprint arXiv:2001.03048*, 2020.

[10] Jian-Hao Luo and Jianxin Wu, "An entropy-based pruning method for CNN compression," *arXiv preprint arXiv:1706.05791*, 2017.

[11] Seul-Ki Yeom, Kyung-Hwan Shim, and Jee-Hyun Hwang, "Toward compact deep neural networks via energy-aware pruning," *arXiv preprint arXiv:2103.10858*, 2021.

[12] Adam Polyak and Lior Wolf, "Channel-level acceleration of deep face representations," *IEEE Access*, vol. 3, pp. 2163–2175, 2015.

[13] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250*, 2016.

[14] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf, "Pruning filters for efficient ConvNets," *International Conference on Learning Representations*, 2017.

[15] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4340–4349, 2019.

[16] Mincheol Park, Woojeong Kim, and Suhyun Kim, "REPrune: Filter pruning via representative election," *arXiv preprint arXiv:2007.06932*, 2020.

[17] Arshdeep Singh and Mark D Plumbley, "A passive similarity based CNN filter pruning for efficient acoustic scene classification," *Interspeech (arXiv preprint arXiv:2203.15751)*, 2022.

[18] Petros Drineas, Michael W Mahoney, and Nello Cristianini, "On the Nyström method for approximating a Gram matrix for improved kernel-based learning," *Journal of Machine Learning Research*, vol. 6, no. 12, 2005.

[19] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al., "CNN architectures for large-scale audio classification," *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131–135, 2017.

[20] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen, "A multi-device dataset for urban acoustic scene classification," *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop*, pp. 9–13, November 2018.