Frequency Domain Compact 3D Convolutional Neural Networks

Hanting Chen[1,2], Yunhe Wang[2], Han Shu[2], Yehui Tang[1,2], Chunjing Xu[2]*,
Boxin Shi[3,4], Chao Xu[1], Qi Tian[2], Chang Xu[5]

[1] Key Lab of Machine Perception (MOE), Dept. of Machine Intelligence, Peking University.
[2] Noah's Ark Lab, Huawei Technologies. [3] NELVT, Dept. of CS, Peking University. [4] Peng Cheng Laboratory.
[5] School of Computer Science, Faculty of Engineering, The University of Sydney.

{htchen, yhtang, shiboxin}@pku.edu.cn, xuchao@cis.pku.edu.cn, c.xu@sydney.edu.au
{yunhe.wang, han.shu, xuchunjing, tian.qi1}@huawei.com

Abstract

*This paper studies the compression and acceleration of 3-dimensional convolutional neural networks (3D CNNs). To reduce the memory cost and computational complexity of deep neural networks, a number of algorithms have been explored by discovering redundant parameters in pre-trained networks. However, most of existing methods are designed for processing neural networks consisting of 2-dimensional convolution filters (*i.e. *image classification and detection) and cannot be straightforwardly applied for 3-dimensional filters (*i.e. *time series data). In this paper, we develop a novel approach for eliminating redundancy in the time dimensionality of 3D convolution filters by converting them into the frequency domain through a series of learned optimal transforms with extremely fewer parameters. Moreover, these transforms are forced to be orthogonal, and the calculation of feature maps can be accomplished in the frequency domain to achieve considerable speed-up rates. Experimental results on benchmark 3D CNN models and datasets demonstrate that the proposed Frequency Domain Compact 3D CNNs (FDC3D) can achieve the state-of-the-art performance, *e.g.* a $2\times$ speed-up ratio on the 3D-ResNet-18 without obviously affecting its accuracy.*

1. Introduction

Deep neural networks, especially convolutional neural networks (CNNs) have been well demonstrated in a large variety of computer vision tasks. Plenty of manually designed convolutional neural networks, such as AlexNet [17], VGGNet [29], and ResNet [13] were proposed to achieve impressive classification accuracy on the challenging ILSVRC 2012 dataset [28]. Similar successes have been repeated in other tasks, including object detection (*e.g.* Faster RCNN [26] and SSD [20]) and segmentation (*e.g.* FCN [21] and Mask r-cnn [12]). In these aforementioned computer vision tasks, each individual image is often processed separately and consumes hundreds of *MB* memory. 3D convolution has been developed to tackle video-based processing task, such as human action recognition [16]. Compared to 2D convolution, more resources would be cost if video frames are investigated at the same for the sake of temporal information. For example, 3D-ResNet-50 [11] requires 354*MB* memory and over 22*G* times of floating number multiplications, which is much higher than the conventional 2D-ResNet-50 with 103*MB* memory and 4*G* FLOPs.

Over the years, considerable methods were proposed for

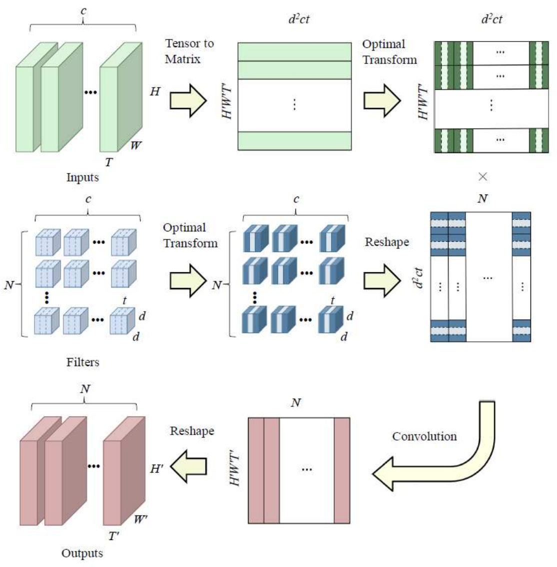


Figure 1. An illustration of the proposed FDC3D. The matrix of feature maps and filters are converted into the frequency domain utilizing the learned optimal transform. Then, the computation cost can be directly reduced by structural pruning.

*Corresponding author.

compressing and speeding up deep neural networks. For instance, Luo *et al*. [22] pruned filters based on statistics information from the next layer. Chen *et al*. [2] used a hash function and represented weights in the same hash bucket with a single parameter. Vanhouche *et al*. [32] explored deep neural networks with 8-bit integer values to replace original models with 32-bit floating values to achieve the compression and speed-up directly. Courbariaux and Bengio [4] explored neural networks with binary weights and activations. Restgari *et al*. [25] further incorporated binary convolutions into the modern neural architecture to achieve higher performance. Wang *et al*. [34] compressed filters in frequency domain with the help of discrete cosine transform.

Although the above mentioned approaches have made tremendous efforts for learning portable deep networks, most of them are designed for 2D CNNs, and may not be optimal to process 3D convolutional neural networks. Compared with traditional convolutional networks, 3D CNNs are developed for processing videos (*e.g*. action recognition [16]) or a series of input images (*e.g*. medical images of a patient [3]), and filters in 3D CNNs have an additional dimension. In fact, either videos or medical images can be regarded as an image sequence where there is high relevance between any two adjacent images. Convolution filters in this additional dimension should also have high temporal similarity to extract useful information from the input data, as shown in Figure 3.

In this paper, we convert filters into the frequency domain to investigate their redundancy and produce compact 3D convolutional neural networks. A series of transforms are learned to convert filters in each convolutional layer from the spatial domain into the frequency domain. Coefficients of these filters will be structurally sparse in the frequency domain, which can be significantly compressed by discarding their subtle components. In addition, these transforms are forced to be orthogonal during the training procedure so that we can relax the convolution operations in the spatial domain to the same operations on frequency coefficients of input data and filters with extremely lower computational complexities. Moreover, redundant filters of small importances will also be abandoned for portable neural networks. The illustration of the proposed method is shown in Figure 1. Experiments on benchmark models including 3D-ResNet-18 and 3D U-Net demonstrate that the proposed scheme outperforms state-of-the-art methods for learning compact 3D convolutional neural networks, in terms of compression and speed-up ratios.

This paper is organized as follows. Section 2 investigates related works on network pruning algorithms and 3D convolutional networks. Section 3 proposes a three-dimensional neural network pruning method by converting filters to frequency domain. Section 5 illustrates experimental results of the proposed method on benchmark datasets and models and Section 6 concludes the paper.

2. Related Works

Here we first study the 3D CNNs in various tasks, and then review pruning methods for deep neural networks.

2.1. 3D Convolutional Neural Networks

3D convolutional neural networks are proposed to tackle multi-frame or multi-image inputs, which become more and more popular in recent applications such as videos and volumetric images processing. The seminal work [16] developed 3D convolution filters by extracting features from temporal dimension, thereby capturing the multi-frame motion information. Tran *et al*. [31] further proposed C3D for generic spatio-temporal feature learning in large-scale dataset and outperforms 2D convolutional neural network, which demonstrate the 3D convolution filters are more suitable to tack the information in videos. Feichtenhofer *et al*. [6] proposed a two-stream convolutional neural network which consist of both 2D and 3D convolution filters to extract both the spatial and temporal information. Hara *et al*. [10] pushed the classical 2D-ResNet [13] to three dimension and achieved better performance than the relatively shallow C3D networks. Moreover, Hara *et al*. [11] evaluated that the current video datasets have sufficient data for the training of very deep models like ResNet-152. Besides the video classification, 3D CNNs have been widely used in medical image segmentation, since the brain MRI images are volumetric. 3D U-Net [3] have been introduced to perform end-to-end segmentation on volumetric medical images.

Although 3D CNNs have achieved satisfactory performance in video and volumetric images processing tasks, their massive computation cost prevents their deployment on mobile and edge devices. There are urgently requirements in the compression of 3D convolutional neural networks.

2.2. Network Pruning

Network Pruning aims to remove redundant weights in CNNs to accelerate and compress the original network. Denton *et al*. [5] decomposed the weights in fully-connected layers by exploiting singular value decomposition (SVD). Han *et al*. [8] introduce pruning, trained quantization and Huffman coding to largely reduce the storage of neural networks. Li *et al*. [19] proposed to remove the filters that have a small impact on the output by calculating their ℓ_1-norm. Molchanov *et al*. [24] pruned the weights based on Taylor expansion to approximate the change in cost functions. Han *et al*. [9] combined pruning, quantization and Huffman coding technique to achieve a higher compression ratio. Hu *et al*. [15] propose a data-driven approach to remove redundant filters with less impacts. Luo *et al*. [22] pruned filters based on the reconstruction error in its next layer instead of current layer and regard filter pruning as an optimization problem. He *et al*. [14] proposed a novel channel pruning method to select important channels in each layer based on LASSO regression. Gui *et al*. [7] proposed a novel Adversarially Trained Model Compression (ATMC) framework to unify existing compression methods (pruning, factorization, quantization). Wu *et al*. [35] proposed a novel spectrally relaxed k-means regularization for convolution filters to achieve through compression weight-sharing. Yang *et al*. [36] designed LegoNets where the conventional filters are replaced with efficient Lego filters.

Although aforementioned trimming method can achieve satisfactory results in CNN compression, they mainly focused on pruning the conventional 2D CNNs instead of the 3D CNNs, which take an important role on video tasks. Zhang *et al*. [37] adapted a regularization based pruning algorithm [33] to 3D CNNs. However, there are no special design for 3D convolution filters with an additional temporal dimension. In this paper, we propose a novel method to learn optimal transform for eliminating the temporal dimensional redundancy of 3D convolution filters in frequency domain.

3. Approach

In this section, we will first introduce the preliminaries of 3D CNNs and then investigate the possibility for compressing 3D filters in the frequency domain. Then, we investigate a novel method for learning a series of orthogonal bases for effectively removing redundant parameters in 3D CNNs.

3.1. 3D Convolution in the Frequency Domain

Different from the conventional networks, the number of dimensionality is four in 3D CNNs for the input data, *i.e.* $\mathcal{X} \in \mathbb{R}^{H \times W \times c \times T}$, where H, W are the height and width of the input data, respectively, c is the number of the channel number, and T denotes the additional temporal dimension. Similarly, each convolution filter in 3D CNNs will have an additional dimension, *i.e.* $\mathcal{F} \in \mathbb{R}^{d \times d \times c \times t \times N}$, where $d \times d$ is the size of filter, N is the number of filters, t and c are numbers of temporal dimension and channel, respectively. The 3D convolution operation can be formulated as

$$\mathcal{Y} = \mathcal{F} * \mathcal{X} + b, \tag{1}$$

where $*$ is the convolution operation, $\mathcal{Y} \in \mathbb{R}^{H' \times W' \times N \times T'}$ is the output feature map, H', W', T' are height, width, and the temporal dimension of $\mathcal{Y}$, respectively, and b is the bias term.

Considering that there are considerable redundant parameters and filters in most of existing 2D CNNs such as VGGNet-16 [29] and ResNet-50 [13], and current 3D CNNs are mainly modified from some 2D models, *e.g.* 3D-ResNet-18 [10] and 3D-UNet [3], we should also develop effective algorithms for recognizing redundancy in 3D CNNs. Existing pruning methods have been proved to be successful in conventional neural networks. Although these methods can be directly adapted in 3D CNNs to achieve considerable speed-up and compression ratio [37], the temporal dimension, which is the major difference between 2D and 3D CNNs, is ignored. In fact, the adjacent frames in videos are highly correlated. The 3D convolution filters, which extract the information in multi-frame inputs, also present similar property. Comparing with the height and length dimension of the filters, the temporal dimension has more redundancy, which is shown in Fig. 3. To this end, we are motivated to convert the 3D convolution filters into frequency domain, where the transformed representations is more sparse than the original signals and can be easily compressed.

There are a number of transforms for factorizing input signals and discovering their redundancy in the frequency domain, *e.g.* Discrete Fourier Transform (DFT [27]) and Discrete Cosine Transform (DCT [1]). Since coefficients in the Fourier frequency domain are imaginary values which do not suitable for compressing deep neural networks, we utilize DCT with real values for compressing 3D convolution filters. As mentioned above, the temporal dimension of 3D convolution filters have more computational cost compared with the spatial dimension. Therefore, we propose to transform these filters into the frequency domain in the temporal dimension. In specific, we first reformulate the convolution operation w.r.t. Fcn. 1 according to the forth dimension (*i.e.* temporal dimension) in $\mathcal{F}$ as

$$Y = X^\top F = \sum_{i=1}^{d^2c} x_i^\top f_i, \tag{2}$$

where $X \in \mathbb{R}^{d^2ct \times H'W'T'}$ converts $\mathcal{X}$ into a matrix according to the filter size and parameters (*e.g.* padding and stride), $F \in \mathbb{R}^{d^2ct \times N}$ and $Y \in \mathbb{R}^{H'W'T' \times N}$ are matrices of filters and output feature maps, respectively. $x_i \in \mathbb{R}^{t \times H'W'T'}$ and $f_i \in \mathbb{R}^{t \times N}$ is achieved by matrix partition: $X^\top = [x_1, \ldots, x_{d^2c}]$, $F = [f_1, \ldots, f_{d^2c}]^\top$. The bias term is dropped for simplicity.

For a given t-dimensional vector $\mathbf{f} \in \mathbb{R}^{t \times 1}$, its DCT representation $\mathbf{c}$ in the frequency domain can be formulated as

$$\mathbf{c}_m = \sum_{i=0}^{t-1} \mathbf{f}_i \cos\left[\frac{\pi}{t} m(i + \frac{1}{2})\right], \tag{3}$$

where $\mathbf{c}_m$ is the m-th coefficient of $\mathbf{f}$ in the DCT frequency domain ($m \in \{1, \cdots, t\}$). Equally, the DCT can be expressed as a matrix multiplication, *i.e.* $\mathbf{c} = S\mathbf{f}$, where S is a $t \times t$ transform matrix of DCT, which can be calculated according to the above function.

Since S is an orthogonal matrix, *i.e.* $S^\top S = I$, where I is a $t \times t$ identity matrix, we can simultaneously apply the DCT and its inverse transformation on the filter matrix and the input data in temporal dimension to convert Eqn. 2 into the frequency domain, *i.e.*

$$
\begin{aligned}
Y &= \sum_{i=1}^{d^2c} x_i^\top S^\top S f_i = X^\top \widehat{S}^\top \widehat{S} F, \\
s.t.\ \ \widehat{S} &= \begin{bmatrix} S & 0 & \cdots & 0 \\ 0 & S & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & S \end{bmatrix},
\end{aligned} \tag{4}
$$

where $\widehat{S}$ is a $d^2ct \times d^2ct$ block diagonal matrix.

By transforming the filters into frequency domain, we can easily compress the network by utilizing the sparsity of converted filters. As the non-structured weight pruning [8] cannot directly accelerate deep neural networks without special implement on the matrices multiplications (*e.g.* sparse convolution), we attempt to address the 3D CNN compression task from the structured pruning perspective [22, 14], which directly reduces the number of columns or rows in matrices of convolution filters. Therefore, we discard the redundant temporal dimension of 3D convolution filters (*i.e.* some rows in $\widehat{S}F$) with subtle ℓ_2 norms and reduce Fcn. 4 to

$$
\begin{aligned}
Y &= \sum_{i=1}^{d^2c} x_i^\top (M \odot S)^\top (M \odot S) f_i \\
&= X^\top (\widehat{M} \odot \widehat{S})^\top (\widehat{M} \odot \widehat{S}) F \\
s.t.\ \ \widehat{M} &= \begin{bmatrix} M & 0 & \cdots & 0 \\ 0 & M & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & M \end{bmatrix},
\end{aligned} \tag{5}
$$

where $M \in t \times t$ is a mask matrix to discard rows with smaller values in frequency domain and $\odot$ is Hadamard product. In specific, the importance value V_i for i-th temporal dimension is formulated as $\|(\sum_{j=1}^{d^2c} Sf_j)_{i,*}\|_2$, *i.e.* the ℓ_2-norm for the converted filters of i-th temporal dimension, where the $(\cdot)_{i,*}$ denotes the i-th row of this matrix. The mask matrix M is then conducted as $M = [M_1, ..., M_t]^\top$, where $M_i = \mathbf{0}$ if V_i is the k smallest in all V_i $(i \in \{1, ...t\})$ and otherwise $M_i = \mathbf{1}$ ($\mathbf{0}, \mathbf{1}$ denotes t-dimensional vector with all values as 0,1). k is determined by the pruning rate, which will be discussed after. Thus, the complexity will be reduced naturally.

Note that the computational complexity for converting the input data into the frequency domain through S in Fcn. 4 is $\mathcal{O}(H'W'T'd^2ct^2)$. If we also apply DCT to other dims (channel dimension and kernel size dimension) in $\mathcal{X}$, the computational complexity will be significantly increased to $\mathcal{O}(H'W'T'd^4c^2t^2)$. Therefore, we only eliminate the redundancy in the temporal dimension using DCT.

3.2. Learning to Transform 3D Filters

A frequency domain based 3D convolutional networks compression method has been proposed in Fcn. 5 by converting all filters using DCT. However, DCT is designed for natural images or videos based on some priories, which is not perfectly suitable for seeking the group sparsity of 3D convolution filters. Therefore, we propose to learn optimal transform to eliminate the redundancy in the temporal dimension of 3D convolution filters. Besides utilizing fixed dictionaries or spatial-frequency domain transforms, we make the transform for converting filter matrix in an arbitrary 3D convolutional layer learnable, *i.e.*

$$
\min_S ||Y - X^\top F||_F^2 + \lambda ||\widehat{S}F||_1, \quad s.t.\ \ S^\top S = I. \tag{6}
$$

where $S \in \mathbb{R}^{t \times t}$ is the desired transformation for converting all convolution filters into the frequency domain, I is a $t \times t$ identity matrix for making S orthogonal to ensure the equality of convolution in the coefficient domain, the last term is the conventional ℓ_1-norm for eliminating subtle elements in SF and λ is a trade-off hyper-parameter to balance the two terms.

As mentioned above, we introduce a mask matrix M to structured pruning the 3D convolution filters. Therefore, the $\ell_{2,1}$-norm is more suitable for pruning the row of filters in frequency domain, that is, discard the redundant rows in $\widehat{S}F$. The object function can be reformulated as:

$$
\min_S ||Y - X^\top F||_F^2 + \lambda || \sum_{i=1}^{d^2c} Sf_i ||_{2,1}, \quad s.t.\ \ S^\top S = I. \tag{7}
$$

where $|| \cdot ||_{2,1}$ is the $\ell_{2,1}$-norm for seeking group sparsity.

Besides the redundancy in the temporal dimension, we can also discard redundant channels for the 3D convolutional networks. We further apply the idea of channel pruning [14]:

$$
\min_{\beta, F} ||Y - \sum_{i=1}^{c} \beta_i X_i^\top F_i||_F^2 + \gamma ||\beta||_1, \tag{8}
$$

where F_i and X_i denotes each channel of filters and inputs, β is introduced to find redundant input channels and γ is a penalty coefficient. The important input channel can be selected by minimizing the ℓ_1-norm of β. Then, we can discard the input channels with smaller β.

By combining the goal of Fcn. 7 and Fcn. 8, the objective function for eliminating the redundancy of temporal dimension as well as the input channels of filters simultaneously

Algorithm 1 Compressing 3D convolutional neural networks in the frequency domain.

```
Input: A pre-trained 3D convolutional neural network $\mathcal{N}$ with $p$ convolutional layers $\mathcal{L}_1, \dots, \mathcal{L}_p$, channel and temporal dimensional pruning rate $p_{c_i}$ and $p_{t_i}$ for each layer $\mathcal{L}_i$, parameters of different objects: $\lambda$ and $\gamma$.
1:  for i = 1 to $p$ do
2:      Extract convolution filters in $\mathcal{L}_i$ to form $F$ and initialize transform $S_i$ and channel sparsity parameter $\beta$;
3:      repeat
4:          Randomly select a batch of data to forward $\mathcal{N}$;
5:          Calculate the input data $X$ for $\mathcal{L}_i$ using $\mathcal{N}$;
6:          Calculate feature maps $Y$ for $\mathcal{L}_i$ using $\mathcal{N}$;
7:          Convert $F$ into frequency domain using $S_i$;
8:          Solve Fcn. 9 to simultaneously updating transform $S_i$ and channel sparsity parameter $\beta$.
9:      until convergence
10:     Discard the subtle filters according to $\beta$ and $p_{c_i}$.
11:     Eliminate the redundancy in temporal dimension according to $p_{t_i}$ and following Fcn. 5.
12:     Save the optimal transform $S_i$ for layer $\mathcal{L}_i$.
13: end for
14: Fine-tune $\widehat{\mathcal{N}}$ by keeping the discarded components;
Output: The compact 3D network $\widehat{\mathcal{N}}$.
```

can be reformulated as

$$\min_{S,F,\beta} ||Y - \sum_{i=1}^{c} \beta_i X_i^\top F_i||_F^2 + \gamma||\beta||_1 + \lambda||| \sum_{j=1}^{d^2c} S f_j ||_{2,1}, \quad s.t. \quad S^\top S = I. \tag{9}$$

Fcn. 9 can be naturally optimized used stochastic gradient descent. After we find the solution of Fcn. 9, we can eliminate the unnecessary filters to compress the neural network. Given the pruning rate of channel and temporal dimensional as p_c and p_t, the filters with p_c smallest β will be discarded and the rows of mask matrix M with $k = t \times p_t$ smallest importance value V will be set as 0. The detailed procedures of the proposed frequency domain compact 3D CNNs (FDC3D) is summarized in Algorithm 1.

4. Analysis on Compression and Speed-up

The computational complexity of the original convolution (*i.e.* Fcn. 2) is:

$$\mathcal{O}(H'W'T'd^2ctN). \tag{10}$$

After learning the optimal transform S using Fcn. 2, the computational complexity of the convolution in the frequency domain can be written as:

$$\mathcal{O}(H'W'T'd^2ct^2) + \mathcal{O}(H'W'T'd^2ctN), \tag{11}$$

which is slighly higher than Fcn. 10, since $t \ll N$ (*e.g.* $t = 3$ and $N = 64$ in the second layer of 3D-ResNet-18).

After removing the redundant parameters, the kernel size of temporal dimension t can be reduced to t', after network trimming, the input channel c can be reduced to c', so the computational complexity can be written as:

$$\mathcal{O}(H'W'T'd^2c't'^2) + \mathcal{O}(H'W'T'd^2c't'N), \tag{12}$$

so the speedup of the compression method can be written as:

$$r_c = \frac{\mathcal{O}(H'W'T'd^2ct'^2) + \mathcal{O}(H'W'T'd^2c't'N)}{\mathcal{O}(H'W'T'd^2ctN)} \approx \frac{c't'}{ct} \tag{13}$$

As for the parameters, we use parameters of one convolution layer to analyse for simplified. The number of parameters is NCd^2t before compression. After compression, the temporal dimension can be reduced to t' and the input channel can be reduced to c' with channel pruning. We also need to add a transformation matrix S which has a number of parameters of t^2. Therefore, the compression rate can be written as

$$r_s = \frac{Nc'd^2t' + t \times t}{Ncd^2t} \tag{14}$$

As the parameters of transformation matrix is relatively small compared to the parameters of convolution filters. the parameters of this matrix can be ignored. Then, the compression rate can be approximated as $\frac{c't'}{ct}$.

5. Experiments

In this section, we will demonstrate the effectiveness of the proposed 3D CNN compression method on UCF101 and Brats18 dataset. Massive experiments on ablation study and visualization are also conducted to have an explicit understanding of the proposed algorithm.

5.1. Experiments on UCF101

We first implement experiments on the UCF101 dataset [30], which is composed of 101 action classes, over 13k clips and 27 hours of video data. We compare the proposed method with Taylor Pruning (TP) [24], Filter Pruning (FP) [19] and Regularization-based Pruning (RP) [37]. We use the 3D-ResNet-18 [31] as the backbone, which is modified from 2D-ResNet-18 by converting each of the 2D convolution filters to 3D convolution filters. The increased temporal dimension parameter of kernel is same with the other two spatial dimensions.

We use stochastic gradient descent (SGD) with a initial learning rate of 0.005, a momentum of 0.9 and a weight decay of $1e-5$ to train the networks. The learning rate is divided by 10 after the validation loss saturates. The network is trained for 300 epoch. Training samples are randomly generated from videos with a 16 sample duration and randomly cropped to 112 × 112 in order to perform data augmentation following [11]. Mean subtraction is performed to subtract the mean values of ActivityNet from the sample for each color channel. p_c and p_t are set as $\frac{1}{4}$ and $\frac{1}{3}$ for 2× speed-up, and $\frac{5}{8}$ and $\frac{1}{3}$ for 4× speed-up, respectively.

Table 1. The increased error when accelerating 3D-ResNet-18 on UCF101 (baseline: 72.50%). 2× and 4× denote the speed-up ratios.

Method	Increased err.(%)	
	2×	4×
TP [24] ([37]'s impl.)	5.72	14.24
FP [19] ([37]'s impl.)	1.60	6.92
RP [37]	0.41	2.87
Ours (FDC3D)	**0.10**	**2.16**

Table 1 reports the compression results of different methods on the UCF101 dataset. The original 3D-ResNet-18 model achieves a 72.50% accuracy. Taylor Pruning (TP) and Filter Pruning (FP) have been successfully applied in 2D-convolutional neural networks. However, the compressed models of these methods suffer degradation when compared with the original models (5.72% and 1.60% for 2× acceleration), which suggests that the conventional 2D compression algorithms are not fully suitable for 3D convolution filters. RP proposed a three-dimensional regularization-based neural network pruning method, which suffers only 0.41% and 2.87% increased errors for 2× and 4× acceleration. However, RP does not consider the high-relevance between filters in temporal dimensional, which is the key difference between compression of 2D and 3D convolution filters. By introducing the optimal transform for the temporal dimensional of 3D CNNs, the proposed method could reach 2× and 4× speed-up with only 0.10% and 2.16% accuracy drop. The proposed method outperforms previous pruning approach, which demonstrate the effectiveness of the proposed method for eliminating the redundancy of temporal dimension for the 3D convolution filters.

5.2. Ablation Study

In the above sections, the effectiveness of the proposed method for learning frequency compact 3D CNNs have been verified. The proposed algorithm introduces optimal transform to convert the filters into the frequency domain instead of using DCT. Moreover, to eliminating the redundancy in channel dimensional as well as temporal dimensional, we introduce the channel pruning [14] and combine it with the proposed optimal transform in Fcn. 9. Therefore, it is necessary to study the influence of the proposed optimal transform.

We conduct the ablation experiment on the UCF-101 dataset. We use the 3D-ResNet-18 as the original model.

Table 2. Effectiveness of eliminating redundancy in the frequency domain of the proposed FDC3D.

Method	Increased err.(%)	
	2×	4×
Ours (only pruning channels)	0.81	3.58
Ours (with DCT)	0.42	2.96
Ours (with optimal transform)	0.10	2.16

The experimental details are the same as those in Section 5.1. Table 2 reports the results of eliminating redundancy in the frequency domain of the proposed FDC3D. If we do not prune the temporal dimension (*i.e.* only pruning the channels), the compressed network suffers 0.81% and 3.58% accuracy drops for 2× and 4× speed-up ratio. By applying the DCT on the filters, the compressed network can achieve higher accuracy under the same speed-up ratio, which demonstrate the effectiveness of eliminating the redundancy in temporal dimensional. However, DCT is designed for natural images and is not perfectly suitable for the filters in 3D CNNs. By learning the optimal transform for each layer, the compressed network only suffers 0.10% and 2.16% accuracy drops for 2× and 4× speed-up ratio. The results shows the superiority of learning optimal transform for compressing the temporal dimension in 3D CNNs.

5.3. Single Layer Pruning

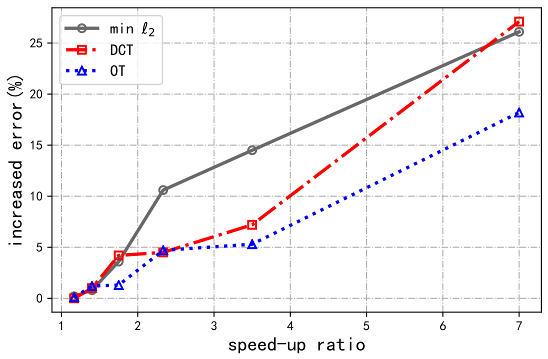


Figure 2. Single layer performance after pruning using different methods (without fine-tuning). The min ℓ_2 denotes directly removing the temporal dimensional of filters with minimal ℓ_2-norm. DCT denotes applying DCT on filters before pruning the temporal dimensional and OT denotes applying the proposed optimal transform before pruning.

In this subsection, we evaluate the performance of the proposed method in a single layer to achieve a explicit understanding of the effectiveness of the frequency domain compression. We use the 3D-ResNet-18 [11] as the original model on the UCF-101 dataset. For convenience, we only compress the temporal dimensional using the proposed method using Fcn. 7. We compare our algorithm with two naive compression strategies on the temporal dimensional: 1. directly removing the temporal dimensional of filters with smaller ℓ_2-norm; 2. using DCT to convert filters into frequency domain instead of optimal transform.

We perform the pruning in the first convolutionnal layer, whose filters are of $7 \times 7 \times 7$ size and number of temporal dimension is 7. The result of single layer pruning is shown in Figure 2. As the speed-up ratio increases, the error increases. Directly removing the temporal dimension with minimal ℓ_2-norm would largely affect the performance of 3D CNNs, which demonstrate each dimension is important for the 3D convolution filter. Therefore, we use DCT to convert the filters into frequency domain, which achieves better performance under the same speed-up ratio. By utilizing the learned optimal transform, the proposed method achieve the best performance, which demonstrate the optimal transform is suitable for seeking the group sparsity of 3D convolution filters.

5.4. Pruning for Different Depths

The effectiveness of the proposed method has been verified on 3D-ResNet-18. To further investigate the generality of the proposed scheme, we evaluate the proposed FDC3D on different depths of 3D-ResNet [11] on the UCF-101 dataset. We exploit the proposed method on ResNet-34, ResNet-50 and ResNet-101. The training settings are the same as those in Section 5.1.

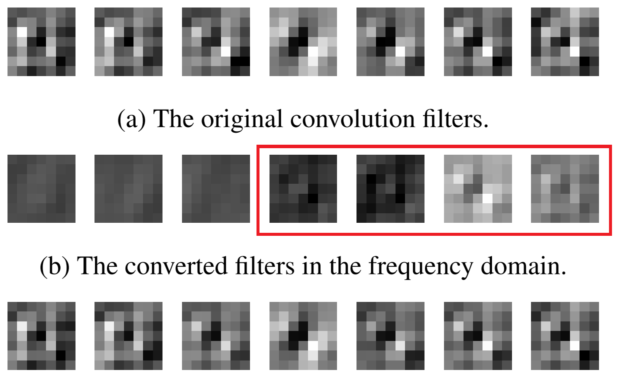
Table 3. The increased error for different architectures on UCF101 dataset. $2\times$ and $4\times$ denote the speed-up ratios.

Architecture	**Accuracy (%)**	**Increased err.(%)**	
		$2\times$	$4\times$
ResNet-34	81.1	0.13	1.94
ResNet-50	81.8	0.06	1.63
ResNet-101	83.5	0.02	1.31

Table 3 shows the compression results on different depths of 3D-ResNet. As the model becomes deeper, its accuracy increases. However, their computational cost and storage are larger. Therefore, we utilize the proposed method to accelerate these networks. The proposed algorithms can achieve $2\times$ speed-up ratio on various architectures without obvious accuracy drop. Moreover, the increased error becomes smaller for deeper model, which suggests that the larger model has more redundancy and the proposed can effectively reduce the computation cost of these heavy models.

5.5. Visualization of filters

To eliminating the redundancy of 3D convolution filters, we transform the filters into the frequency domain. By applying optimal transform to the feature maps, the convolution can be directly calculated in the frequency domain. Though we do not need to transform the compressed filters back into the spatial domain when calculating 3D convolutions, we reconstruct the convolution filters in the spatial domain for a more intuitive visualization.


(a) The original convolution filters.

(b) The converted filters in the frequency domain.

(c) The reconstructed convolution filters.

Figure 3. Visualzation of filters on the UCF101 datset. The filters in red box are reserved.

We visualize the filters of 3D-ResNet-18 [11] in the first convolution layer on the UCF101 dataset in Figure 3. The size of original filter is $7 \times 7 \times 7$ with 3 input channels and 64 output channels. For convenience, we only visualize one filter among these channels. Figure 3 (a) shows the original filters. As the adjacent input images are highly relevant, the filter to extract the volumetric images also shows similar pattern for adjacent temporal dimension, which suggests that there are lot of redundancy in 3D convolutional nerual networks. Conventional 2D CNN compression methods which pruning the weights with smaller values are thus not suitable in this situation. Therefore, we introduce optimal transform to convert filters into frequency domain. Figure 3 (b) shows the filters in the frequency domain transformed by the learned optimal transform. The converted filters can be easily divided into high-frequency filters and low-frequency filters. Therefore, we can keep the four filters with rich information in Figure 3 (b), which is boxed out with the red lines. Figure 3 (c) shows the filters reconstructed by the reserved four filters in the frequency domain. By introducing the optimal transform, we can compress the 3D convolution filters with little loss of information.

5.6. 3D-UNet on Brats 2018

Besides video recognition, another important application of 3D CNNs is medical image segmentation. The Brain Tumor Segmentation (Brats) dataset [23] provides multi-modal magnetic resonance imaging (MRI) images and expert-labeled ground truth for the segmentation of brain tumors. The task of this dataset is to produce segmentation labels of different glioma sub-regions by using the provided data in per-operative MRI scans. The sub-regions contain the enhancing tumor, the tumor core and the whole tumor. The dataset consists of 285 samples and is separated into training set with 228 samples and validation set with 57 samples, respectively. As the medical images are volumetric, conventional 2D CNNs cannot fully extract the information from the multi-image inputs. 3D U-Net [3] is therefore proposed for tackle this medical image segmentation problem. We implement experiments on the Brats 2018 dataset using the Residual 3D U-Net [18] as the backbone.

We use stochastic gradient descent (SGD) with a initial learning rate of 0.001, a momentum of 0.9 and a weight decay of $1e-5$ to train the networks. The learning rate is divided by 10 after the validation loss saturates. The network is trained for 300 epoch. Training samples are reshaped to $160 \times 192 \times 128$. The batch size is set as 1. For the proposed method, p_c and p_t are set as $\frac{1}{4}$ and $\frac{1}{3}$.

Table 4. The dice coefficient (higher is better) when accelerating 3D U-Net on the Brats 2018 dataset. The speed-up ratio is 2× for the FC3D U-Net.

Model	**Dice(ET)**	**Dice(WT)**	**Dice(TC)**
3D U-Net	0.7974	0.7971	0.6908
FC3D U-Net	0.7832	0.7831	0.6816

Table 4 reports the compression results of the proposed method on the Brats 2018 dataset. We use the dice coefficient as the evaluation index, which is widely used in the medical segmentation tasks. The dice similarity coefficient is a similarity measure to calculate the spatial overlap between two samples. The lower dice coefficient indicates the better performance. The original 3D U-Net model achieves 0.7974, 0.7971 and 0.6908 dice score for the segmentation of enhancing tumor (ET), whole tumor (WT) and tumor core (TC), respectively. Then we apply the proposed FDC3D to the 3D U-Net. As a result, the compressed network achieves 0.7832, 0.7831 and 0.6816 dice score for the segmentation of ET, WT and TC, which demonstrate the proposed method can perform well in the volumetric image segmentation tasks.

To further evaluate the performance of the compressed network with the original network, we visualize segmentation results by using 3D U-Net and the frequency domain compressed 3D U-Net in Figure 4. The enhancing tumor, tumor core and whole tumor are marked as blue, green and yellow, respectively. Figure 4 (a) shows the ground truth and Figure 4 (b) shows the segmentation results of 3D U-Net. As a result, the 3D U-Net can successfully segment different parts for the medical volumetric MRL images. Considering that the heavy computation cost of 3D U-Net, we exploit the proposed FDC3D to eliminate its redundancy. Figure 4 (c) shows the segmentation result of the compressed network. The proposed method can achieve similar result with the original network while with fewer parameters and lower computation cost.

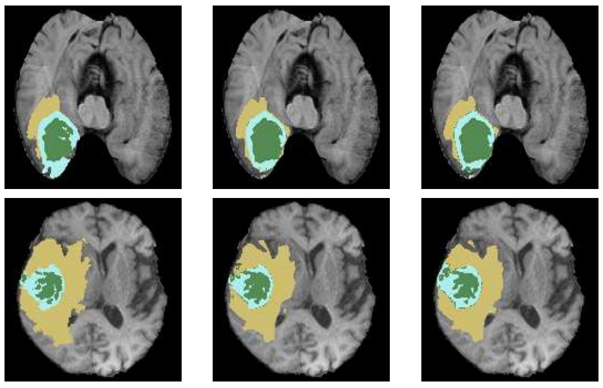


Figure 4. Example segmentation results on the Brats 2018 dataset.

6. Conclusions

As videos become ubiquitous due to the growing of multimedia on Internet, the 3-dimensional convolutional neural networks have been proposed to tackle multi-frame or multi-image datasets. However, 3D CNNs requires enormous computation resources, which prevents its usage in edge devices such as cameras and mobile phones. Here we present an effective compression method to eliminate the redundancy of 3D convolution filters in the frequency domain, namely FDC3D. The 3D convolution filters are converted into the frequency domain with structural sparsity in temporal dimensional utilizing the learned optimal transform, where the redundant parameters can be easily removed. Then the convolution can be calculated in the frequency domain by also applying optimal transform to the feature maps. The redundancy in the channel dimensional is also considered to achieve higher speed-up ratio. Detailed analysis including ablation study and visualization are conducted to demonstrate the effectiveness of the proposed algorithm. Experiments on action classification and medical image segmentation shows that the proposed FDC3D achieve higher performance than the state-of-the-art methods.

Acknowledgement

We thank anonymous reviewers for their helpful comments. This work is supported by National Natural Science Foundation of China under Grant No. 61876007, 61872012, National Key R&D Program of China (2019YFF0302902), Beijing Academy of Artificial Intelligence (BAAI), and Australian Research Council under Project DE-180101438.

References

[1] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974. 3

[2] Wenlin Chen, James T Wilson, Stephen Tyree, Kilian Q Weinberger, and Yixin Chen. Compressing convolutional neural networks. *arXiv preprint arXiv:1506.04449*, 2015. 2

[3] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016. 2, 3, 8

[4] Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 2

[5] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014. 2

[6] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, pages 1933–1941, 2016. 2

[7] Shupeng Gui, Haotao N Wang, Haichuan Yang, Chen Yu, Zhangyang Wang, and Ji Liu. Model compression with adversarial robustness: A unified optimization framework. In *NeuriPS*, pages 1283–1294, 2019. 3

[8] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 2, 4

[9] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015. 2

[10] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3154–3160, 2017. 2, 3

[11] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6546–6555, 2018. 1, 2, 6, 7

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 3

[14] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017. 3, 4, 6

[15] 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. 3

[16] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013. 1, 2

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[18] Kisuk Lee, Jonathan Zung, Peter Li, Viren Jain, and H Sebastian Seung. Superhuman accuracy on the snemi3d connectomics challenge. *arXiv preprint arXiv:1706.00120*, 2017. 8

[19] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 2, 5, 6

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 1

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[22] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017. 2, 3, 4

[23] Bjoern H Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, Yuliya Burren, Nicole Porz, Johannes Slotboom, Roland Wiest, et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE transactions on medical imaging*, 34(10):1993–2024, 2014. 7

[24] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016. 2, 5, 6

[25] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *arXiv preprint arXiv:1603.05279*, 2016. 2

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1

[27] Oren Rippel, Jasper Snoek, and Ryan P Adams. Spectral representations for convolutional neural networks. In *NIPS*, 2015. 3

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 1

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 1, 3

[30] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 5

[31] Du Tran, Lubomir D Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3d: generic features for video analysis. *CoRR, abs/1412.0767*, 2(7):8, 2014. 2, 5

[32] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS*, 2011. 2

[33] Huan Wang, Qiming Zhang, Yuehai Wang, and Roland Hu. Structured deep neural network pruning by varying regularization parameters. *ArXiv preprint: 1804.09461*, 2018. 3

[34] Yunhe Wang, Chang Xu, Shan You, Dacheng Tao, and Chao Xu. Cnnpack: Packing convolutional neural networks in the frequency domain. In *NIPS*, 2016. 2

[35] Junru Wu, Yue Wang, Zhenyu Wu, Zhangyang Wang, Ashok Veeraraghavan, and Yingyan Lin. Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. 2018. 3

[36] Zhaohui Yang, Yunhe Wang, Chuanjian Liu, Hanting Chen, Chunjing Xu, Boxin Shi, Chao Xu, and Chang Xu. Legonet: Efficient convolutional neural networks with lego filters. In *ICML*, pages 7005–7014, 2019. 3

[37] Yuxin Zhang, Huan Wang, Yang Luo, Lu Yu, Haoji Hu, Hangguan Shan, and Tony QS Quek. Three-dimensional convolutional neural network pruning with regularization-based method. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4270–4274. IEEE, 2019. 3, 5, 6