Constructing Fast Network through Deconstruction of Convolution

Yunho Jeon and Junmo Kim

School of Electrical Engineering
Korea Advanced Institute of Science and Technology

NeurIPS 2018

Goal

CNN has achieved outstanding accuracy with deeper and wider networks

Can we make fast CNN with smaller resources while retaining accuracy?

How to make a fast network

- *Reduce FLOPs*
  - Grouped or depthwise convolution
  - Network pruning

How to make a fast network

- *Reduce FLOPs*
  - Grouped or depthwise convolution
  - Network pruning

→*But, Lower FLOPs ≠ Faster speed due to memory access!*

How to make a fast network

- *Reduce FLOPs*
  - Grouped or depthwise convolution
  - Network pruning

  →*But, Lower FLOPs ≠ Faster speed due to memory access!*

- *Reduce memory access*
  - Reduce spatial convolutions

- *Maximize utilization of accessed memory*
  - Use 1x1 convolutions

How to make a fast network

- *Reduce FLOPs*
  - Grouped or depthwise convolution
  - Use 1x1 convolutions
- *R*
- *M*

Key Idea

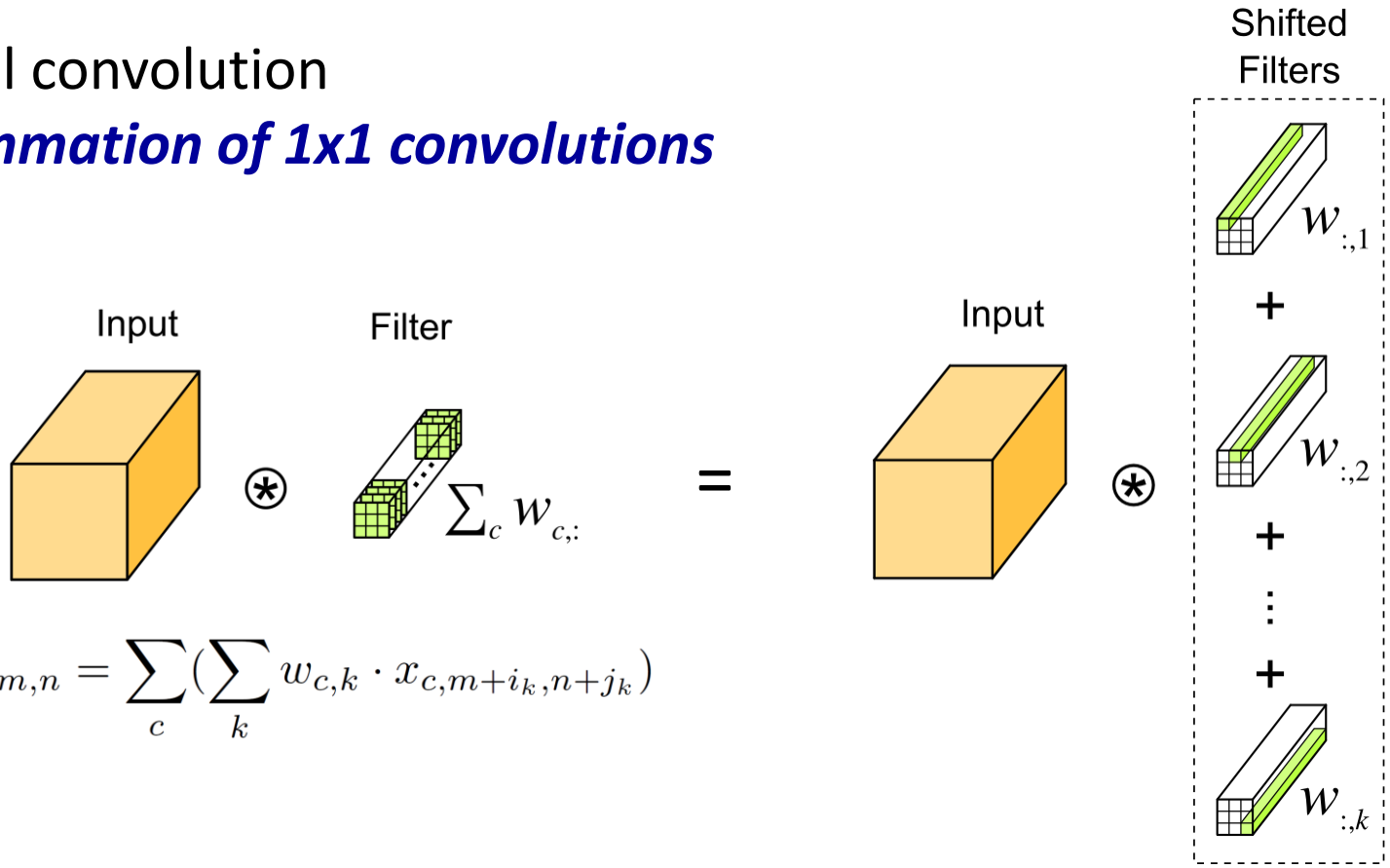
Deconstruct spatial convolution

into atomic operations

Deconstruction of convolution (1/3)

Insight

- Spatial convolution
  = ***Summation of 1x1 convolutions***

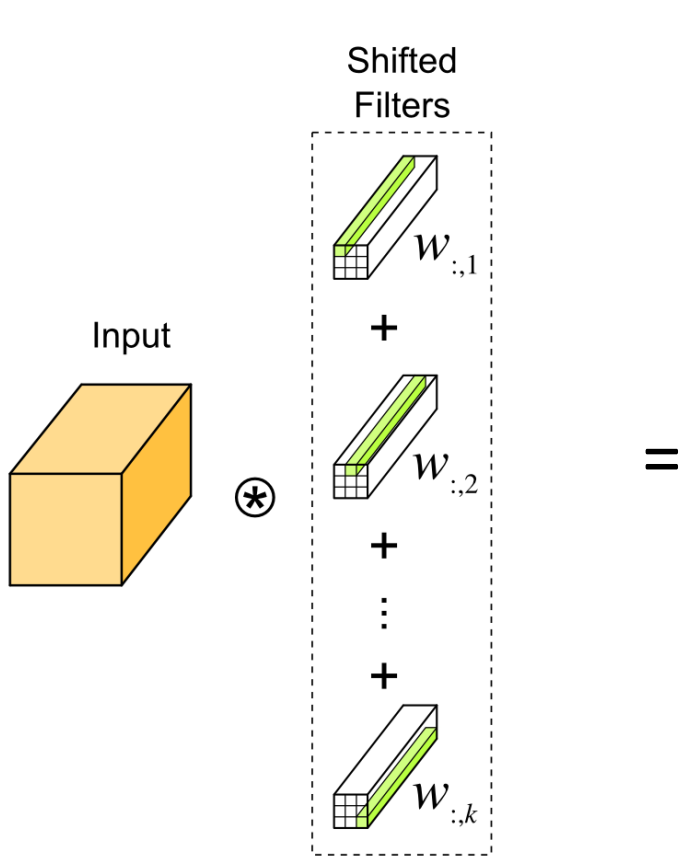


$$y_{m,n} = \sum_c (\sum_k w_{c,k} \cdot x_{c,m+i_k,n+j_k})$$

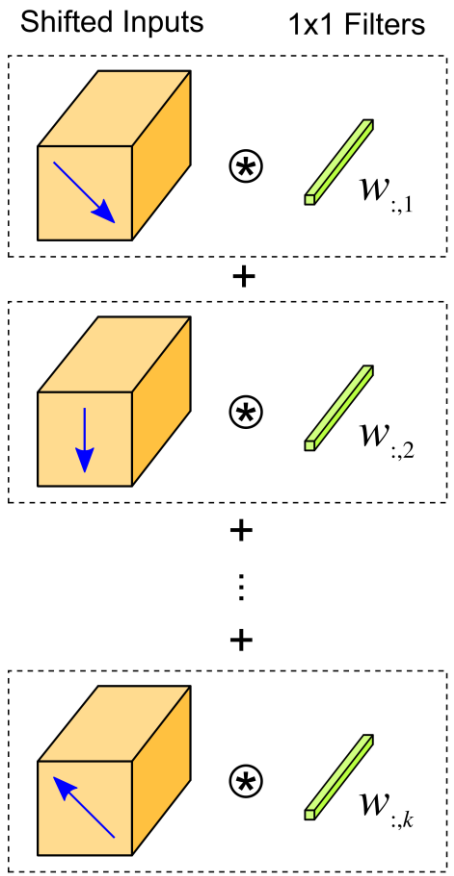
$$y_{m,n} = \sum_k (\sum_c w_{c,k} \cdot x_{c,m+i_k,n+j_k})$$

Deconstruction of convolution (2/3)

Shift Inputs *instead of filters*



=



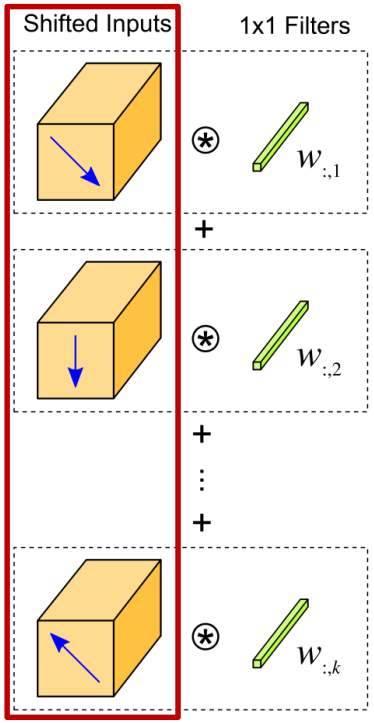
$$y_{m,n} = \sum_k (\sum_c w_{c,k} \cdot x_{c,m+i_k,n+j_k})$$

$$y_{m,n} = \sum_k (\sum_c w_{c,k} \cdot S_k(x_{c,m,n}))$$

Deconstruction of convolution (3/3)

If we can ***share shifted inputs,***

Share shifted inputs



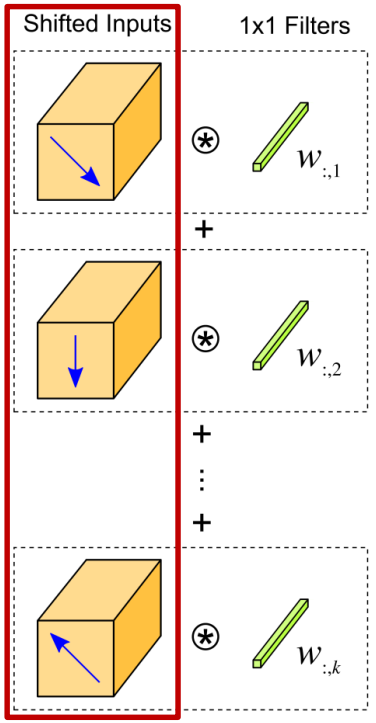
$$y_{m,n} = \sum_k (\sum_c w_{c,k} \cdot S_k(x_{c,m,n}))$$

Deconstruction of convolution (3/3)

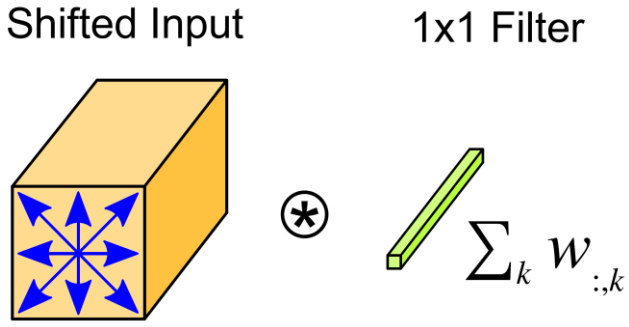
If we can ***share shifted inputs,***

– *Reduce FLOPs & memory access*

Share shifted inputs



$$y_{m,n} = \sum_k (\sum_c w_{c,k} \cdot S_k(x_{c,m,n}))$$



$$y_{m,n} = \sum_c (\sum_k w_{c,k} \cdot S(x_{c,m,n}))$$

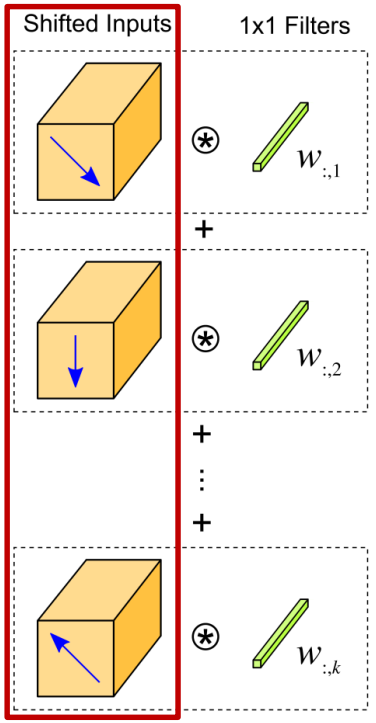
$$y_{m,n} = \sum_c S(x_{c,m,n}) \cdot (\sum_k w_{c,k})$$

Deconstruction of convolution (3/3)

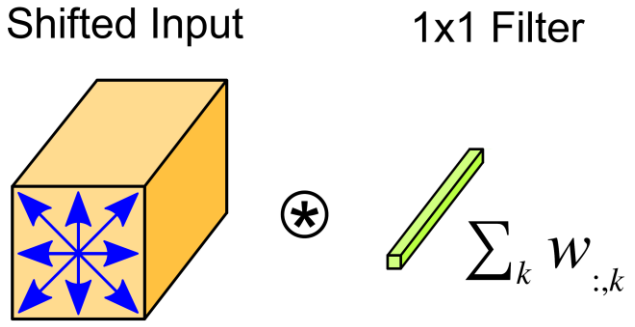
If we can ***share shifted inputs,***

- *Reduce FLOPs & memory access*
- But, *expressive power is limited* *if shifting to one direction*

Share shifted inputs



$$y_{m,n} = \sum_k (\sum_c w_{c,k} \cdot S_k(x_{c,m,n}))$$



$$y_{m,n} = \sum_c (\sum_k w_{c,k} \cdot S(x_{c,m,n}))$$

$$y_{m,n} = \sum_c S(x_{c,m,n}) \cdot (\sum_k w_{c,k})$$

Deconstruction of convolution (3/3)

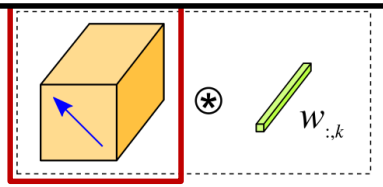
If we can ***share shifted inputs,***

- *Reduce FLOPs & memory access*
- But, *exp*

Key Challenge

How to shift inputs?

Share shifted inputs



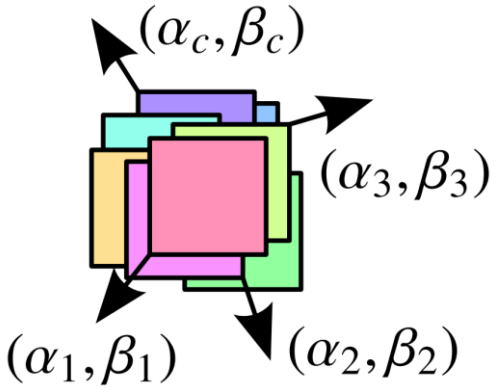
$$y_{m,n} = \sum_k (\sum_c w_{c,k} \cdot S_k(x_{c,m,n}))$$

$$y_{m,n} = \sum_c S(x_{c,m,n}) \cdot (\sum_k w_{c,k})$$

Our approach

- ***Active Shift Layer (ASL)***
  1. Use *depthwise shift*

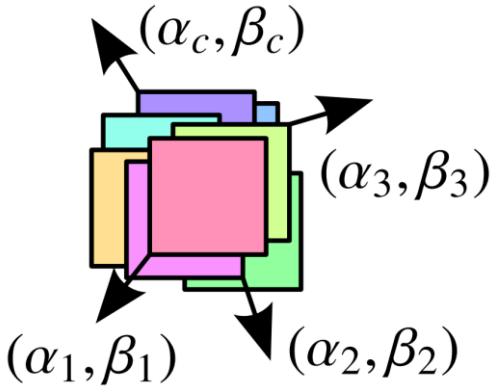
Our approach



- ***Active Shift Layer (ASL)***
  1. Use *depthwise shift*
  2. *Introduce new shift parameters for each channel*

$$\theta_s = \{(\alpha_c, \beta_c) | 1 \leq c \leq C\}$$

Our approach



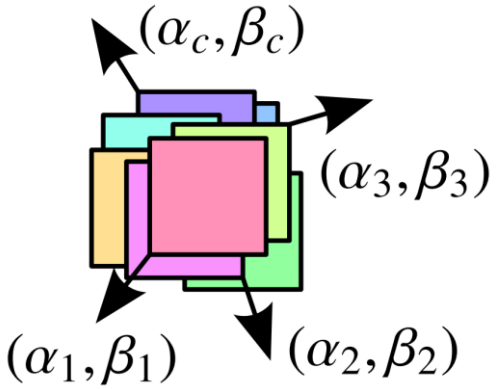
- ***Active Shift Layer (ASL)***
  1. Use *depthwise shift*
  2. *Introduce new shift parameters for each channel*

$$\theta_s = \{(\alpha_c, \beta_c) | 1 \leq c \leq C\}$$

  3. *Expand to non-integer shift* using interpolation

$$\begin{aligned}\tilde{x}_{c,m+\alpha_c,n+\beta_c} =& Z_c^1 \cdot (1-\Delta\alpha_c) \cdot (1-\Delta\beta_c) + Z_c^3 \cdot \Delta\alpha_c \cdot (1-\Delta\beta_c) \\ &+ Z_c^2 \cdot (1-\Delta\alpha_c) \cdot \Delta\beta_c + Z_c^4 \cdot \Delta\alpha_c \cdot \Delta\beta_c, \\ \Delta\alpha_c = \alpha_c - \lfloor\alpha_c\rfloor&, \Delta\beta_c = \beta_c - \lfloor\beta_c\rfloor,\end{aligned}$$

Our approach



- ***Active Shift Layer (ASL)***
  1. Use *depthwise shift*
  2. *Introduce new shift parameters for each channel*

$$\theta_s = \{(\alpha_c, \beta_c) | 1 \leq c \leq C\}$$

  3. *Expand to non-integer shift* using interpolation

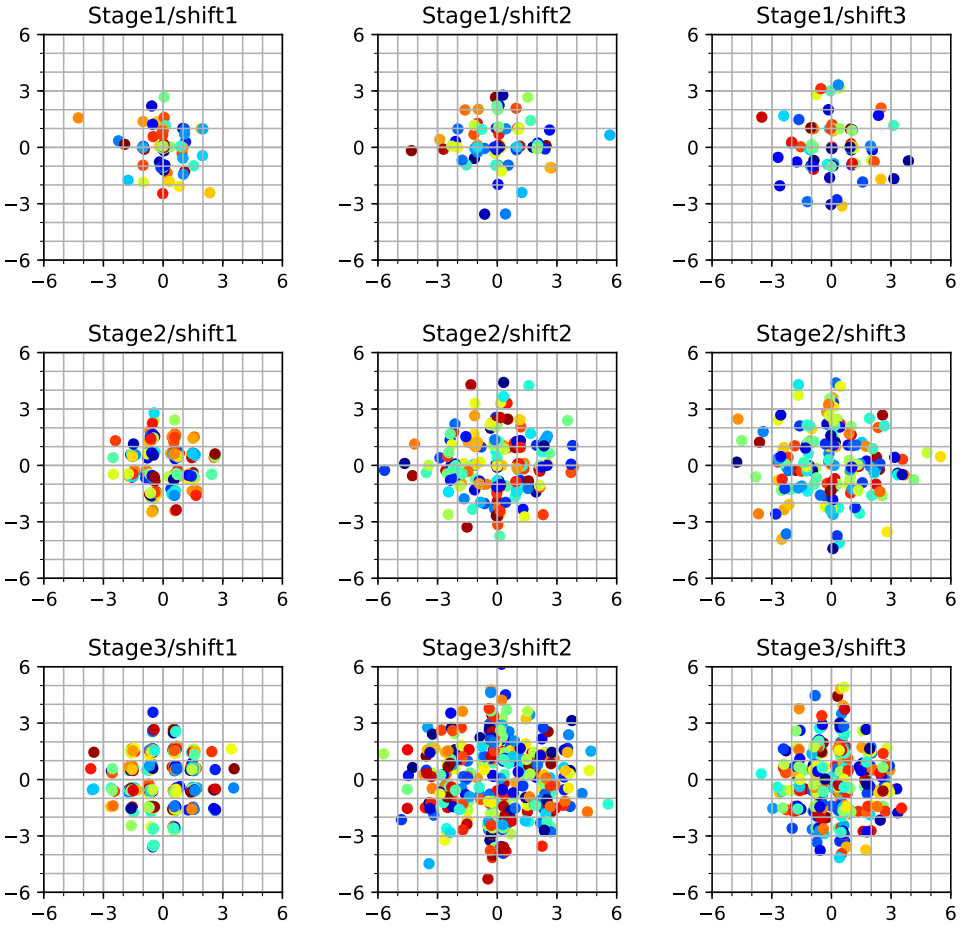
$$\begin{aligned}\tilde{x}_{c,m+\alpha_c,n+\beta_c} =& Z_c^1 \cdot (1-\Delta\alpha_c) \cdot (1-\Delta\beta_c) + Z_c^3 \cdot \Delta\alpha_c \cdot (1-\Delta\beta_c) \\ &+ Z_c^2 \cdot (1-\Delta\alpha_c) \cdot \Delta\beta_c + Z_c^4 \cdot \Delta\alpha_c \cdot \Delta\beta_c, \\ &\Delta\alpha_c = \alpha_c - \lfloor\alpha_c\rfloor, \Delta\beta_c = \beta_c - \lfloor\beta_c\rfloor,\end{aligned}$$

  - *Shift values are differentiable!*

→ *Shift values are trained through network itself*

Example of Learned Shift

Enlarge receptive fields by shifting inputs



Experiment (ImageNet)

- *Better accuracy with the smaller number of parameters*
- *Faster inference time with similar accuracy*

Network	Top-1	Top-5	Param(M)	FLOPs(M)	Inference Time[a] CPU(ms)	GPU(ms)
MobileNetV1[8]	70.6	-	4.2	569	-	-
ShiftNet-A[23]	70.1	89.7	4.1	1.4G	74.1	10.04
MobileNetV2[18]	71.8	**91**	3.47	300	54.7	7.07
AS-ResNet-w68(ours)	**72.2**	90.7	3.42	729	**47.9**	**6.73**
ShuffleNet-×1.5[26]	71.3	-	3.4	292	-	-
MobileNetV2-×0.75	69.8	**89.6**	2.61	209	40.4	6.23
AS-ResNet-w50(ours)	**69.9**	89.3	1.96	404	32.1	6.14
MobileNetV2-×0.5	65.4	86.4	1.95	97	**26.8**	**5.73**
MobileNetV1-×0.5	63.7	-	1.3	149	-	-
SqueezeNet[10]	57.5	80.3	1.2	-	-	-
ShiftNet-B	61.2	83.6	1.1	371	31.8	7.88
AS-ResNet-w32(ours)	**64.1**	**85.4**	0.9	171	**18.7**	**5.37**
ShiftNet-C	58.8	82	0.78	-	-	-

[a]Measured by Caffe [12] using an Intel i7-5930K CPU with a single thread and GTX Titan X (Maxwell). Inference time for MobileNet and ShiftNet (including FLOPs) are measured by using their network description.

Thank you

For more information,
Please visit our ***poster #22***