Supplementary Material: Parallel Separable 3D Convolution for Video and Volumetric Data Understanding

Felix Gonda
fgonda@g.harvard.edu
Donglai Wei
donglai@seas.harvard.edu
Toufiq Parag
paragt@seas.harvard.edu
Hanspeter Pfister
pfister@g.harvard.edu

Harvard John A. Pualson School of Engineering and Applied Sciences
Camabridge MA, USA

1 Detailed Mathematical Formulation

We here explain the mathematical details in Sec. 3.1 to justify the need of parallel streams.

Separable Convolution Kernel. A 4D tensor $\mathcal{A}$ is separable along the third dimension if $\mathcal{A}_{j_1,j_2,j_3}$, the vector of channel values at each spatial location, can be decomposed as

$$\mathcal{A}_{j_1,j_2,j_3} = \mathcal{A}^{(3)}_{j_3} \mathcal{A}^{(1,2)}_{j_1,j_2} \quad \forall (j_1,j_2,j_3), \tag{1}$$

where $\mathcal{A}^{(3)}, \mathcal{A}^{(1,2)}$ are sub-tensors with size $1\times 1\times J_3\times 1$ and $J_1\times J_2\times 1\times C$. Similarly, we can decompose $\mathcal{A}$ along other spatial dimensions. Thus, for any 4D input tensor $\mathcal{D}$,

$$\mathcal{A} * \mathcal{D} = \sum_{j_1,j_2,j_3} \mathcal{A}_{j_1,j_2,j_3} \cdot \mathcal{D}_{j_1,j_2,j_3} = \sum_{j_3} \mathcal{A}^{(3)}_{j_3} \Big(\sum_{j_1,j_2} \mathcal{A}^{(1,2)}_{j_1,j_2} \cdot \mathcal{D}_{j_1,j_2,j_3} \Big) = \mathcal{A}^{(3)} * (\mathcal{A}^{(1,2)} * \mathcal{D}), \tag{2}$$

which can be implemented with a chain of 2D and 1D convolution layer.

General Convolution Kernel. Given a general 4D tensor $\mathcal{A}$, we show how to decompose it into the sum of separable tensors. First, we use high-order singular value decomposition (HOSVD) [2] to decompose $\mathcal{A}$ with orthogonal matrices $\{U^{(k)}\}_{k\in\{1,2,3\}}$ of the size $\{I_k \times J_k\}_{k\in\{1,2,3\}}$ and singular value tensor $\mathcal{S}$ of the size $I_1\times I_2\times I_3\times C$, where each channel vector $\overrightarrow{\mathcal{A}_{i_1,i_2,i_3}}$ can be expressed as

$$\mathcal{A}_{j_1,j_2,j_3} = \sum_{i_1,i_2,i_3} \mathcal{S}_{i_1,i_2,i_3}\, U^{(1)}_{i_1,j_1}\, U^{(2)}_{i_2,j_2}\, U^{(3)}_{i_3,j_3}. \tag{3}$$

Then, we can further decompose the singular value tensor $\mathcal{S}$ into the sum of subtensors $\{\mathcal{S}^{\alpha_l,\beta_l}\}_{\alpha_l\in\{1,2,3\},\beta_l\in\{1,..,I_{\alpha_l}\}}$, whose non-zero entries are only on the β_l-th sub-tensor along

the α_l-th dimension.

$$\mathcal{S} = \sum_l \mathcal{S}^{\alpha_l,\beta_l}, \text{ where } \mathcal{S}^{\alpha_l,\beta_l} U^{(\alpha_l)}_{i_{\alpha_l},j_{\alpha_l}} = 0 \quad \text{for } i_{\alpha_l} \neq \beta_l \tag{4}$$

Suppose $\alpha_{l_0} = 1$, then $\mathcal{S}^{\alpha_{l_0},\beta_{l_0}}_{i_1,i_2,i_3} = \vec{0}$ for $i_1 \neq \beta_{l_0}$, we can define $\mathcal{A}^{l_0}_{j_1,j_2,j_3}$ as

$$\mathcal{A}^{l_0}_{j_1,j_2,j_3} = \sum_{i_1,i_2,i_3} \mathcal{S}^{\alpha_{l_0},\beta_{l_0}} U^{(1)}_{i_1,j_1} U^{(2)}_{i_2,j_2} U^{(3)}_{i_3,j_3} = U^{(1)}_{\beta_{l_0},j_1} \sum_{i_2,i_3} \mathcal{S}^{\alpha_{l_0},\beta_{l_0}} U^{(2)}_{i_2,j_2} U^{(3)}_{i_3,j_3}, \tag{5}$$

where $\mathcal{A}^{l}_{j_1,j_2,j_3}$ is a separable convolution kernel according to Eqn. 1. Thus, the given tensor $\mathcal{A}$ can be written as

$$\mathcal{A}_{j_1,j_2,j_3} = \sum_{i_1,i_2,i_3} (\sum_l \mathcal{S}^{\alpha_l,\beta_l}) U^{(1)}_{i_1,j_1} U^{(2)}_{i_2,j_2} U^{(3)}_{i_3,j_3} = \sum_l (\sum_{i_1,i_2,i_3} \mathcal{S}^{\alpha_l,\beta_l} U^{(1)}_{i_1,j_1} U^{(2)}_{i_2,j_2} U^{(3)}_{i_3,j_3}) = \sum_l \mathcal{A}^{l}_{j_1,j_2,j_3}, \tag{6}$$

where is a sum of separable convolution kernel along the α_l-th dimension. To reduce computation, we need to select a subset of $\{\mathcal{A}^l\}$ to learn for (2+1)D convolution. Previous approaches [3, 4] chooses $\mathcal{A}^l$ whose 1D and 2D decomposition correspond to temporal and spatial convolution for the video input specifically. Here, we achieve better modeling capacity for a fixed number of l by using $\{\mathcal{A}^l\}$ with sub-tensors of different orientations. Thus, we define $l_k = \{l : \alpha_l = k\}$ and group the sub-tensors by their orientations α_l as

$$\mathcal{A} * \mathcal{D} = (\sum_{l \in l_1} \mathcal{A}^{l,(1)} * (\mathcal{A}^{l,(2,3)} * \mathcal{D})) + (\sum_{l \in l_2} \mathcal{A}^{l,(2)} * (\mathcal{A}^{l,(1,3)} * \mathcal{D})) + (\sum_{l \in l_3} \mathcal{A}^{l,(3)} * (\mathcal{A}^{l,(1,2)} * \mathcal{D})), \tag{7}$$

which can be implemented as the sum of three parallel streams of 2D and 1D convolution layers with different orientations.

2 Toy Experiments

In this section, we analyze various design choices of our $\mathrm{P}_m\mathrm{SC}_n$ convolution module with toy experiments on random approximations of 3D ConvNets.

Dataset We generate 20k random data volumes of sizes $30\times20\times20\times100$ from normal distribution and randomly sample consecutive channels in the last dimension for the network input. For our target network, we generate a set of ground truth labels for input volumes as the network's output and apply data augmentation. For the approximated network, we use the output prediction of the target network as input labels.

Setup Our target network is a simplified 3D ConvNet depicted in Table 1. Our goal is to functionally approximate the target network by varying the hyper-parameters m and n, the number of streams and the dimension of the subspace respectively. We first train the target network. Then we use its output as input labels to the approximated networks. We train each network for 100 epochs using Adam optimizer with a learning rate of 0.001. We report validation loss and accuracies to assess the quality of our approximations.

Effects of m and n: We explore different values of $m \in (1,2,3)$, the number of streams to use, and $n \in (1,2,*)$, the number of channels in the intermediate layers. For n, a value of 1 indicates one-to-one convolution mapping, 2 is groups of two, and * signifies whole network approximation where the subspace spans the entire ConvNet operations except the last layer in the network. In Figure 1 (a), we show validation accuracy, and in Figure 1 (b),

#Index	Layer	Filter Size	Filter Num.	Output Size	Activation
0	Conv	$3 \times 3 \times 3$	60	$30 \times 30 \times 30$	ReLu
1	Conv	$3 \times 3 \times 3$	60	$30 \times 30 \times 30$	ReLu
2	Pooling	$2 \times 2 \times 2$	0	$15 \times 15 \times 15$	
3	Conv	$3 \times 3 \times 3$	60	$15 \times 15 \times 15$	ReLu
4	Conv	$1 \times 1 \times 1$	3	$15 \times 15 \times 15$	Sigmoid

Table 1: Target 3D ConvNet Architecture

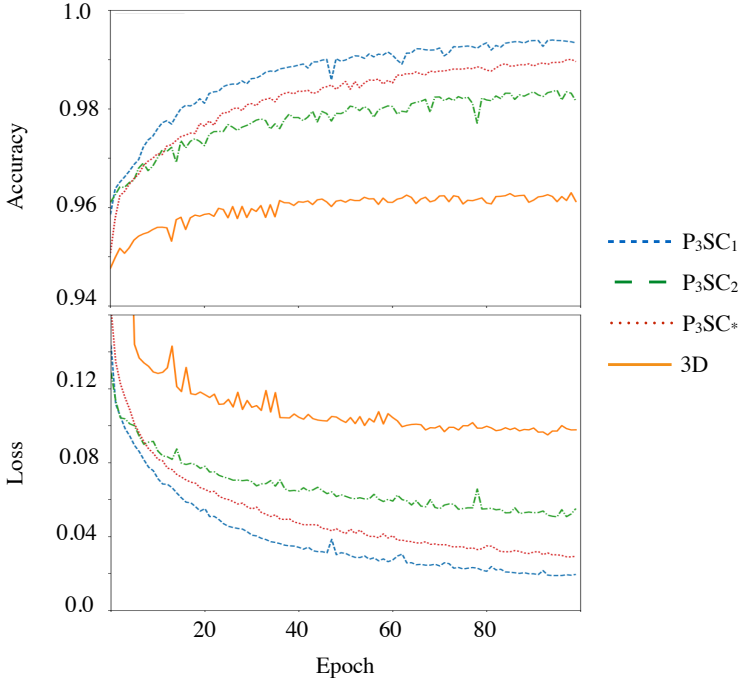

Figure 1: Comparison of validation accuracy and loss for three streams for subspaces 1, 2, and whole network against 3D target. In all cases, our P_mSC_n models achieved better error and accuracy than the 3D target.

we show validation loss for each of our P_3SC_1, P_3SC_2, and P_3SC_* modules. Irrespective of the number of streams, we observe similar performance, so we report the tri-streams results.

Varying the subspace parameter n, have more pronounce impact on performance. When approximating an entire network with a 1:1 replacement of convolutions with $n = 1$, we achieve the best approximation of the target 3D network. We think this is because the 1:1 replacement is closer in functionality to conventional 3D than configurations of $n > 1$.

3 Training Details

3.1 Action recognition

We perform training for 50 epochs with an initial learning rate of $1e^{-5}$ and decay of $1e^{-6}$. A batch size of 16 is used and computation is performed on a single GPU. For augmentation, we use temporal jittering and spatial random shifts, shears, and zooms.

3.2 MRI brain extraction

We train each network architecture from scratch using Adam optimizer with an initial learning rate of $1e^{-5}$ and decay of $1e^{-6}$. The input patch sizes is set to $64 \times 64 \times 64$ and a mini-batch size of 4 is used. Training is performed for 50 epochs on a single GPU.

3.3 EM segmentation

In order to predict the 3D affinities, we used the MALIS training loss [1]. The initial learning rate of $1e^{-4}$ is used with a momentum of 0.9 and a weight decay of $1e^{-5}$. For the first 50K iterations, we use euclidean loss function. Then MALIS loss function is swapped-in for the rest of the training iterations. After 300k iterations, we report Variation of Information (VI) scores for each network architecture. The training data was augmented with 16 three dimensional rotation, flipping and also by adding intensity noise.

4 Additional Results

4.1 MRI brain extraction

We show a qualitative comparison, in Figure **??**, of **(a)** ground truth data provided by human expert, **(b)** segmentation output of our best model P_2SC_2, and **(c)** segmentation output from 3D baseline model. We separate the segmentation by class to better observe the quality differences.

4.2 ResNet Layer Activations

In Figure 2, we show a visualization of the first eight filters of the first convolution layer of the (a) baseline 3D network, compared to the individual streams of our P_3SC_1 network depicted in (b), (c), and (d). In (b), a spatial stream captures mostly the appearance attributes of the video frames. In (c) and (d), motion dynamics are captured with orthogonal temporal streams.

References

[1] Kevin Briggman, Winfried Denk, Sebastian Seung, Moritz N. Helmstaedter, and Srinivas C. Turaga. Maximin affinity learning of image segmentation. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*. 2009.

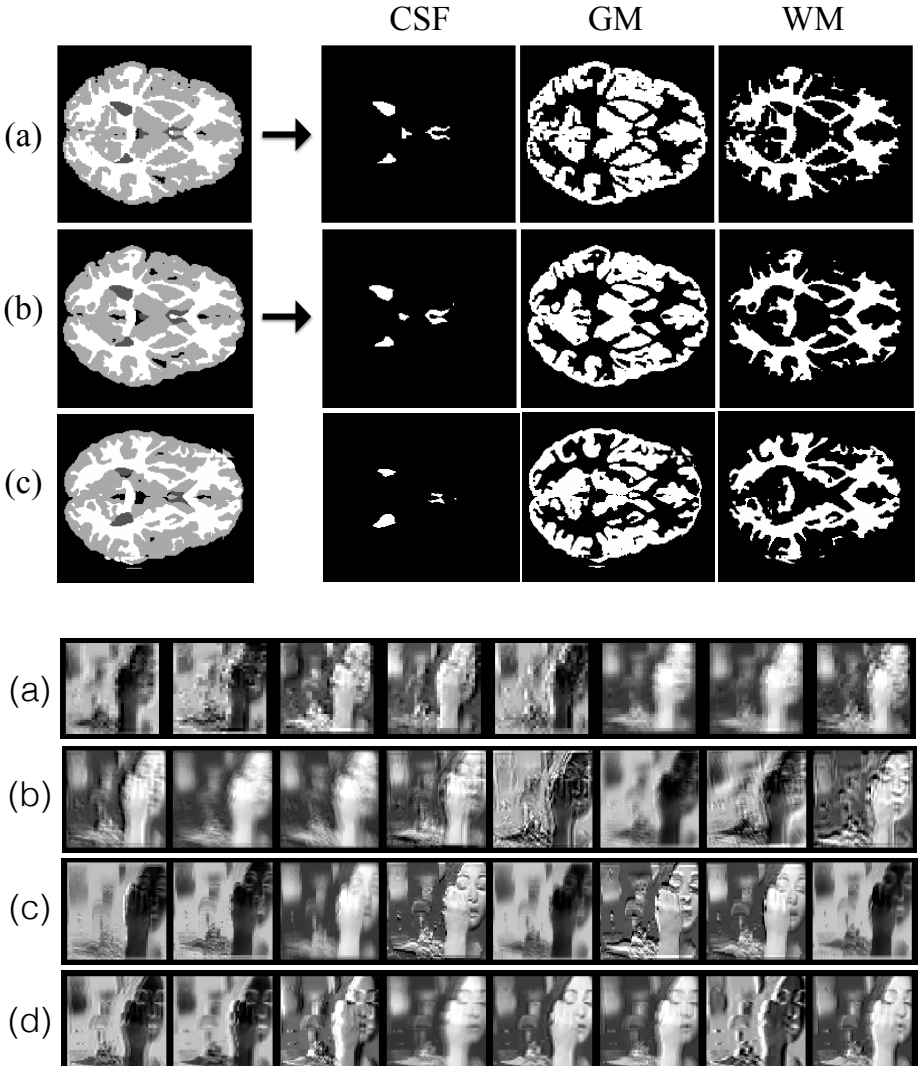

Figure 2: Comparison of layer activation of the first convolution layer for (a) 3D convolution, 1st stream (b), 2nd stream (c), and 3rd stream (d) of P_3SC_1 layers.

[2] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[3] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. 2018.

[4] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. *arXiv preprint arXiv:1712.04851*, 2017.