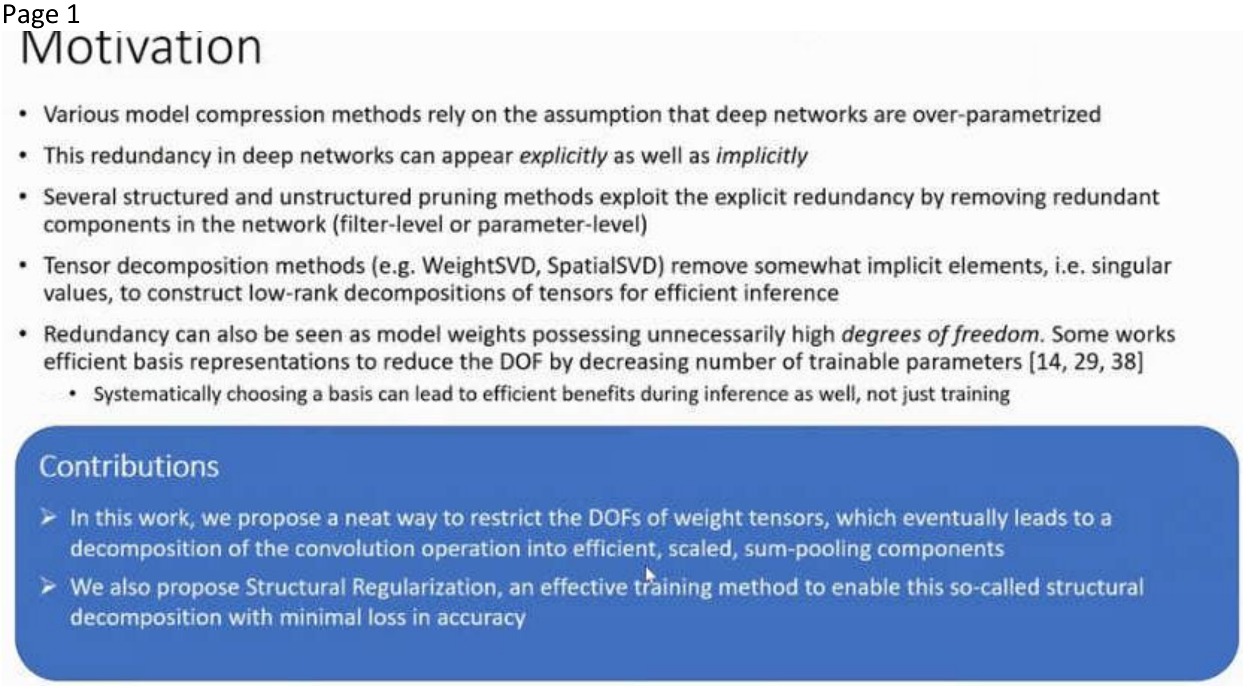
微信公众号与哔哩哔哩有视频版本同步：

1.

https://mp.weixin.qq.com/s?__biz=Mzk0OTI4NzgyNg==&tempkey=MTEzM19GVHpxWTdsbFRTWDJPQjNSZjgySXUxSDRwekVUbnhHdzFORUtQWGpVYjlPSW5ULTJZcHRGT2pMaERBUVVUX0s2VXlSTVQ0c2VXSWN5TjZCa0ZNNTctYm9lXy1kYldBTENkeWtLZ2p3UE5UcTFuUzJrYjlMZTh0czVJYUd0LVlacjdBRzV0YUtvMFVQZXRPZmxneHZMUjBJSi1zYld2eEdnU0t6ZDhBfn4%3D&chksm=435beff3742c66e58e0ac2342fa14426eb498ad1aa6c46e4d86612b60de0d9684cd50597db18#rd

Page 1



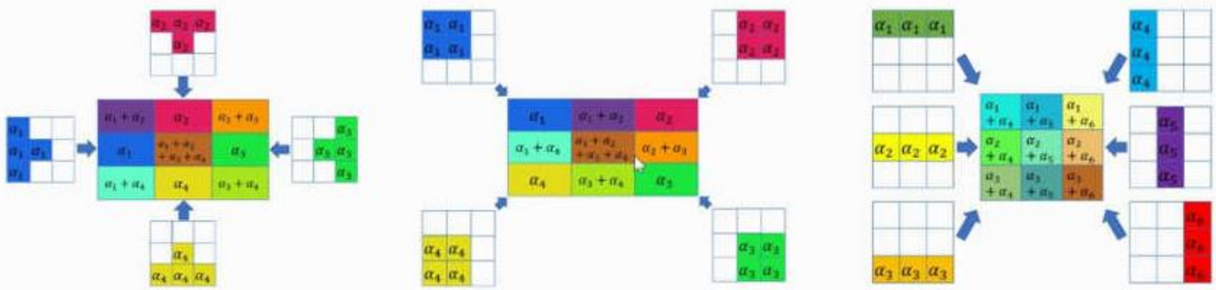
Page2

Composite Kernels

Definition 1: *For $\mathbb{R}^{C\times N\times N}$, a Composite Basis $\mathbb{B} = \{\beta_1, \beta_2, \dots, \beta_M\}$ is a linearly independent set of binary tensors of dimension $C \times N \times N$ as its basis elements. That is, $\beta_m \in \mathbb{R}^{C\times N\times N}$, each element β_{mijk} of $\beta_m \in \{0,1\}$ and $\sum_{m=1}^{M} \alpha_m \beta_m = 0$ iff $\alpha_m = 0\ \forall m$.*

Definition 2: *A kernel $W \in \mathbb{R}^{C\times N\times N}$ is a Composite Kernel if it lies in the subspace of a Composite Basis $\mathbb{B}$, i.e. it can be constructed as a linear combination of the elements of $\mathbb{B}$: $\exists \boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_M]$ such that $W \coloneqq \sum_{m=1}^{M} \alpha_m \beta_m$*

Example below shows A 3×3 Composite Kernel with different underlying structures / basis. Left two kernels possess 4 DOF, whereas the rightmost kernel possesses 6 DOF.



Page3

Convolution with a Composite Kernels

- Consider a convolution operation with a Composite Kernel W of size $C \times N \times N$. To compute an output, W is convolved with a $C \times N \times N$ volume of the input feature map

$$X * W = X * \sum_{m=1}^{M} \alpha_m \beta_m = \sum_{m=1}^{M} \alpha_m (X * \beta_m) = \sum_{m=1}^{M} \alpha_m\, sum(X \bullet \beta_m) = \sum_{m=1}^{M} \alpha_m E_m \quad (1)$$

- Ordinarily, the convolution $X * W$ would involve CN^2 multiplications and $CN^2 - 1$ additions
- But since β_m is a binary tensor, the operation $sum(X \cdot \beta_m)$ is same as adding the elements of X wherever $\beta_m = 1$, thus no multiplications!
- Hence, from (1), we see that we only need M multiplications, and the number of additions are given by:

$$\text{Num additions} = \sum_{m=1}^{M} \underbrace{(sum(\beta_m) - 1)}_{\text{from } sum(X \bullet \beta_m)} + \underbrace{(M-1)}_{\text{from } \sum \alpha_m E_m} = \sum_{m=1}^{M} sum(\beta_m) - 1 \quad (2)$$

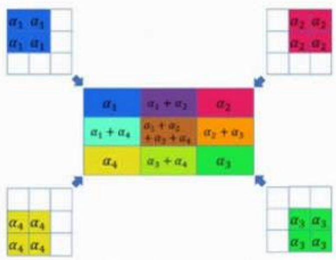
- Depending on the structure ($\mathbb{B}$), num additions *per output* can be larger than $CN^2 - 1$, e.g. in the second example on previous slide where $C = 1, N = 3, M = 4$, we have $\sum_m sum(\beta_m) - 1 = 15 > CN^2 - 1$
- Next, we show that the num additions can be amortized to as low as $M - 1$ ☺

Pag5

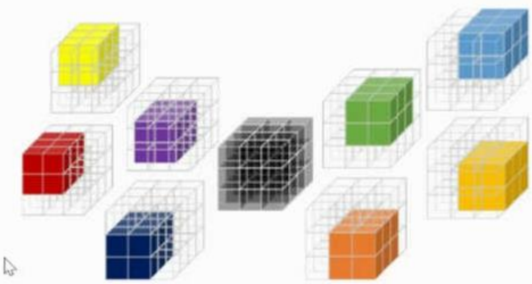
Structured Kernels

Definition 3: *A kernel in $\mathbb{R}^{C\times N\times N}$ is a Structured Kernel if it is a Composite Kernel $M = cn^2$ for some $1 \le n \le \mathrm{N}$ and $1 \le c \le C$, and if each basis tensor β_m is made of a $(C-c+1)\times(N-n+1)\times(N-n+1)$ cuboid of $1'$s, while rest of its coefficients being 0.*

- A Structured Kernel is characterized by its dimensions C, N and its underlying parameters c, n.
- Convolutions performed using Structured Kernels are called Structured Convolutions.



This is a 2D case of a 3×3 structured kernel where $C = 1$, $N = 3, c = 1, n = 2$. As shown, there are $M = cn^2 = 4$ basis elements and each element has a 2×2 sized patch of 1's

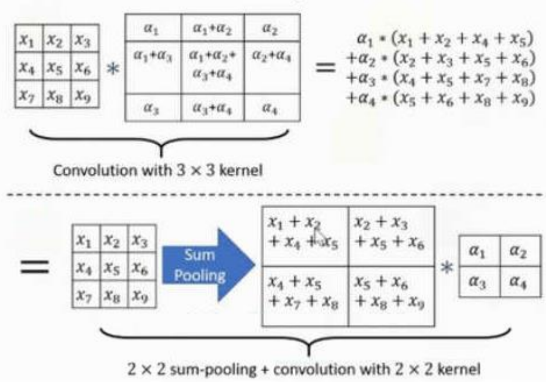


This shows a 3D case where $C = 4, N = 3, c = 2, n = 2$. Here, there are $M = cn^2 = 8$ basis elements and each element has a $3\times2\times2$ cuboid of $1's$. Note how these cuboids of $1's$ (shown in colors) cover the entire $4\times3\times3$ grid

Page6

Structured Convolutions

- A major advantage of Structured Kernels is that all the basis elements are just shifted versions of each other
- This means, considering the convolution $X * W$ (in Eq. (1)) for the *entire* feature map X, the summed outputs $X * \beta_m$ for all $\beta_m' s$ **are the same** (except on the edges of X)
- As a result, the outputs $\{X * \beta_1, \dots, X * \beta_{cn^2}\}$ can be computed using a single sum-pooling operation on X with a kernel size of $(C-c+1)\times(N-n+1)\times(N-n+1)$
- Example below shows how a convolution with a 3×3 structured kernel can be broken into a 2×2 sum-pooling followed by a 2×2 convolution with a kernel made of α_i's:



Page4

Structural Decomposition

- Consider a general convolution layer of size $C_{out} \times C \times N \times N$ that has C_{out} kernels of size $C \times N \times N$
- In the proposed design, the same underlying basis $\mathbb{B} = \{\beta_1, \ldots, \beta_{cn^2}\}$ is used for all C_{out} kernels in the layer
- Suppose any two structured kernels in this layer, $W_1 := \sum_{m=1}^{M} \alpha_m^{(1)} \beta_m$ and $W_2 := \sum_{m=1}^{M} \alpha_m^{(2)} \beta_m$. The convolution outputs will be, $X * W_1 = \sum_m \alpha_m^{(1)} (X * \beta_m)$ and $X * W_2 = \sum_m \alpha_m^{(2)} (X * \beta_m)$
- The computation $X * \beta_m$ is common to all the kernels of this layer. **Hence, the sum-pooling only needs to be computed once and reused across all layers.** This important result leads to the following decomposition:

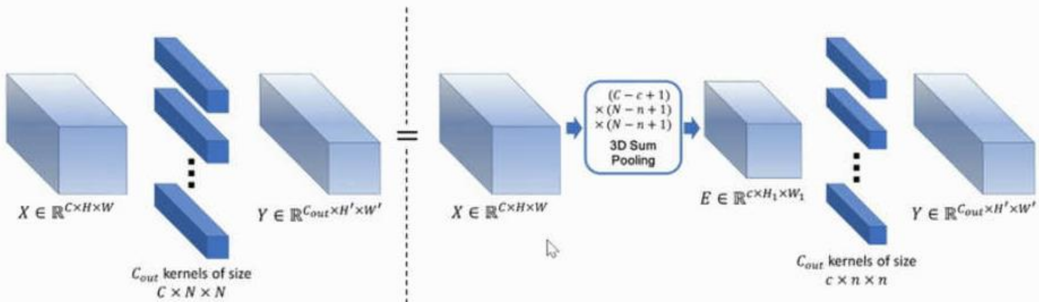


Figure: *Decomposition of Structured Convolution. On the left, the conventional operation of a structured convolutional layer of size $C_{out} \times C \times N \times N$ is shown. On the right, we show that it is equivalent to performing a 3D sum-pooling followed by a convolutional layer of size $C_{out} \times c \times n \times n$.*

Improvement in Computational Complexity

- The following table shows a comparison of the number of parameters and operations (multiplications and additions) before and after the structural decomposition

	Conventional convolution (before decomposition)	Sum-pooling + Smaller convolution (after decomposition)
# Parameters	$C_{out}CN^2$	$C_{out}cn^2$
# Multiplications	$CN^2 \times C_{out}H'W'$	$cn^2 \times C_{out}H'W'$
# Additions	$(CN^2 - 1) \times C_{out}H'W'$	$((C - D + 1)(N - k + 1)^2 - 1) \times cH_1W_1$ $+ (cn^2 - 1) \times C_{out}H'W'$

- We see that both number of parameters and multiplications have reduced by a factor of $\mathbf{cn^2}/\boldsymbol{CN^2}$
- Number of additions after decomposition can be re-written as:

$$C_{out} \left(\frac{((C - c + 1)(N - n + 1)^2 - 1)cH_1W_1}{C_{out}} + (cn^2 - 1)H'W' \right)$$

- Hence, when C_{out} is large enough, the first term inside parentheses (the sum-pooling component) gets amortized and num additions become $\approx (cn^2 - 1) \times C_{out}H'W' = (M - 1) \times C_{out}H'W'$ ... since $M = cn^2$
  - Hence, additions also decrease by approximately $\mathbf{cn^2}$**/CN^2**

Applicability of Structural Decomposition

- Standard convolution ($C \times N \times N$), depthwise convolution ($1 \times N \times N$), and pointwise convolution ($C \times 1 \times 1$) kernels can all be constructed as 3D structured kernels, which means that this decomposition can be widely applied to existing architectures
- It can also be extended to Fully Connected layers by noting that the linear operation WX is mathematically equivalent to a 1×1 convolution $unsqueezed(X) * unsqueezed(W)$, where $unsqueezed(X)$ is the same as X but with dimensions $Q \times 1 \times 1$ and $unsqueezed(W)$ is the same as W but with dimensions $P \times Q \times 1 \times 1$

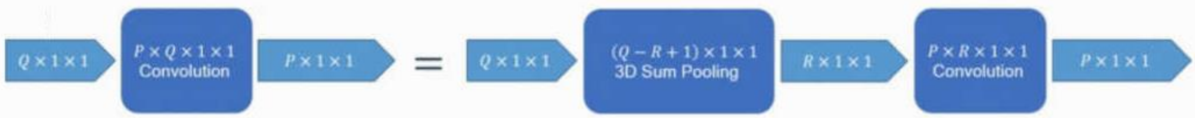


Figure: Structural decomposition of a matrix multiplication

- Same as before, we get a reduction in both number of parameters and multiplications by a factor of $\boldsymbol{R/Q}$ and number of additions by a factor of $\frac{R(Q-R)+(PR-1)P}{(PQ-1)P}$

Structural Regularization

- We propose training a deep network with a Structural Regularization loss that can gradually push the deep network's kernels to be structured via training:

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \lambda \sum_{l=1}^{L} \frac{\left\|(I - A_l A_l^{+})W_l\right\|_F}{\|W_l\|_F}$$

- **Proposed Training Scheme:**
  - Step 1: Train the original architecture with the Structural Regularization loss.
    - After Step 1, all weight tensors in the deep network will be *almost* structured.
  - Step 2: Apply the decomposition on every layer and compute $\alpha_l = A_l^{+} W_l$.
    - This results in a smaller and more efficient decomposed architecture with α_l's as the weights. Note that, every convolution / linear layer from the original architecture is now replaced with a sum-pooling layer and a smaller convolution / linear layer.

- We show in the paper that the Structural Regularization training method leads to significantly better performance than just training the Structured Convolutions as an architectural feature

Results – Image Classification with ResNets

Table: ResNets on CIFAR-10

Architecture	Adds ($\times 10^6$)	Mults ($\times 10^6$)	Params ($\times 10^6$)	Acc. (in %)
ResNet56	125.49	126.02	0.85	93.03
Struct-56-A (ours)	63.04	62.10	0.40	92.65
Struct-56-B (ours)	48.49	33.87	0.21	91.78
Ghost-Res56 [7]	63.00	63.00	0.43	92.70
ShiftRes56-6 [42]	51.00	51.00	0.58	92.69
AMC-Res56 [11]	63.00	63.00	–	91.90
ResNet32	68.86	69.16	0.46	92.49
Struct-32-A (ours)	35.59	35.09	0.22	92.07
Struct-32-B (ours)	26.29	18.18	0.11	90.24
ResNet20	40.55	40.74	0.27	91.25
Struct-20-A (ours)	20.77	20.42	0.13	91.04
Struct-20-B (ours)	15.31	10.59	0.067	88.47
ShiftRes20-6 [42]	23.00	23.00	0.19	90.59

Table: ResNets on ImageNet

Architecture	Adds ($\times 10^9$)	Mults ($\times 10^9$)	Params ($\times 10^6$)	Acc. (in %)
ResNet50	4.09	4.10	25.56	76.15
Struct-50-A (ours)	2.69	2.19	13.49	75.65
Struct-50-B (ours)	1.92	1.38	8.57	73.41
ChPrune-R50-2x [12]	2.04	2.05	17.89	75.44
WeightSVD-R50 [48]	2.04	2.05	13.40	75.12
Ghost-R50 (s=2) [7]	2.20	2.20	13.00	75.00
Versatile-v2-R50 [40]	3.00	3.00	11.00	74.50
ShiftResNet50 [42]	–	–	11.00	73.70
Slim-R50 0.5x [45]	1.10	1.10	6.90	72.10
ResNet34	3.66	3.67	21.80	73.30
Struct-34-A (ours)	1.71	1.71	9.82	72.81
Struct-34-B (ours)	1.47	1.11	5.60	69.44
ResNet18	1.81	1.82	11.69	69.76
Struct-18-A (ours)	0.88	0.89	5.59	69.13
Struct-18-B (ours)	0.79	0.63	3.19	66.19
WeightSVD-R18 [48]	0.89	0.90	5.83	67.71
ChPrune-R18-2x [12]	0.90	0.90	7.45	67.69
ChPrune-R18-4x	0.44	0.45	3.69	61.56

Table: MobileNetV2 and EfficientNet

Architecture	Adds ($\times 10^6$)	Mults ($\times 10^6$)	Params ($\times 10^6$)	Acc. (in %)
MobileNetV2	0.30	0.31	3.50	72.19
Struct-V2-A (ours)	0.26	0.23	2.62	71.29
Struct-V2-B (ours)	0.29	0.17	1.77	64.93
AMC-MV2 [11]	0.21	0.21	–	70.80
ChPrune-MV2-1.3x	0.22	0.23	2.58	68.99
Slim-MV2 [45]	0.21	0.21	2.60	68.90
WeightSVD 1.3x	0.22	0.23	2.45	67.48
ChPrune-MV2-2x	0.15	0.15	1.99	63.82

Architecture	Adds ($\times 10^6$)	Mults ($\times 10^6$)	Params ($\times 10^6$)	Acc. (in %)
EfficientNet-B1 [37]	0.70	0.70	7.80	78.50[1]
Struct-EffNet (ours)	0.62	0.45	4.93	76.40[1]
EfficientNet-B0 [37]	0.39	0.39	5.30	76.10[1]

Concluding remarks

- In this work, we proposed Composite Kernels and Structured Convolutions in an attempt to exploit redundancy in the implicit structure of convolution kernels
- We show that Structured Convolutions can be decomposed into a computationally cheap sum-pooling component followed by a significantly smaller convolution, by training the model using a Structural Regularization loss.
- Sum-pooling relies purely on additions, which are known to be extremely power-efficient. **Hence, our method shows promise in deploying deep models on low-power devices.**
- Since our method keeps the convolutional structures, it allows integration of further model compression schemes, which we leave as future work.