WSNet: Compact and Efficient Networks Through Weight Sampling

Xiaojie Jin [1 2] **Yingzhen Yang** [2] **Ning Xu** [2] **Jianchao Yang** [3] **Nebojsa Jojic** [4] **Jiashi Feng** [1] **Shuicheng Yan** [5 1]

Abstract

We present a new approach and a novel architecture, termed WSNet, for learning compact and efficient deep neural networks. Existing approaches conventionally learn full model parameters independently and then compress them via *ad hoc* processing such as model pruning or filter factorization. Alternatively, WSNet proposes learning model parameters by sampling from a compact set of learnable parameters, which naturally enforces parameter sharing throughout the learning process. We demonstrate that such a novel weight sampling approach (and induced WSNet) promotes both weights and computation sharing favorably. By employing this method, we can more efficiently learn much smaller networks with competitive performance compared to baseline networks with equal numbers of convolution filters. Specifically, we consider learning compact and efficient 1D convolutional neural networks for audio classification. Extensive experiments on multiple audio classification datasets verify the effectiveness of WSNet. Combined with weight quantization, the resulted models are up to **180**× smaller and theoretically up to **16**× faster than the well-established baselines, without noticeable performance drop.

1. Introduction

Despite remarkable successes in various applications, deep neural networks (DNNs) usually suffer following two problems that stem from their inherent huge parameter space. First, most of state-of-the-art deep architectures are prone to over-fitting even when trained on large datasets (Simonyan & Zisserman, 2015; Szegedy et al., 2015). Secondly, DNNs usually consume large amount of storage memory and energy (Han et al., 2016), which makes it difficult to use them in devices with limited memory and power (such as portable devices or chips). Different from most existing works (Han et al., 2015; Li et al., 2017; Jaderberg et al., 2014; Lebedev et al., 2014; Hinton et al., 2015) on model compression and acceleration that ignore the strong dependencies among weights and learn filters independently based on existing network architectures, this paper proposes to explicitly enforce the parameter sharing among filters to more effectively learn compact and efficient deep networks.

In this paper, we propose a **W**eight **S**ampling deep neural network (*i.e.* WSNet) to significantly reduce both the model size and computation cost, achieving more than 100× smaller size and up to 16× speedup at negligible performance drop or even achieving better performance than the baseline (*i.e.* conventional networks that learn filters independently). Specifically, WSNet is parameterized by layer-wise *condensed filters* from which each filter participating in actual convolutions can be directly sampled, in both spatial and channel dimensions. Since condensed filters have significantly fewer parameters than independently trained filters as in conventional CNNs, learning by sampling from them makes WSNet a more compact model compared to conventional CNNs. In addition, to reduce the ubiquitous computational redundancy in convolving the overlapped filters and input patches, we propose an integral image based method to dramatically reduce the computation cost of WSNet in both training and inference. The integral image method is also advantageous because it enables weight sampling with different filter size and minimizes computational overhead to enhance the learning capability of WSNet.

In order to demonstrate the efficacy of WSNet, we conduct extensive experiments on challenging audio classification tasks. On each test dataset, including ESC-50 (Piczak, 2015a), UrbanSound8K (Salamon et al., 2014), DCASE (Stowell et al., 2015) and MusicDet200K (a self-collected dataset, as detailed in Section 4), WSNet significantly reduces the model size of the baseline by 100× with comparable or even higher classification accuracy. When compressing more than 180×, WSNet is only subject to negligible accuracy drop. At the same time, WSNet significantly reduces the computation cost (up to 16×). Such results strongly establish the capability of WSNet to learn compact and efficient networks. Last but not the least, we provide an intuitive method to extend WSNet from 1D CNNs to

[1]National University of Singapore, Singapore [2]Snap Inc. Research, Los Angeles, USA [3]Bytedance Inc., Menlo Park, USA [4]Microsoft Research, Redmond, USA [5]360 AI Institute, Beijing, China. Correspondence to: Xiaojie Jin <xjjin0731@gmail.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

2D CNNs. Experimental results on MNIST and CIFAR10 strongly evidence the potential capability of WSNet to learn efficient networks on 2D CNNs.

2. Related Works

2.1. Deep Model Compression and Acceleration

Recent works in network compression adopt weight pruning (Han et al., 2015; Collins & Kohli, 2014; Anwar et al., 2017; Lebedev & Lempitsky, 2016; Kim et al., 2015; Luo et al., 2017; Li et al., 2017), filter decomposition (Sindhwani et al., 2015; Denton et al., 2014; Jaderberg et al., 2014), hashed networks (Chen et al., 2015; 2016) and weight quantization (Han et al., 2016). However, although those works reduce model size, they also suffer from large performance drop. Bucilu et al. (2006) and Ba & Caruana (2014) are based on student-teacher approches which may be difficult to apply in new tasks since they require training a teacher network in advance. Denil et al. (2013) predicts parameters based on a few number of weight values. Jin et al. (2016) proposes an iterative hard thresholding method, but only achieve relatively small compression ratios. Gong et al. (2014) uses a binning method which can only be applied over fully connected layers. Hinton et al. (2015) compresses deep models by transferring the knowledge from pre-trained larger networks to smaller networks.

In terms of deep model acceleration, the factorization and quantization methods listed above can reduce computation latency in inference. FFT (Mathieu et al., 2013) and LCNN (Bagherinezhad et al., 2016) are also used to speed up computation in pratice. Comparatively, WSNet is superior because it learns networks that have both smaller model size and faster computation versus baselines.

2.2. Efficient Model Design

WSNet presents a class of novel models with the appealing properties of a small model size and small computation cost. Some recently proposed efficient model architectures include the class of Inception models (Szegedy et al., 2015; Ioffe & Szegedy, 2015; Chollet, 2016), the class of Residual models (He et al., 2016; Xie et al., 2017; Chen et al., 2017) and the factorized networks which use fully factorized convolutions. MobileNet (Howard et al., 2017) and Flattened networks (Jin et al., 2014) are based on factorization convolutions. ShuffleNet (Zhang et al., 2017) uses group convolution and channel shuffle to reduce computational cost. Compared with the above works, WSNet presents a new model design strategy which is more flexible and generalizable: the parameters in deep networks can be obtained conveniently from a more compact representation through the proposed weight sampling method.

3. Method

3.1. Notations

Before diving into the details, we first introduce the notations used in this paper. The traditional 1D convolution layer takes as input the feature map $\mathbf{F} \in \mathbb{R}^{T \times M}$ and produces an output feature map $\mathbf{G} \in \mathbb{R}^{T \times N}$ where (T, M, N) denotes the spatial length of input, the channel of input and the number of filters respectively. We assume that the output has the same spatial size as input which holds true by using zero padded convolution. The 1D convolution kernel $\mathbf{K}$ used in the actual convolution of WSNet has the shape of (L, M, N) where L is the kernel size. Let $\mathbf{k}_n, n \in \{1, \cdots N\}$ denotes a filter and $\mathbf{f}_t, t \in \{1, \cdots T\}$ denotes a input patch that spatially spans from t to $t + L - 1$, then the convolution assuming stride one and zero padding is computed as:

$$\mathbf{G}_{t,n} = \mathbf{f}_t \cdot \mathbf{k}_n = \sum_{l=0}^{L-1} \sum_{m=0}^{M-1} \mathbf{F}_{t+l,m} \times \mathbf{K}_{l,m,n}, \quad (1)$$

where $\cdot$ stands for the vector inner product. Note we omit the element-wise activation function to simplify the notation.

In WSNet, instead of learning each weight independently, $\mathbf{K}$ is obtained by sampling from a learned *condensed filter* $\mathbf{\Phi}$ which has the shape of (L^*, M^*). The goal of training WSNet is thus cast to learn more compact DNNs which satisfy the condition of $L^*M^* < LMN$. WSNet uses a condensed filter per convolutional layer. To quantize the advantage of WSNet in achieving compact networks, we define the *compactness* of $\mathbf{K}$ in a learned layer in WSNet w.r.t. the conventional layer with independently learned weights as:

$$\text{compactness} = \frac{LMN}{L^*M^*}. \quad (2)$$

In the following section, we demonstrate WSNet learn compact networks by sampling weights in two dimensions: the spatial dimension and the channel dimension.

3.2. Weight sampling

3.2.1. Along spatial dimension

In conventional CNNs, the filters in a layer are learned independently which presents two disadvantages. Firstly, the resulted DNNs have a large number of parameters, which impedes their deployment in computation resource constrained platforms. Second, such over-parameterization makes the network prone to overfitting and getting stuck in (extra introduced) local minima. To solve these two problems, a novel weight sampling method is proposed to efficiently reuse the weights among filters. Specifically, in each convolutional layer of WSNet, all convolutional filters $\mathbf{K}$ are sampled from the condensed filter $\mathbf{\Phi}$, as illustrated in Figure 1. By scanning the weight sharing filter with a window size of L

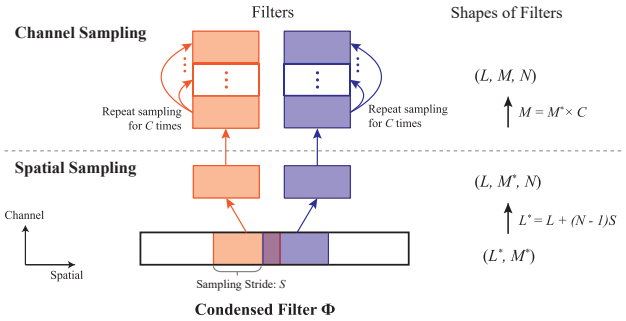

Figure 1: Illustration of WSNet that learns small condensed filters with weight sampling along two dimensions: spatial dimension (the bottom panel) and channel dimension (the top panel). The figure depicts procedure of generating two continuous filters (in pink and purple respectively) that convolve with input. In **spatial sampling**, filters are extracted from the condensed filter with a stride of S. In **channel sampling**, the channel of each filter is sampled repeatedly for C times to achieve equal with the input channel. Please refer to Section 3.2 for detailed explanations. All figures in this paper are best viewed in zoomed-in pdf.

and stride of S, we could sample out N filters with filter size of L. Formally, the equation between the filter size of the condensed filter and the sampled filters is:

$$L^* = L + (N-1)S. \tag{3}$$

The *compactness* along spatial dimension is $\frac{LM^*N}{L^*M^*} \approx \frac{L}{S}$. Note that since the minimal value of S is 1, the minimal value of L^* (*i.e.* the minimum spatial length of the condensed filter) is $L + N - 1$ and the maximal achievable compactness is therefore L.

3.2.2. Along Channel Dimension

Although it is experimentally verified that the weight sampling strategy could learn compact deep models with negligible loss of classification accuracy (see Section 4), the maximal compactness is limited by the filter size L, as mentioned in Section 3.2.1.

In order to seek more compact networks without such limitation, we propose a channel sharing strategy for WSNet to learn by weight sampling along the channel dimension. As illustrated in Figure 1 (top panel), the actual filter used in convolution is generated by repeating sampling for C times. The relation between the channels of filters before and after channel sampling is:

$$M = M^* \times C, \tag{4}$$

Therefore, the *compactness* of WSNet along the channel dimension achieves C. As introduced later in Experiments (Section 4), we observe that the repeated weight sampling along the channel dimension significantly reduces the model size of WSNet without significant performance drop. One notable advantage of channel sharing is that the maximum compactness can be as large as M (*i.e.* when the condensed filter has channel of 1), which paves the way for learning much more aggressively smaller models (*e.g.* more than $100\times$ smaller models than baselines). We attribute the effectiveness of channel sharing to reducing the redundancy along the channel dimension, especially in top layers. In general architecture design, the number of filter channels grows linearly with the layer depth. However, the spatial size of kernels becomes smaller or remains unchanged. This implies redundancy in higher layers mainly come from the channel dimension.

The above analysis for weight sampling along spatial/channel dimensions can be conveniently generalized from convolution layers to fully connected layers. For a fully connected layer, we treat its weights as a flattened vector with channel of 1, along which the spatial sampling (ref. Section 3.2.1) is performed to reduce the size of learnable parameters. For more details, please refer the supplementary material.

3.2.3. The Training of Condensed Filters

WSNet is trained from the scratch in a similar way to conventional deep convolutional networks by using standard error back-propagation. Since every weight $\mathbf{K}_{l,m,n}$ in the convolutional kernel $\mathbf{K}$ is sampled from the condensed filter $\mathbf{\Phi}$ along the spatial and channel dimension, the only difference is the gradient of $\mathbf{\Phi}_{i,j}$ is the summation of all gradients of weights that are tied to it. Therefore, by simply recording the position mapping $\mathcal{M} : (i,j) \to (l,m,n)$ from $\mathbf{\Phi}_{i,j}$ to all the tied weights in $\mathbf{K}$, the gradient of $\mathbf{\Phi}_{i,j}$ is calculated as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{\Phi}_{i,j}} = \sum_{s \in \mathcal{M}(i,j)} \frac{\partial \mathcal{L}}{\partial \mathbf{K}_s} \tag{5}$$

where $\mathcal{L}$ is the conventional cross-entropy loss function. In open-sourced machine learning libraries which represent computation as graphs, such as TensorFlow (Abadi et al., 2016), Equation (5) can be calculated automatically.

3.3. Denser Weight Sampling

The performance of WSNet might be adversely affected when the size of condensed filter is decreased aggressively (*i.e.* when S and C are large). To enhance the learning capability of WSNet, we could sample more filters from the condensed filter. Specifically, we use a smaller sampling stride $\bar{S}$ ($\bar{S} < S$) when performing spatial sampling. In order to keep the shape of weights unchanged in the following layer, we append a 1×1 convolution layer with the shape of $(1, \bar{n}, n)$ to reduce the channels of densely sampled filters. It is experimentally verified that denser weight sampling can effectively improve the performance of WSNet in Section 4. However, since it also brings extra parameters and

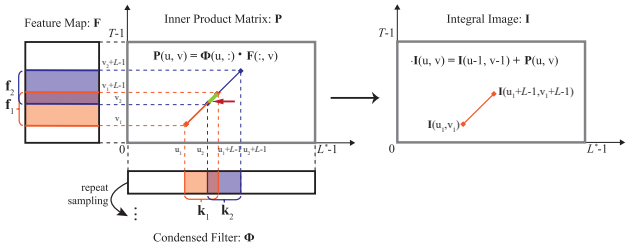

Figure 2: Illustration of efficient computation with integral image in WSNet. The inner product map $\mathbf{P} \in \mathbb{R}^{T \times L^*}$ calculates the inner product of each row in $\mathbf{F}$ and each column in $\mathbf{\Phi}$ as in Eq. (7). The convolution result between a filter $\mathbf{k}_1$ which is sampled from $\mathbf{\Phi}$ and the input patch $\mathbf{f}_1$ is then the summation of all values in the segment between (u, v) and $(u + L - 1, v + L - 1)$ in $\mathbf{P}$ (recall that L is the convolutional filter size). Since there are repeated calculations when the filter and input patch are overlapped, *e.g.* the green segment indicated by arrow when performing convolution between $\mathbf{k}_2$ and $\mathbf{s}_2$, we construct the integral image $\mathbf{I}$ using $\mathbf{P}$ according to Eq. (8). Based on $\mathbf{I}$, the convolutional results between any sampled filter and input patch can be retrieved directly in time complexity of O(1) according to Eq. (9), *e.g.* the results of $\mathbf{k}_1 \cdot \mathbf{s}_1$ is $\mathbf{I}(u_1 + L - 1, v_1 + L - 1) - \mathbf{I}(u_1 - 1, v_1 - 1)$. For notation definitions, please refer to Sec. 3.1. The comparisons of computation costs between WSNet and the baselines using conventional architectures are introduced in Section 3.4.

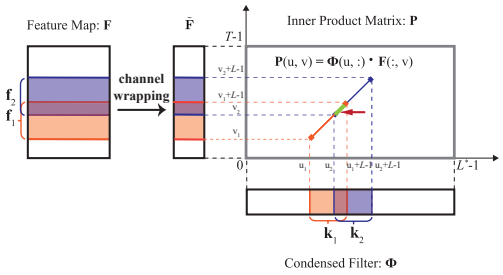

Figure 3: A variant of the integral image method used in practice which is more efficient than that illustrated in Figure 2. Instead of repeatedly sampling along the channel dimension of $\mathbf{\Phi}$ to convolve with the input $\mathbf{F}$, we wrap the channels of $\mathbf{F}$ by summing up C matrixes that are evenly divided from $\mathbf{F}$ along the channels, *i.e.* $\tilde{\mathbf{F}}(i, j) = \sum_{c=0}^{C-1} \mathbf{F}(i, j + cM^*)$. Since the channle of $\tilde{\mathbf{F}}$ is only $1/C$ of the channel of $\mathbf{F}$, the overall computation cost is reduced as demonstrated in Eq. (11).

computational cost to WSNet, denser weight sampling is only used in lower layers of WSNet whose filter number (n) is small. Besides, one can also conduct channel sampling on the added 1×1 convolution layers to further reduce their sizes.

3.4. Efficient Computation with integral image

According to Equation 1, the computation cost in terms of the number of multiplications and adds (*i.e.* Mult-Adds) in a conventional convolutional layer is:

$$TMLN \tag{6}$$

However, as illustrated in Figure 2, since all filters in a layer in WSNet are sampled from a condensed filter $\mathbf{\Phi}$ with stride S, calculating the results of convolution in the conventional way as in Eq. (1) incurs severe computational redundancies. Concretely, as can be seen from Eq. (1), one item in the ouput feature map is equal to the summation of L inner products between the row vector of $\mathbf{f}$ and the column vector of $\mathbf{k}$. Therefore, when two overlapped filters that are sampled from the condensed filter (*e.g.* $\mathbf{k}_1$ and $\mathbf{k}_2$ in Fig. 2) convolves with the overlapped input windows (*e.g.* $\mathbf{f}_1$ and $\mathbf{f}_2$ in Fig. 2)), some partially repeated calculations exist (*e.g.* the calculations highlight in green and indicated by arrow in Fig. 2). To eliminate such redundancy in convolution and speed-up WSNet, we propose a novel integral image method to enable efficient computation via sharing computations.

We first calculate an inner product map $\mathbf{P} \in \mathbb{R}^{T \times L^*}$ which stores the inner products between each row vector in the input feature map (*i.e.* $\mathbf{F}$) and each column vector in the condensed filter (*i.e.* $\mathbf{\Phi}$):

$$\mathbf{P}(u, v) = \begin{cases} \mathbf{F}_{u,:} \cdot \mathbf{\Phi}_{:,v}, & u \in [0, T-1] \text{ and } v \in [0, L^* - 1] \\ 0, & otherwise. \end{cases} \tag{7}$$

The integral image for speeding-up convolution is denoted as $\mathbf{I}$. It has the same size as $\mathbf{P}$ and can be conveniently obtained throught below formulation:

$$\mathbf{I}(u, v) = \begin{cases} \mathbf{I}(u-1, v-1) + \mathbf{P}(u, v), & u > 0, v > 0 \\ \mathbf{P}(u, 0), & v = 0 \\ \mathbf{P}(0, v), & u = 0 \end{cases} \tag{8}$$

Based on $\mathbf{I}$, all convolutional results can be obtained in time complexity of $O(1)$ as follows

$$\mathbf{G}_{t,n} = \mathbf{I}(t + L - 1, nS + L - 1) - \mathbf{I}(t - 1, nS - 1) \tag{9}$$

Recall that the n-th filter lies in the spatial range of $(nS, nS + L - 1)$ in the condensed filter $\mathbf{\Phi}$. Since $\mathbf{G} \in \mathbb{R}^{T \times N}$, it thus takes TN times of calculating Eq. (9) to get $\mathbf{G}$. In Eq. (7) $\sim$ Eq. (9), we omit the case of padding for clear description. When zero padding is applied, we can freely get the convolutional results for the padded areas even without using Eq. (9) since $\mathbf{I}(u, v) = \mathbf{I}(T, v - 1), u > T$.

Based on Eq. (7) $\sim$ Eq. (9), the computation cost of the proposed integral image method is

$$\underbrace{TML^*}_{\text{Eq. (7)}} + \underbrace{TL^*}_{\text{Eq. (8)}} + \underbrace{TN}_{\text{Eq. (9)}} = T(M+1)L^* + TN. \tag{10}$$

Note the computation cost of $\mathbf{P}$ (*i.e.* Eq. (7)) is the dominating term in Eq. (10). Based on Eq. (6), Eq. (10) and Eq. (3),

the theoretical acceleration ratio is

$$\frac{TMLN}{T(M+1)L^* + TN} \approx \frac{L}{S}$$

Recall that L is the filter size and S is the pre-defined stride when sampling filters from the condensed filter $\mathbf{\Phi}$ (ref. to Eq. (3)).

In practice, we adopt a variant of the above method to further boost the computation efficiency of WSNet, as illustrated in Fig 3. In Eq. (7), we repeat $\mathbf{\Phi}$ by C times along the channel dimension to make it equal with the channel of the input $\mathbf{F}$. However, we could first wrap the channels of $\mathbf{F}$ by accumulating the values with interval of L along its channel dimension to a thinner feature map $\tilde{\mathbf{F}} \in \mathbb{R}^{T \times M^*}$ which has the same channel number as $\mathbf{\Phi}$, *i.e.* $\tilde{\mathbf{F}}(i,j) = \sum_{c=0}^{C-1} \mathbf{F}(i, j+cM^*)$. Both Eq. (8) and Eq. (9) remain the same. Then the computational cost is reduced to

$$\underbrace{TM^*(C-1)}_{\text{channel warp}} + \underbrace{TM^*L^*}_{\text{Eq. (7)}} + \underbrace{TL^*}_{\text{Eq. (8)}} + \underbrace{TN}_{\text{Eq. (9)}} \tag{11}$$

where the first item is the computational cost of warping the channels of $\mathbf{F}$ to obtain $\tilde{\mathbf{F}}$. Since the dominating term (*i.e.* Eq. (7)) in Eq (11) is smaller than in Eq. (10), the overall computation cost is thus largely reduced. By combining Eq. (11) and Eq. (6), the theoretical acceleration compared to the baseline is

$$\frac{MLN}{M^*(C+L^*-1) + (L^*+N)} \tag{12}$$

Finally, we note that the integral image method applied in WSNet naturally takes advantage of the property in weight sampling: redundant computations exist between overlapped filters and input patches. Different from other deep model speedup methods (Sindhwani et al., 2015; Denton et al., 2014) which require to solve time-consuming optimization problems and incur performance drop, the integral image method can be seamlessly embeded in WSNet without negatively affecting the final performance.

3.5. An Intuitive Extension of WSNet from 1D convnet to 2D convnet

In this paper, we focus on WSNet with 1D convnets. Comprehensive experiments clearly demonstrate its advantages in learning compact and computation-efficient networks. We note that WSNet is general and can also be applied to build 2D convnets. In 2D convnets, each filter has three dimensions including two spatial dimensions (*i.e.* along X and Y directions) and one channel dimension. One straightforward extension of WSNet to 2D convnets is as follows: for spatial sampling, each filter is sampled out as a patch (with the same number of channels as in condensed filter) from condensed filter. Channel sampling remains the same as in 1D convnets, *i.e.* repeat sampling in the channel dimension of condensed filter. Following the notations for WSNet with 1D convnets (ref. to Sec. 3.1), we denote the filters in one layer as $\mathbf{K} \in \mathbb{R}^{w \times h \times M \times N}$ where (w, h, M, N) denote the width and height of each filter, the number of channels and the number of filters respectively. The condensed filter $\mathbf{\Phi}$ has the shape of (W, H, M^*). The relations between the shape of condensed filter and each sampled filter are:

$$\begin{aligned} W &= w + (\lceil \sqrt{N} \rceil - 1) S_w \\ H &= h + (\lceil \sqrt{N} \rceil - 1) S_h \\ M &= M^* \times C \end{aligned} \tag{13}$$

where S_w and S_h are the sampling strides along two spatial dimensions and C is the compactness of WSNet along channel dimension. The compactnesses (ref. to Eq. (2) for denifinition) of WSNet along spatial and channel dimension are $\frac{WH}{whN}$ and C respectively. However, such straightforward extension of WSNet to 2D convnets may not be optimum and we believe there are more sophisticated and effective methods for applying WSNet to 2D convnets and we would like to explore in our future work. Nevertheless, we conduct preliminary experiments on 2D convents using above intuitive extension and verify the effectiveness of WSNet in image classification tasks (on MNIST and CIFAR10).

4. Experiments

4.1. Experimental Settings

Datasets and baseline networks We collect a large-scale music detection dataset (MusicDet200K) from publicly available platforms (*e.g.* Facebook, Twitter, *etc.*) for conducting experiments. For fair comparison with previous literatures, we also test WSNet on three standard, publicly available datasets, *i.e* ESC-50, UrbanSound8K and DCASE. Due to space limit, please refer to the details of used datasets in supplementary material.

To test the scability of WSNet to different network architectures (*e.g.* whether having fully connected layers or not), two baseline networks are used in comparision. Their architectures are shown in Table 1 and Table 2 respectively.

Evaluation criteria To demonstrate that WSNet is capable of learning more compact and efficient models than conventional CNNs, three evaluation criteria are used in our experiments: model size, the number of multiply and adds in calculation (mult-adds) and classification accuracy. For the results of WSNet models, we also give the std of five different runs.

Implementation details WSNet is implemented and trained from scratch in Tensorflow (Abadi et al., 2016).

Table 1: Baseline-1: configurations of the baseline network used on MusicDet200K. Each convolutional layer is followed by a nonlinearity layer (*i.e.* ReLU), batch normalization layer and pooling layer, which are omitted in the table for brevity. The strides of all pooling layers are 2. The padding strategies adopted for both convolutional layers and fully connected layers are all "size preserving".

Layer	conv1	conv2	conv3	conv4	conv5	conv6	conv7	fc1	fc2
Filter sizes	32	32	16	8	8	8	4	1536	256
#Filters	32	64	128	128	256	512	512	256	128
Stride	2	2	2	2	2	2	2	1	1
#Params	1K	65K	130K	130K	260K	1M	1M	390K	33K
#Mult-Adds (10^8)	4.1	65.5	32.7	8.2	4.1	4.2	1.0	0.1	0.007

Table 2: Baseline-2: configuration of the baseline network used on ESC-50, UrbanSound8K and DCASE. This baseline is adapted from SoundNet (Aytar et al., 2016) by applying pooling layers to all but the last convolutional layer. For brevity, the nonlinearity layer (*i.e.* ReLU), batch normalization layer and pooling layer following each convolutional layer are omitted. The kernel sizes for pooling layers following conv1-conv4 and conv5-conv7 are 8 and 4 respectively. The stride of every pooling layers is 2.

Layer	conv1	conv2	conv3	conv4	conv5	conv6	conv7	conv8
Filter sizes	64	32	16	8	4	4	4	8
#Filters	16	32	64	128	256	512	1024	1401
Stride	2	2	2	2	2	2	2	2
#Params	1K	16K	32K	65K	130K	520K	2M	11M
#Mult-Adds (10^8)	2.3	9.0	4.5	2.3	1.2	1.2	1.2	2.3

Table 3: Ablative study of the effects of different settings of WSNet on the model size, computation cost (in terms of #mult-adds) and classification accuracy on ESC-50. For clear description, we name WSNets with different settings by the combination of symbols S/C/D/Q. "S" denotes the weight sampling along spatial dimension; "C" denotes the weight sampling along the channel dimension. "D" denotes denser filter sampling. "Q" denotes weight quantization. The numbers in subscripts of S/C/D/Q denotes the maximum compactness (ref. to Sec. 3.1 for the definition of compactness) on spatial/channel dimension in all layers, the ratio of the number of filters in WSNet versus in the baseline and the ratio of WSNet's size before and after weight quantization, respectively. The model size and the computational cost are provided for the baseline. For the model size and #mult-adds of WSNet, we provide the ratio of the baseline's model size versus WSNet's model size and the ratio of the baseline's #Mult-Adds versus WSNet's #Mult-Adds.

WSNet's settings	conv{1-4}			conv5			conv6			conv7			conv8			Acc.	Model size	Mult-Adds
	S	C	D	S	C	D	S	C	D	S	C	D	S	C	D			
Baseline	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	66.0 ± 0.2	13M (1×)	2.4e8 (1×)
BaselineQ_4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	65.7 ± 0.2	4×	1×
S_2	2	1	1	2	1	1	2	1	1	2	1	1	2	1	1	66.6 ± 0.3	2×	1×
S_4	4	1	1	4	1	1	4	1	1	4	1	1	4	1	1	66.3 ± 0.1	4×	1.6×
S_8	8	1	1	4	1	1	4	1	1	4	1	1	8	1	1	65.2 ± 0.1	7×	4.7×
C_2	1	2	1	1	2	1	1	2	1	1	2	1	1	2	1	66.8 ± 0.2	2×	1×
C_4	1	4	1	1	4	1	1	4	1	1	4	1	1	4	1	66.5 ± 0.3	4×	1.6×
C_8	1	8	1	1	4	1	1	4	1	1	4	1	1	8	1	65.8 ± 0.3	8×	2.8×
S_4C_4	4	4	1	4	4	1	4	4	1	4	4	1	4	4	1	65.6 ± 0.3	16×	6.3×
S_8C_8	4	4	1	4	8	1	4	8	1	4	8	1	8	8	1	65.2 ± 0.3	60×	**18.1**×
$S_8C_4D_2$	4	4	2	4	4	1	4	4	1	4	4	1	8	4	1	$\mathbf{66.5 \pm 0.1}$	25×	2.3×
$S_8C_4D_2Q_4$	4	4	2	4	4	1	4	4	1	4	4	1	8	4	1	66.2 ± 0.1	100×	2.3×
$S_8C_8D_2$	4	4	1	4	8	1	4	8	1	4	8	1	8	8	1	66.1 ± 0.0	45×	**2.4**×
$S_8C_8D_2Q_4$	4	4	2	4	8	1	4	8	1	4	8	1	8	8	1	65.8 ± 0.0	**180**×	2.4×

Following (Aytar et al., 2016), the Adam (Kingma & Ba, 2014) optimizer, a fixed learning rate of 0.001, and momentum term of 0.9 and batch size of 64 are used throughout experiments. We initialized all the weights to zero mean gaussian noise with a standard deviation of 0.01. In the network used on MusicDet200K, the dropout ratio for the dropout layers (Srivastava et al., 2014) after each fully connected layer is set to be 0.8. The overall training takes 100,000 iterations. The codes are available at https://github.com/AIROBOTAI/wsnet-v1.

4.2. Results and analysis

4.2.1. ESC-50

Ablation analysis We investigate the effects of each component in WSNet on the model size, computational cost and classification accuracy. The comparative study results of different settings of WSNet are listed in Table 3. For clear description, we name WSNets with different settings by the combination of symbols S/C/D/Q. Please refer to the caption of Table 3 for detailed meanings.

(1) Spatial sampling. We test the performance of WSNet by using different sampling stride S in spatial sampling. As listed in Table 3, S_2 and S_4 slightly outperforms the classification accuracy of the baseline, possibly due to reducing the overfitting of models. When the sampling stride is 8, *i.e.* the compactness in spatial dimension is 8 (ref. to Section 3.2.1), the classification accuracy of S_8 only drops by 0.6%. Note that the maximum compactness along the spatial dimension is equal to the filter size, thus for the layer "conv{5-7}" which have filter sizes of 4, their compactnesses are limited by 4 (highlighted by underlines in Table 3). Above results clearly demonstrate that the spatial sampling enables WSNet to learn significantly smaller model with comparable accuracies w.r.t. the baseline.

(2) Channel sampling. Three different compactness along the channel dimension, *i.e.* 2, 4 and 8 are tested by comparing with baslines. It can be observed from Table 3 that C_2 and C_4 and C_8 have linearly reduced model size without incurring noticeable drop of accuracy. In fact, C_2 and C_4 can even improve the accuracy upon baselines, demonstrating the effectiveness of channel sampling in WSNet. When learning more compact models, C_8 demonstrates better performance compared to S_8 that has the same compactness in the spatial dimension, which suggests we should focus on the channel sampling when the compactness along the spatial dimension is high.

We then simultaneously perform weight sampling on both the spatial and channel dimensions. As demonstrated by the results of S_4C_4 and S_8C_8, WSNet can learn highly compact models without significant performance drop (less than 1%).

(3) Denser weight sampling. Denser weight sampling is used to enhance the learning capability of WSNet with aggressive compactness (*i.e.* when S and C are large) and make up the performance loss caused by sharing too much parameters among filters. As shown in Table 3, by sampling $2\times$ more filters in conv{1-4}, $S_8C_8D_2$ significantly outperforms the S_8C_8. Above results demonstrate the effectiveness of denser weight sampling to boost the performance.

(4) Integral image for efficient computation. As evidenced in the last column in Table 3, the proposed integral image method consistently reduces the computation cost of WSNet. For S_8C_8 which is $60\times$ smaller than the baseline, the computation cost (in terms of #mult-adds) is significantly reduced by 18.1 times. Due to the extra computation cost brought by the 1×1 convolution in denser filter sampling, $S_8C_8D_2$ achieves lower acceleration ($2.4\times$). Group convolution (Xie et al., 2017) can be used to alleviate the computation cost of the added 1×1 convolution layers. We will explore this direction in our future work.

(5) Weight quantization. It can be observed from Table 3 that by using 256 bins to represent each weight by one byte (*i.e.* 8bits), $S_8C_8D_2Q_4$ and $S_8C_4D_2Q_4$ have much smaller model size compared with baselines while incurring negligible accuracy loss. The above result demonstrates that the weight quantization is complementary to WSNet and they can be used jointly to effectively reduce the model size of WSNet. Please ref. to supplementary material for the details of the weight quantization methods.

(6) WSNet versus narrowed baselines. To further verify WSNet's capacity of learning compact models, we compare WSNet with baselines compressed in an intuitive way, *i.e.* reducing the number of filters in each layer. If #filters in each layer is reduced by T, the overall #parameters in baselines is reduced by T^2 (*i.e.* the compression ratio of model size is T^2). Due to space limitation, we defer the experimental comparisons and analysis to supplementary material.

4.2.2. Comparison with State-of-the-art

The comparison of WSNet with other state-of-the-arts on ESC-50 is listed in Table 4. Compared with the SoundNet trained with provided data, WSNets significantly outperform its classification accuracy by over 10% with more than $100\times$ smaller models. After pre-training using a large number of unlabeled videos, SoundNet* achieves better accuracy than WSNet. However, since the unsupervised pre-training method is orthogonal to WSNet, we believe that WSNet can achieve better performance by training in a similar way as SoundNet (Aytar et al., 2016) on a large amount of unlabeled video data. Due to space limit, for experimental results on other datasets as well as the ablative study on MusicDet200K, please refer to supplementary material.

Table 4: Comparison with state-of-the-arts using 1D CNNs on ESC-50. All results of WSNet are obtained by 10-folder validation. Please refer to Table 3 for the meaning of symbols S/C/D/Q. SoundNet* use extra training data while other methods use only provided training data.

Model	Acc. (%)	Model size
Piczak ConvNet (Piczak, 2015b)	64.5	28M
SoundNet (Aytar et al., 2016)	51.1	13M
SoundNet* (Aytar et al., 2016)	72.9	13M
WSNet ($S_8C_4D_2$)	66.5 ± 0.10	0.52M
WSNet ($S_8C_4D_2Q_4$)	$\mathbf{66.25 \pm 0.25}$	**0.13M**
WSNet ($S_8C_8D_2$)	66.1 ± 0.15	0.29M
WSNet ($S_8C_8D_2Q_4$)	$\mathbf{65.8 \pm 0.25}$	**0.07M**

Table 5: Test error rates (in %) of WSNet and HashNet on CIFAR10 and MNIST. The baselines used for MNIST/CIFAR10 are simple 3-layer fully connected network and 5-layer convolutional network respectively. The model size is provided for the baseline. For the model size of WSNet/HashNet, we provide the ratio (*i.e.* n) of the baseline's model size versus the model size of WSNet/HashNet. For WSNet, we set the layer-wise compactness to be n. Specifically, for each convolutional layer in WSNet, we set its compactness along spatial/channel dimension to be $\sqrt{n}/\sqrt{n}$, respectively.

Model	Model size	Error rate	Model size	Error rate
	CIFAR10		MNIST	
baseline	1.2M (×)	14.91	800K (1×)	1.37
HashNet	16×	21.42	8×	1.43
HashNet	64×	30.79	64×	2.41
WSNet	16×	**17.82**	8×	**1.29**
WSNet	64×	**23.59**	64×	**1.97**

4.3. Discussions

We argue that there are two reasons for the success of WSNet: **(1)** The epitome methods (Benoît et al.; Aharon & Elad, 2008; Jojic et al.) have been successfully deployed in sparse coding literatures, where the coding dictionaries are formed by overlapping patches in the epitome which has few free parameters. This indicates effective representations of complex signals can be generated from a low-dimensional space (with high parameter efficiency). It thus motivates us to learn compact (or epitomic) filters in deep neural networks, *i.e.* all filters which participate in the actual convolution are generated from the condensed filters. **(2)** Weight quantization techniques were successfully applied for compressing deep models where multiple weights are encoded into the same value. WSNet goes further to overcome limitations of existing quantization methods through capturing the common correlations among learned filters. For example, filters of the first layer in SoundNet (Aytar et al., 2016) (as illustrated in Figure 5 in (Aytar et al., 2016)) all learn similar constituent patterns, *e.g.* the descending/ascending slope lines. The proposed weight sampling method enables WSNet to learn shared patterns by explicitly sampling filters from the condensed filter with overlapping. At the same time, the non-overlapped parts of sampled filters are able to learn different features which endows WSNet with strong learning capabilities. This is the **main reason** that why WSNet can learn much smaller networks without noticeable performance drop compared to baselines. Moreover, as the sampled filters are overlapped, we could use an integral image based method to speed up WSNets (ref. to Section 3.4). In this way, WSNet is able to learn both smaller and faster networks effectively.

4.4. Experimental results of WSNet on 2D CNNs

Since both WSNet and HashNet (Chen et al., 2015; 2016) explore weights tying, we compare them on MNIST and CIFAR10. For fair comparison, we use the same baselines used in (Chen et al., 2015; 2016). All hyperparameters during training follow (Chen et al., 2015; 2016). For each dataset, we hold out 20% of training samples to form a validation set. The comparison results between WSNet and HashNet on MNIST/CIFAR10 are listed in Table 5, from which one can observe that when learning networks with the same sizes, WSNet achieves significantly lower error rates than HashNet on both datasets. Above results clearly demonstrate the advantages of WSNet in learning compact models.

Furthermore, we also conduct experiment on CIFAR10 with the state-of-the-art ResNet50 (He et al., 2016) as baseline. ResNet50 achieves top-1 accuracy of 93.03% with #params of 0.85M. For WSNet, we set $S_w = S_h = 2$ and $C = 4$. The experimental settings follow those in (He et al., 2016). WSNet is able to achieve 9× smaller model size with slight performance drop (**0.5%**). Such promising results further demonstrate the effectiveness of WSNet.

5. Conclusion

In this paper, we present a class of **W**eight **S**ampling networks (WSNet) which are highly compact and efficient. A novel weight sampling method is proposed to sample filters from condensed filters which are much smaller than the independently trained filters in conventional networks. The weight sampling in conducted in two dimensions of the condensed filters, *i.e.* by spatial sampling and channel sampling. Taking advantage of the overlapping property of the filters in WSNet, we propose an integral image method for efficient computation. Extensive experiments on four audio classification datasets including MusicDet200K, ESC-50, UrbanSound8K and DCASE clearly demonstrate that WSNet can learn compact and efficient networks with competitive performance.

Acknowledgements Jiashi Feng was partially supported by NUS startup R-263-000-C08-133, MOE Tier-I R-263-000-C21-112, NUS IDS R-263-000-C67-646, ECRA R-263-000-C87-133 and MOE Tier-II R-263-000-D17-112.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Aharon, M. and Elad, M. Sparse and redundant modeling of image content using an image-signature-dictionary. *SIAM Journal on Imaging Sciences*, 1(3):228–247, 2008.

Anwar, S., Hwang, K., and Sung, W. Structured pruning of deep convolutional neural networks. *J. Emerg. Technol. Comput. Syst.*, 13(3):32:1–32:18, February 2017. ISSN 1550-4832. doi: 10.1145/3005348. URL http://doi.acm.org/10.1145/3005348.

Aytar, Y., Vondrick, C., and Torralba, A. Soundnet: Learning sound representations from unlabeled video. In *NIPS*, 2016.

Ba, J. and Caruana, R. Do deep nets really need to be deep? In *NIPS*, 2014.

Bagherinezhad, H., Rastegari, M., and Farhadi, A. Lcnn: Lookup-based convolutional neural network. *arXiv preprint arXiv:1611.06473*, 2016.

Benoît, L., Mairal, J., Bach, F., and Ponce, J. Sparse image representation with epitomes. In *CVPR*.

Bucilu, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In *KDD*, 2006.

Chen, W., Wilson, J., Tyree, S., Weinberger, K., and Chen, Y. Compressing neural networks with the hashing trick. In *ICML*, 2015.

Chen, W., Wilson, J. T., Tyree, S., Weinberger, K. Q., and Chen, Y. Compressing convolutional neural networks in the frequency domain. In *KDD*, 2016.

Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., and Feng, J. Dual path networks. *arXiv preprint arXiv:1707.01629*, 2017.

Chollet, F. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

Collins, M. D. and Kohli, P. Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*, 2014.

Denil, M., Shakibi, B., Dinh, L., de Freitas, N., et al. Predicting parameters in deep learning. In *NIPS*, 2013.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., and Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014.

Gong, Y., Liu, L., Yang, M., and Bourdev, L. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. In *NIPS*, 2015.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Jaderberg, M., Vedaldi, A., and Zisserman, A. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

Jin, J., Dundar, A., and Culurciello, E. Flattened convolutional neural networks for feedforward acceleration. *arXiv preprint arXiv:1412.5474*, 2014.

Jin, X., Yuan, X., Feng, J., and Yan, S. Training skinny deep neural networks with iterative hard thresholding methods. *arXiv preprint arXiv:1607.05423*, 2016.

Jojic, N., Frey, B. J., and Kannan, A. Epitomic analysis of appearance and shape.

Kim, Y.-D., Park, E., Yoo, S., Choi, T., Yang, L., and Shin, D. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2014.

Lebedev, V. and Lempitsky, V. Fast convnets using group-wise brain damage. In *CVPR*, 2016.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., and Lempitsky, V. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. In *ICLR*, 2017.

Luo, J.-H., Wu, J., and Lin, W. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017.

Mathieu, M., Henaff, M., and LeCun, Y. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2013.

Piczak, K. J. Esc: Dataset for environmental sound classification. In *ACM MM*, 2015a.

Piczak, K. J. Environmental sound classification with convolutional neural networks. In *MLSP*, 2015b.

Salamon, J., Jacoby, C., and Juan Pable, B. A dataset and taxonomy for urban sound research. In *ACM MM*, 2014.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Sindhwani, V., Sainath, T., and Kumar, S. Structured transforms for small-footprint deep learning. In *NIPS*, 2015.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

Stowell, D., Giannoulis, D., Benetos, E., Lagrange, M., and Plumbley, M. D. Detection and classification of acoustic scenes and events. *IEEE Transactions on Multimedia*, 17(10):1733–1746, 2015.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *CVPR*, 2015.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.