Training CNNs with Selective Allocation of Channels

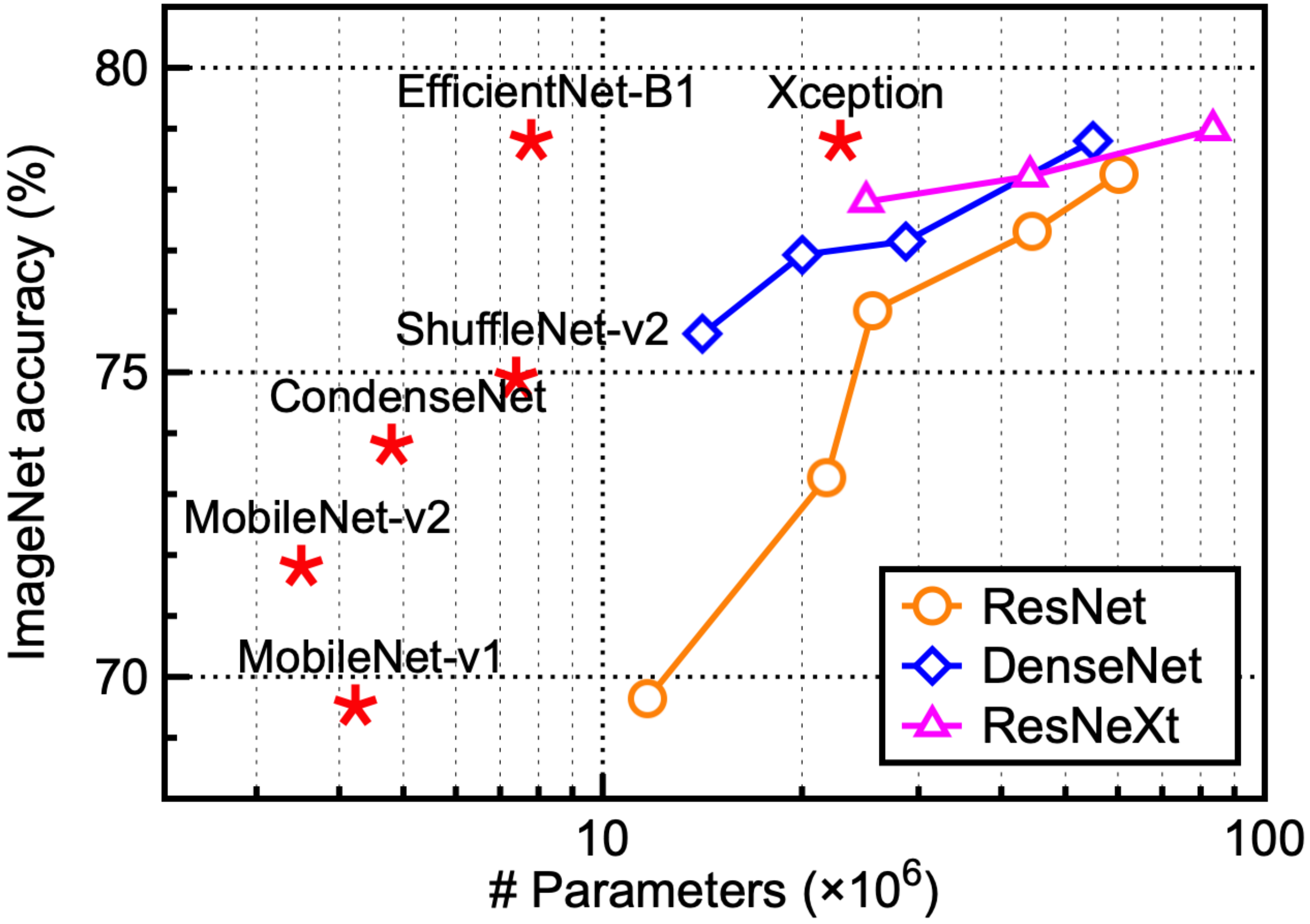
Jongheon Jeong[1]   Jinwoo Shin[1,2]

[1] Korea Advanced Institute of Science and Technology (KAIST)
[2] AITRICS

ICML 2019

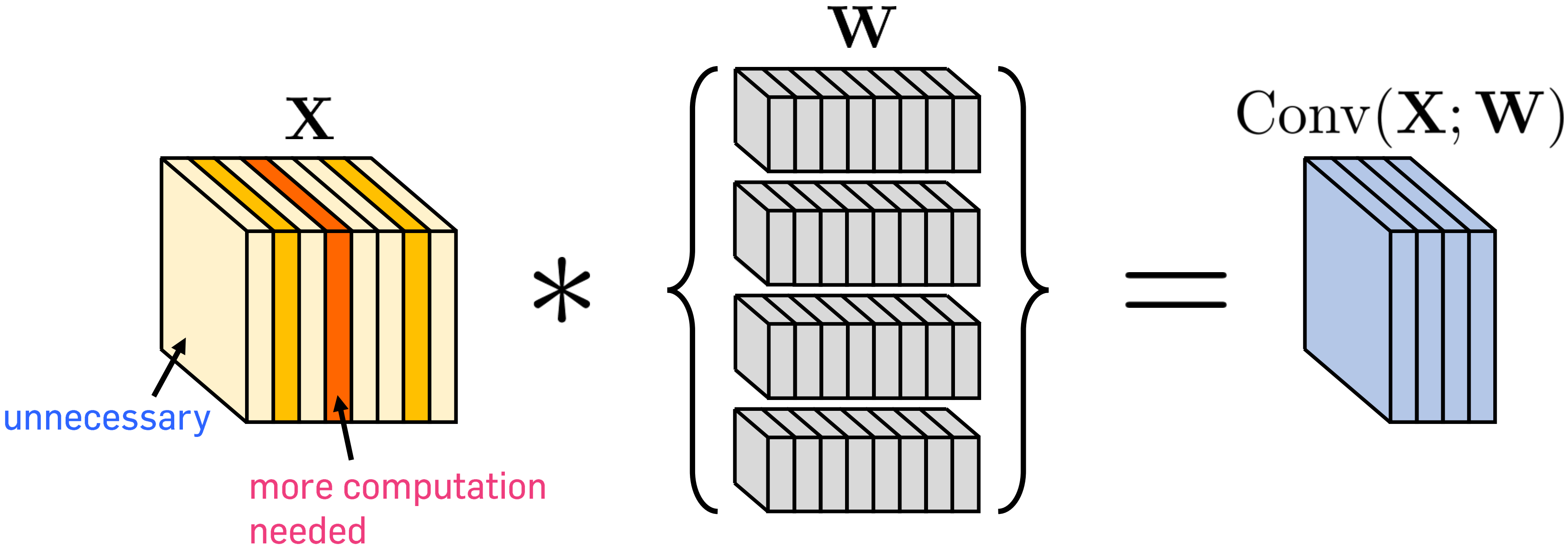
Channel Inefficiency in “Static” CNNs

- CNN architecture design typically focus on static layers



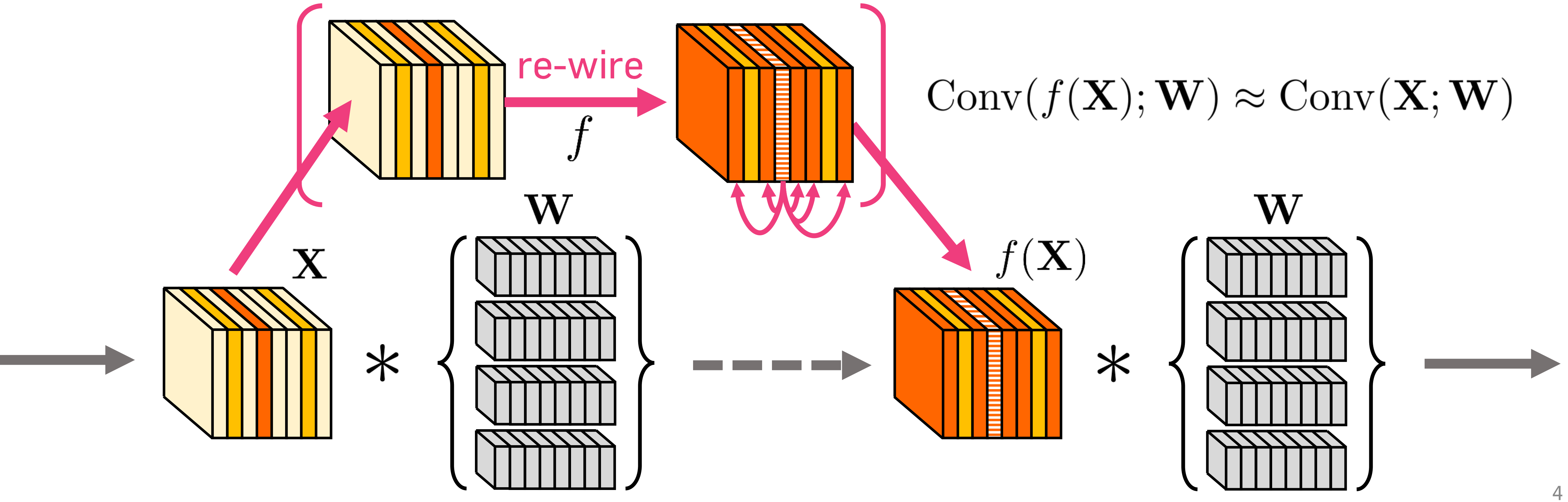
Channel Inefficiency in "Static" CNNs

- Current CNNs allocate parameters uniformly across channels
  - The structure is **fixed** until the end of training
  - Each convolutional layer may contain unnecessary channels to compute
- Can we utilize them in training for efficiency? 🤔



Key Points

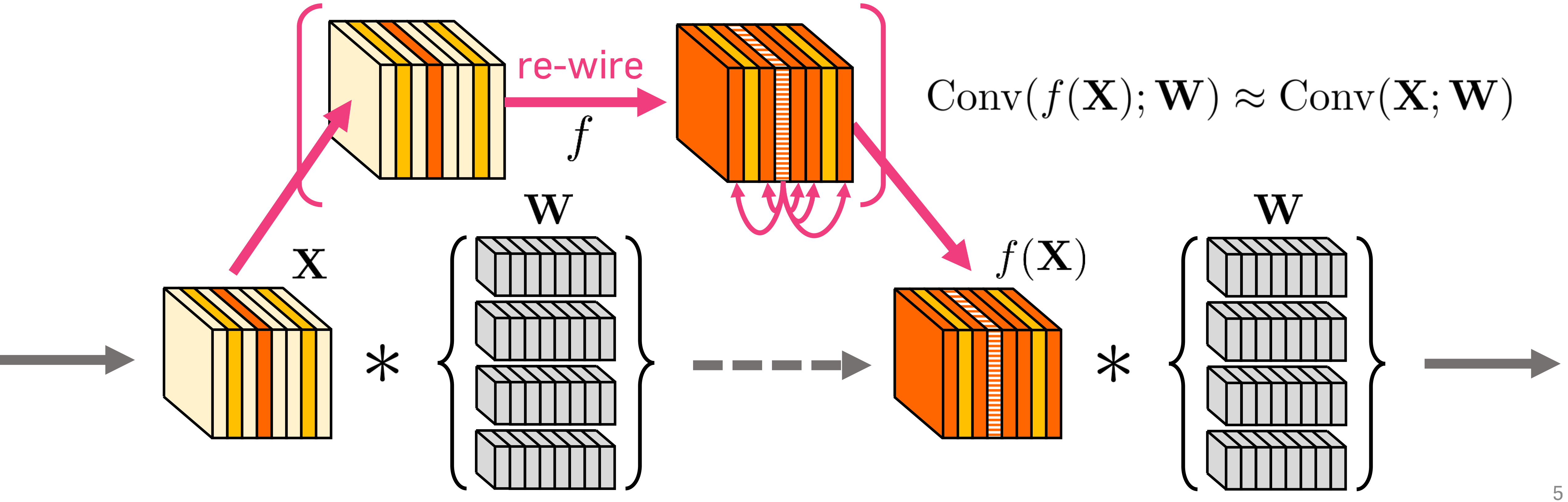
- Idea: Training with dynamic re-wiring operations
- Incorporating function-preserving operations for rewiring channels
  - Connectivity is updated without affecting the overall training



$$\mathrm{Conv}(f(\mathbf{X}); \mathbf{W}) \approx \mathrm{Conv}(\mathbf{X}; \mathbf{W})$$

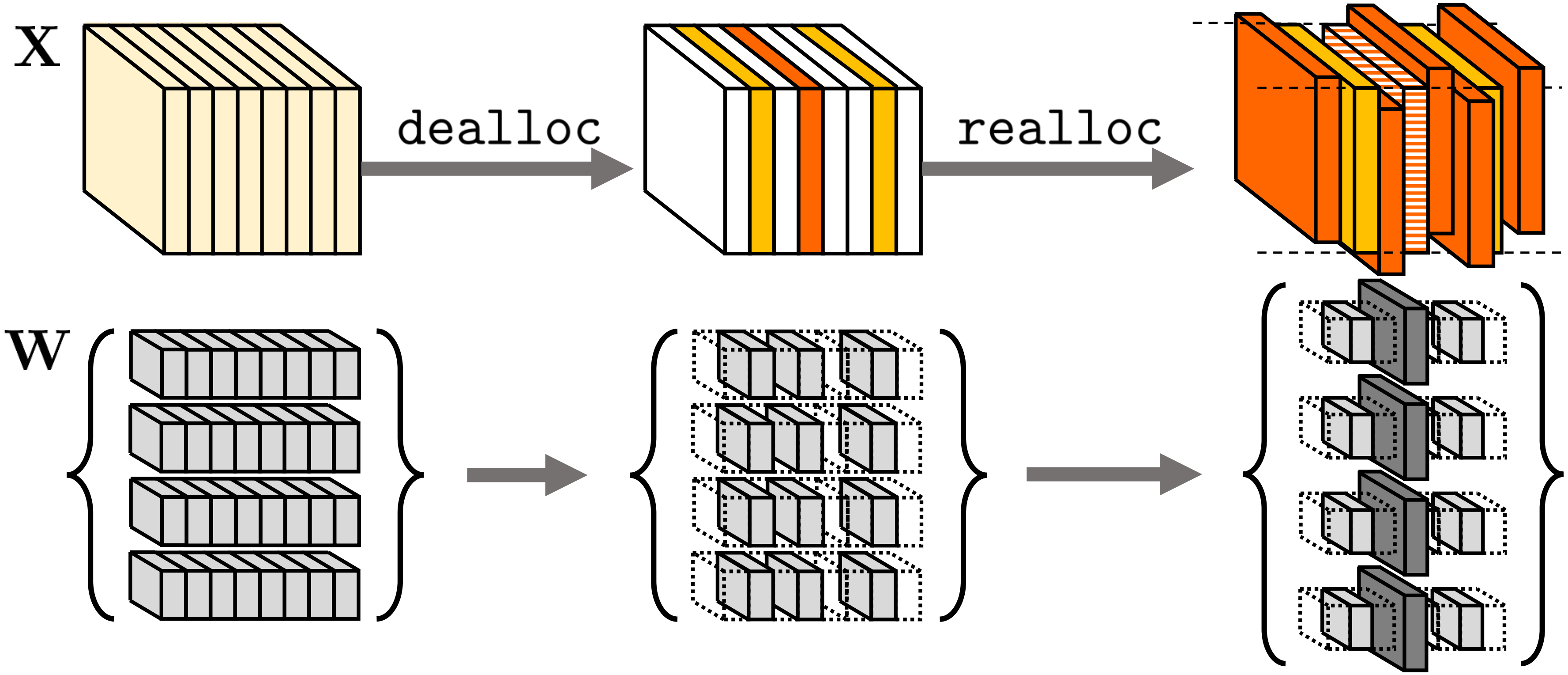
Key Points

- Idea: Training with dynamic re-wiring operations 💡
- Incorporating function-preserving operations for rewiring channels
  - Connectivity is updated without affecting the overall training
  - Manipulation on channels rather than parameters → architecture-agnostic



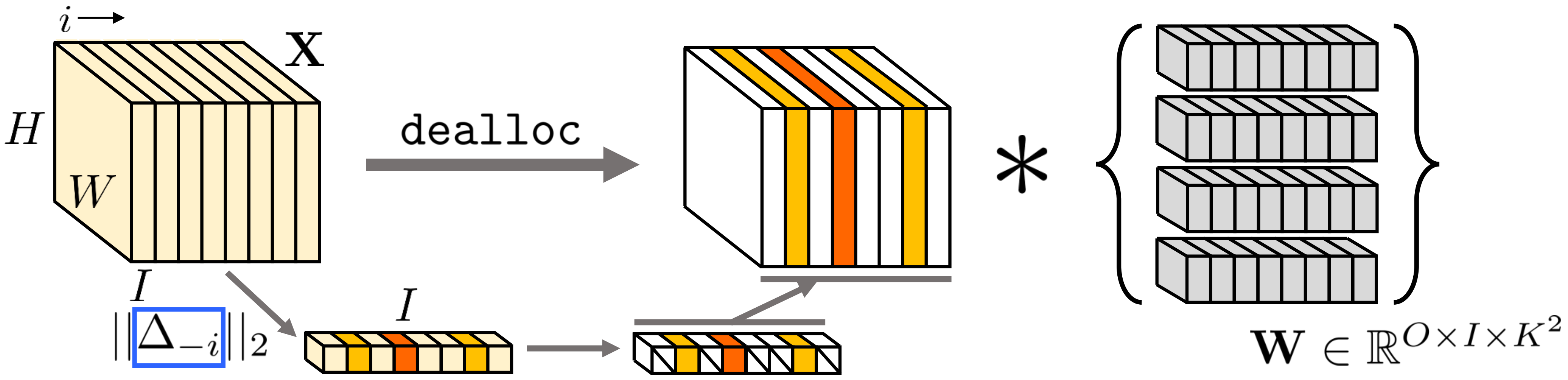
Selective Convolutional Layer

- Idea: Training with dynamic re-wiring operations 💡
- Two function-preserving operations: `dealloc` & `realloc`
  1. `dealloc`: Release unimportant channels → pruning parameters
  2. `realloc`: Replicate important channels → re-using the pruned parameter



Selective Convolutional Layer

- Two operations during training: `dealloc` & `realloc`
  1. **Channel de-allocation (`dealloc`)**: Release "unimportant" channels

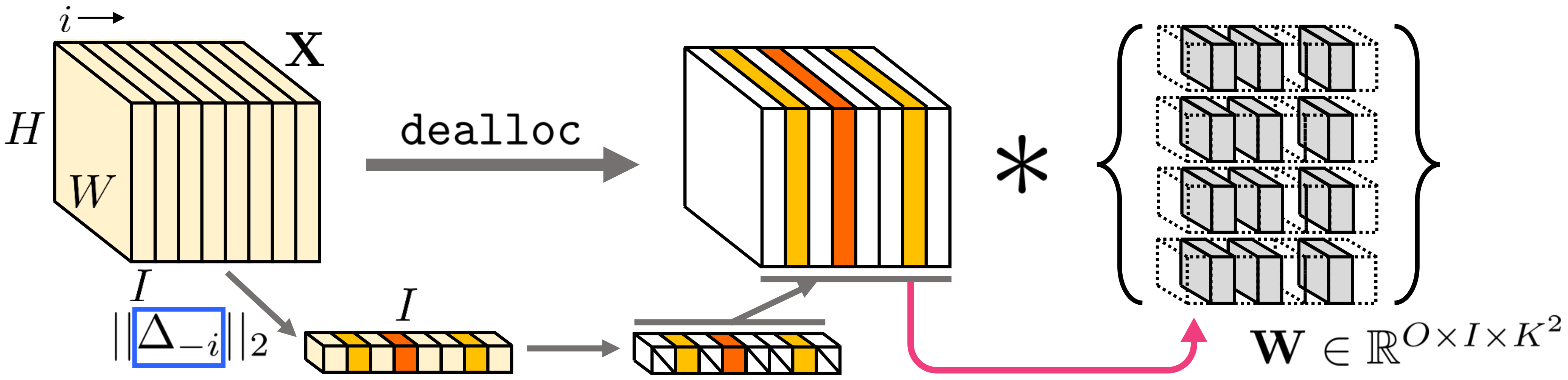


- We measure expected channel damage for channel importance

$$\boxed{\Delta_{-i}} := \frac{1}{HW} \sum_{h,w} \mathbb{E}_{\mathbf{X}}[\mathrm{Conv}(\mathbf{X};\mathbf{W}) - \mathrm{Conv}(\mathbf{X};\underbrace{\mathbf{W}_{-i}}_{\mathbf{W} \text{ but } \mathbf{W}_{i,:,:} = 0})]_{:,h,w} \in \mathbb{R}^O$$

Selective Convolutional Layer

- Two operations during training: `dealloc` & `realloc`
  1. **Channel de-allocation (`dealloc`)**: Release "unimportant" channels



- We measure expected channel damage for channel importance

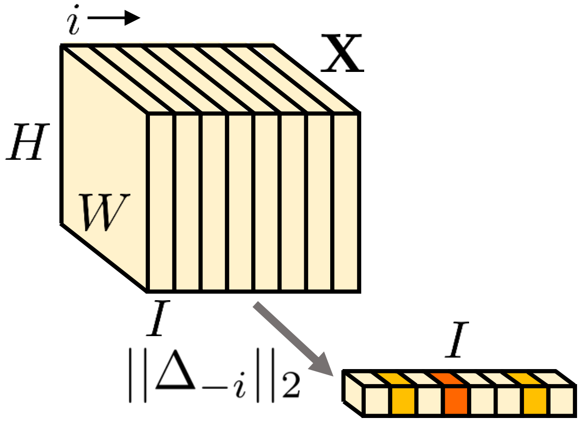
$$\boxed{\Delta_{-i}} := \frac{1}{HW} \sum_{h,w} \mathbb{E}_{\mathbf{X}}[\mathrm{Conv}(\mathbf{X}; \mathbf{W}) - \mathrm{Conv}(\mathbf{X}; \underbrace{\mathbf{W}_{-i}}_{\mathbf{W} \text{ but } \mathbf{W}_{i,:,:} = 0})]_{:,h,w} \in \mathbb{R}^{O}$$

Selective Convolutional Layer

- We measure expected channel damage for channel importance

$$\Delta_{-i} := \frac{1}{HW}\sum_{h,w}\mathbb{E}_{\mathbf{X}}[\mathrm{Conv}(\mathbf{X};\mathbf{W}) - \mathrm{Conv}(\mathbf{X};\underbrace{\mathbf{W}_{-i}}_{\mathbf{W}\text{ but }\mathbf{W}_{i,:,:}=0})]_{:,h,w} \in \mathbb{R}^{O}$$

  - Difference after pruning channel i → function-preserving property

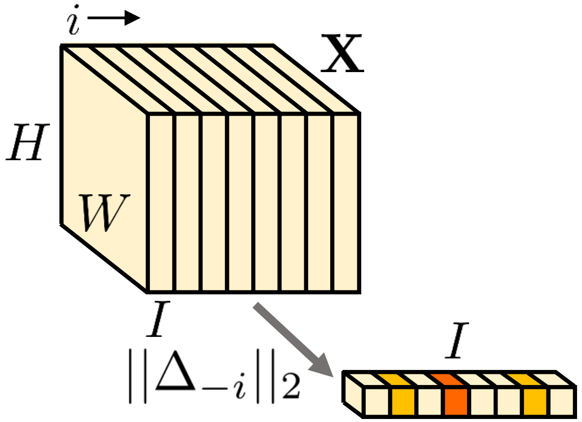


Selective Convolutional Layer

- We measure expected channel damage for channel importance

$$\Delta_{-i} := \frac{1}{HW}\sum_{h,w}\mathbb{E}_{\mathbf{X}}[\mathrm{Conv}(\mathbf{X};\mathbf{W}) - \mathrm{Conv}(\mathbf{X};\underbrace{\mathbf{W}_{-i}}_{\mathbf{W}\text{ but }\mathbf{W}_{i,:,:}=0})]_{:,h,w} \in \mathbb{R}^O$$

  - Difference after pruning channel i → function-preserving property



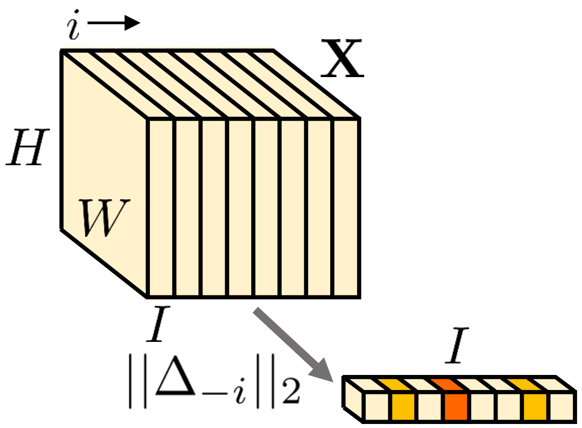
- Challenge: Computing Δ_{-i} requires a marginalization over $\mathbf{X}$ 🤔

Selective Convolutional Layer

- We measure expected channel damage for channel importance

$$\Delta_{-i} := \frac{1}{HW}\sum_{h,w}\mathbb{E}_{\mathbf{X}}[\mathrm{Conv}(\mathbf{X};\mathbf{W}) - \mathrm{Conv}(\mathbf{X};\underbrace{\mathbf{W}_{-i}}_{\mathbf{W}\text{ but }\mathbf{W}_{i,:,:}=0})]_{:,h,w} \in \mathbb{R}^O$$

  - Difference after pruning channel i → function-preserving property



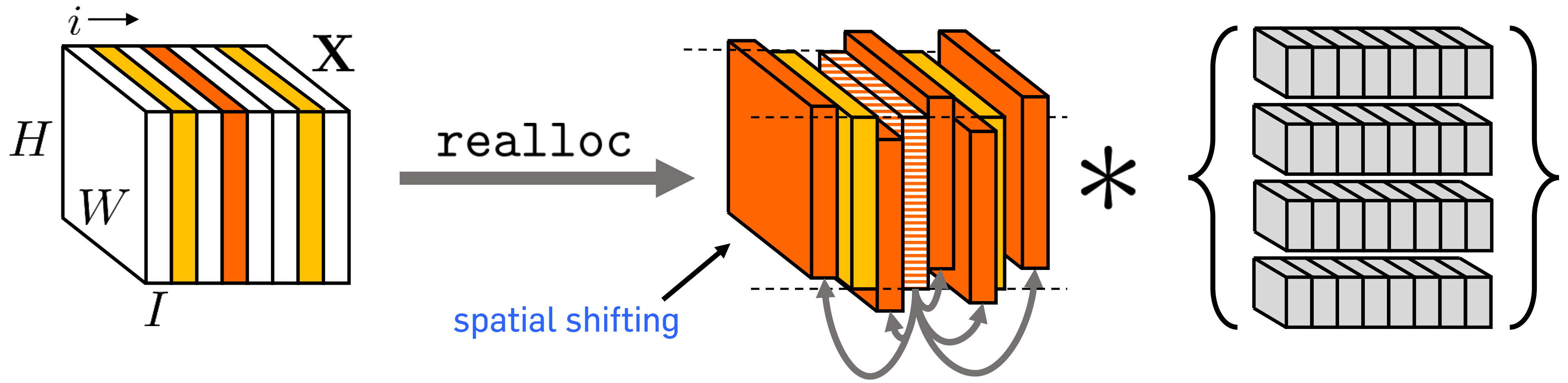
- Challenge: Computing Δ_{-i} requires a marginalization over $\mathbf{X}$ 🤔
- Idea: Use BatchNorm statistics to approximate Δ_{-i} 💡
  - Assuming $\underbrace{\mathrm{BN}(x) \sim \mathcal{N}(\beta, \gamma^2)}_{\text{BN parameters}}$ and $\underbrace{\mathbf{X} = \mathrm{ReLU}(\mathrm{BN}(\mathbf{Y}))}_{\text{Common design}}$, we get:

$$\Delta_{-i} \approx \underbrace{\left(|\gamma_i|\phi_{\mathcal{N}}\left(\frac{\beta_i}{|\gamma_i|}\right) + \beta_i\Phi_{\mathcal{N}}\left(\frac{\beta_i}{|\gamma_i|}\right)\right)}_{\text{activity level}} \cdot \underbrace{\sum_{k=1}^{K^2}\mathbf{W}_{i,:,k}}_{\text{magnitude}}$$

p.d.f. c.d.f.

Selective Convolutional Layer

- Two operations during training: `dealloc` & `realloc`
  2. Channel re-allocation (`realloc`): Replicate "important" channels into the released area



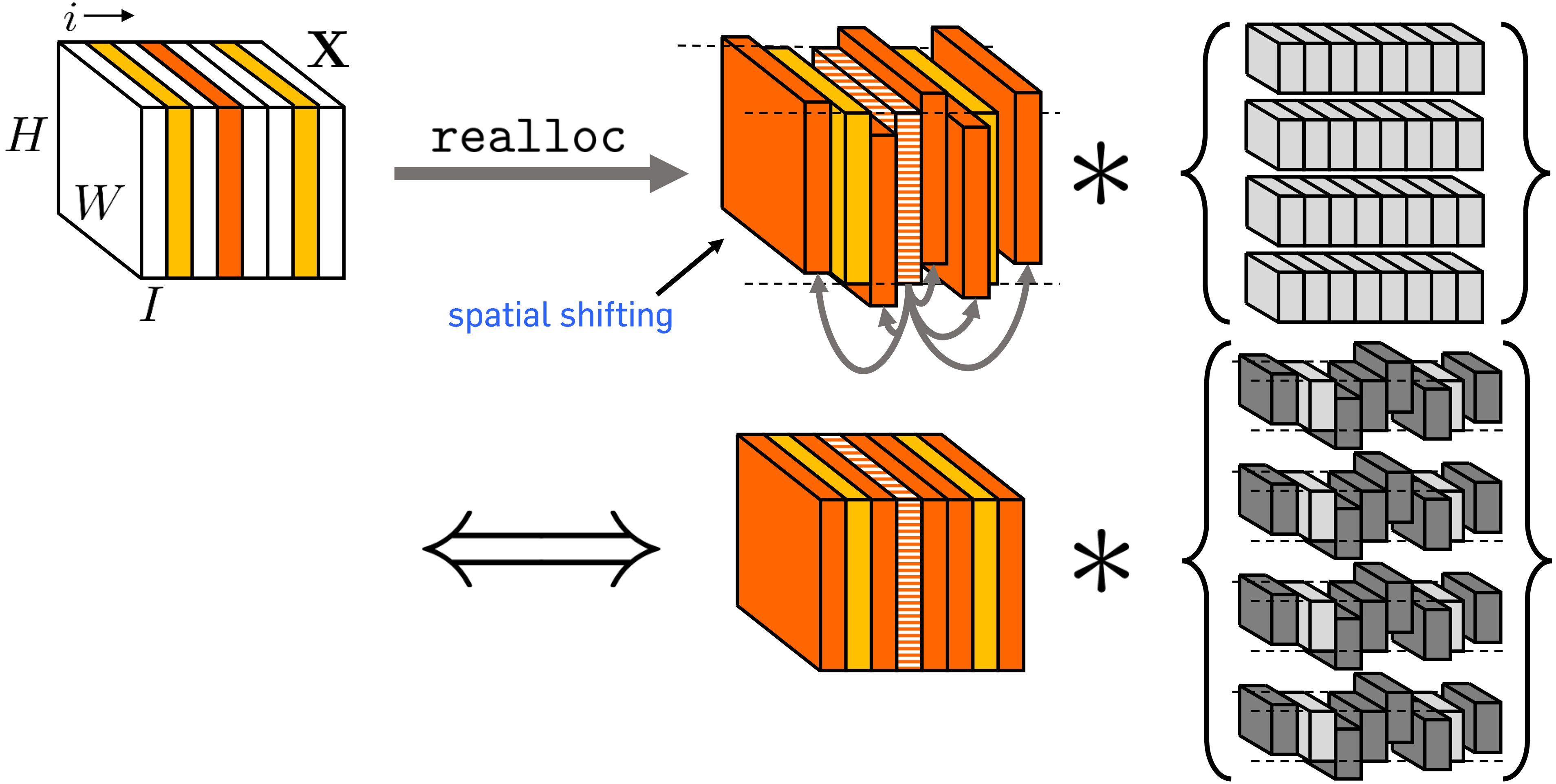
- Channels with high $||\Delta_{-i}||_2$ are copied, but with spatial shifting bias $\underbrace{b = (b^h, b^w)}_{\text{learnable}} \in \mathbb{R}^2$

$$\mathrm{shift}\left(\mathbf{X}, b\right)_{x,y} := \sum_{n=1}^{H} \sum_{m=1}^{W} \mathbf{X}_{n,m} \times \max\left(0, 1 - |x - n + b^h|\right) \times \max\left(0, 1 - |y - m + b^w|\right)$$

"bilinear interpolation kernel"

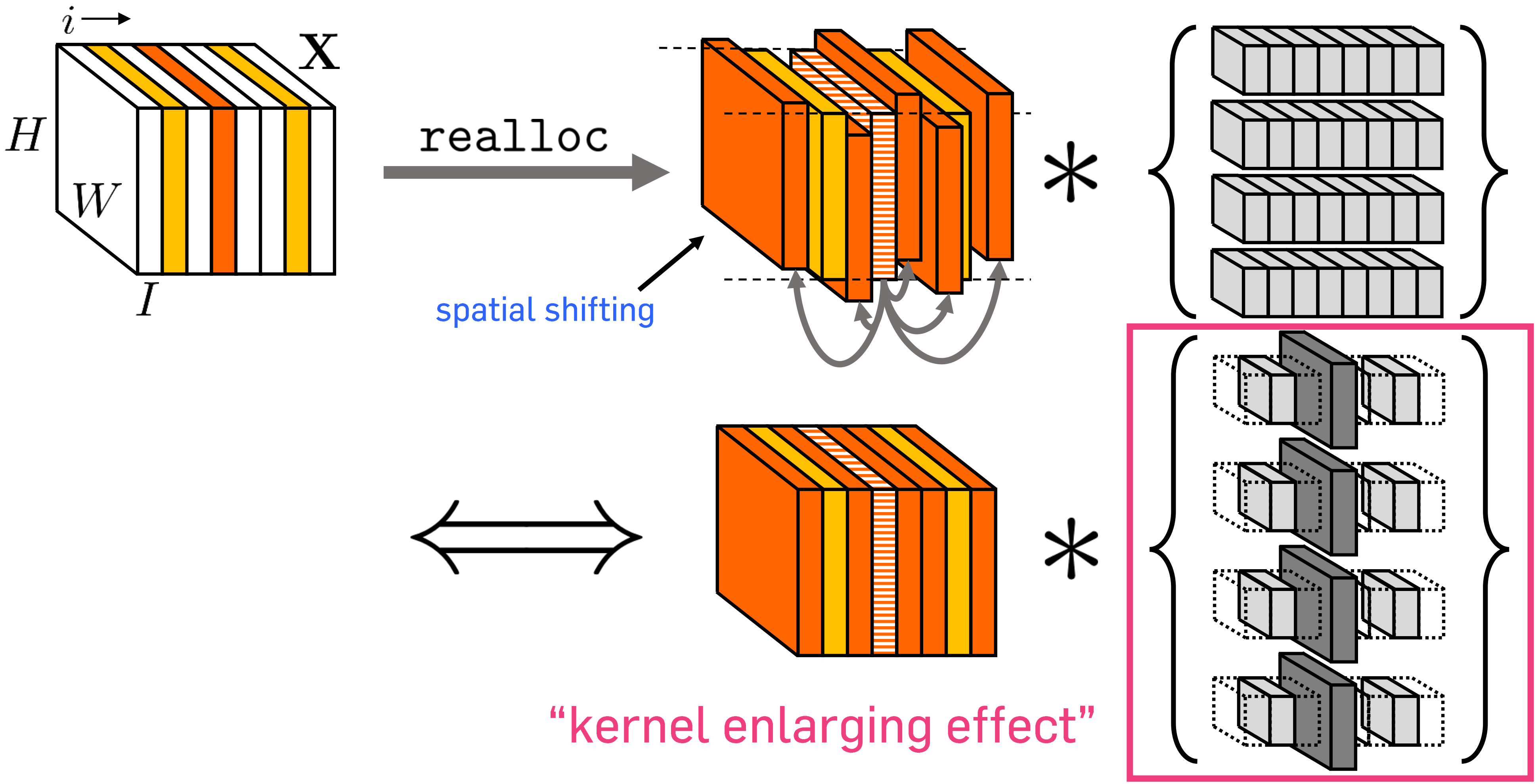
Selective Convolutional Layer

- Two operations during training: `dealloc` & `realloc`
  2. **Channel re-allocation (`realloc`)**: Replicate "important" channels into the released area



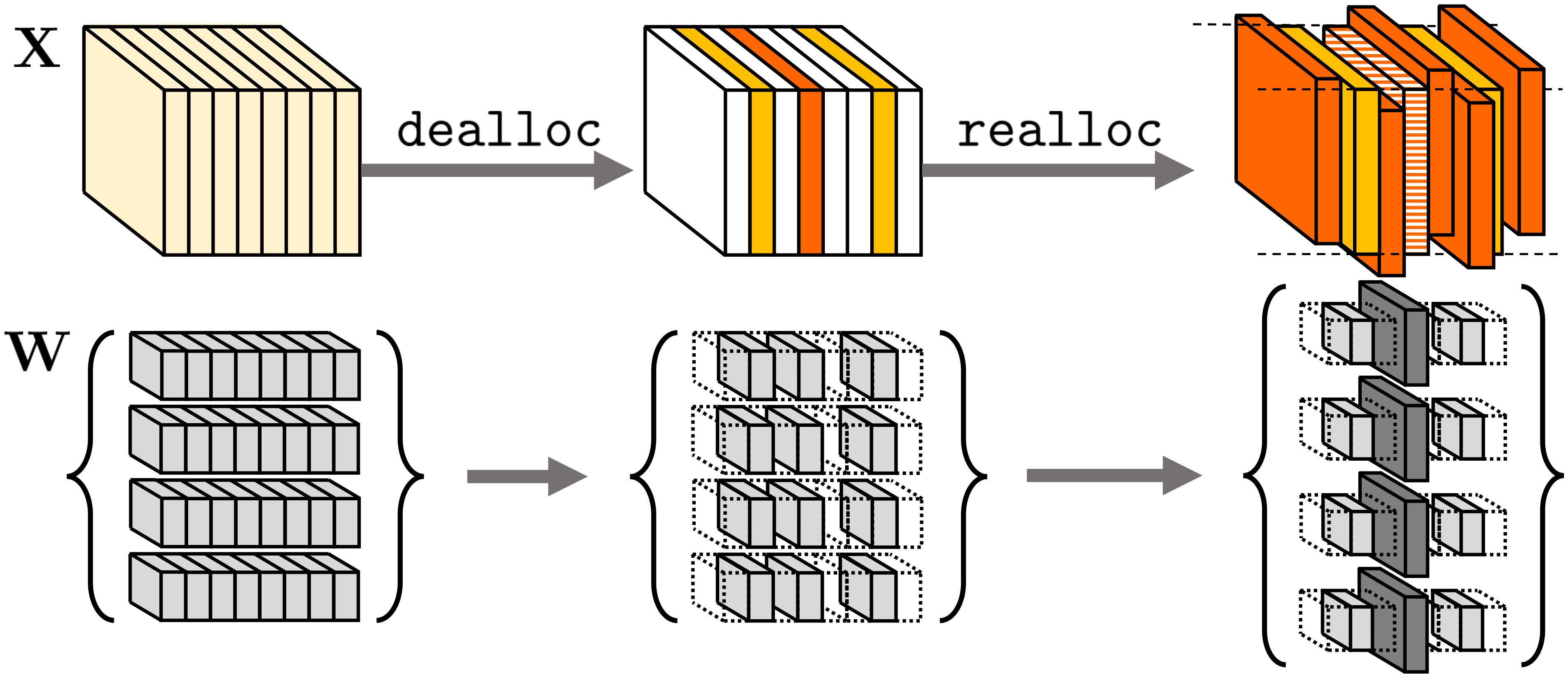
Selective Convolutional Layer

- Two operations during training: `dealloc` & `realloc`
  2. **Channel re-allocation (`realloc`)**: Replicate "important" channels into the released area



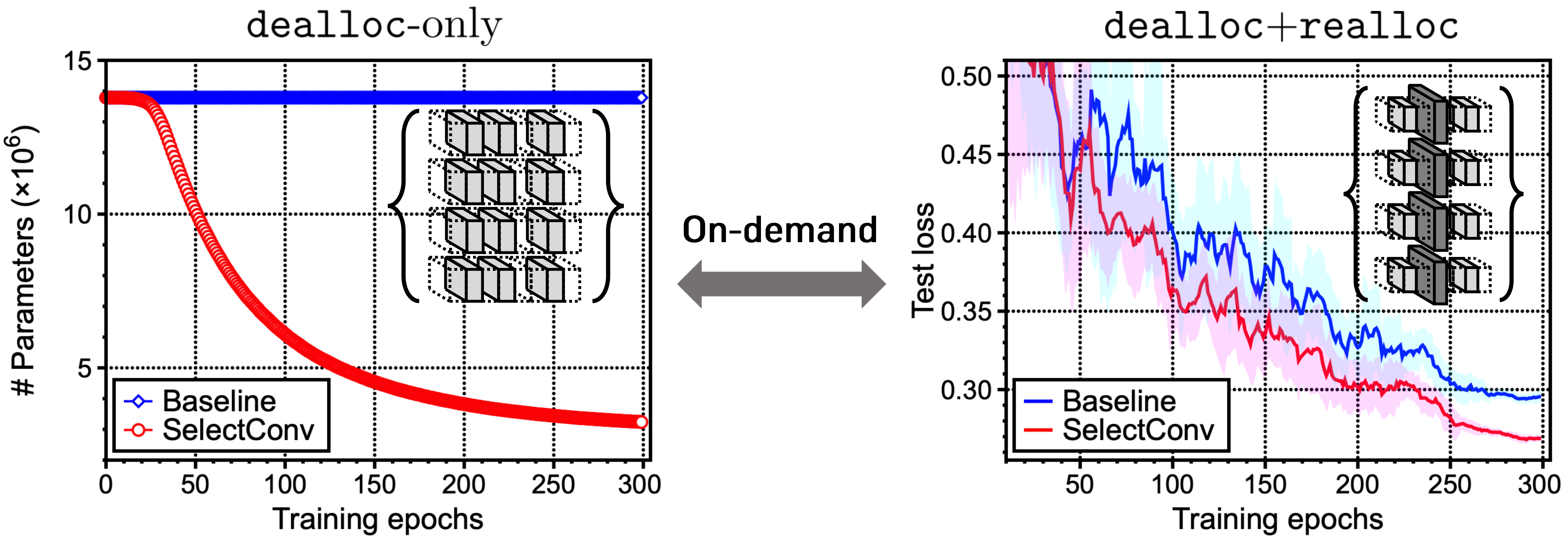
Selective Convolutional Layer

- Idea: Dynamic re-wiring of parameters → selective kernel expansion
- Two function-preserving operations: `dealloc` & `realloc`



Selective Convolutional Layer

- **Framework**: Incorporating **re-wiring operations** in training
- Two operations during training: `dealloc` & `realloc`
  - Flexible training: model reduction ↔ accuracy improvement



Experiments: Improving Modern CNNs

- Selective convolution can be readily applied to various existing CNNs

Model	Params	Method	Error rates (%) CIFAR-10	CIFAR-100	Fashion-MNIST	Tiny-ImageNet
DenseNet-40 (bottleneck, $k = 12$)	0.21M	Baseline	6.62±0.15	29.9±0.1	5.03±0.07	45.8±0.2
		SelectConv	**6.09**±0.10 (**-8.01%**)	**28.8**±0.1 (**-3.42%**)	**4.73**±0.06 (**-5.96%**)	**44.4**±0.2 (**-3.03%**)
DenseNet-100 (bottleneck, $k = 12$)	1.00M	Baseline	4.51±0.04	22.8±0.3	4.70±0.06	41.0±0.1
		SelectConv	**4.29**±0.08 (**-4.88%**)	**22.2**±0.1 (**-2.64%**)	**4.58**±0.05 (**-2.55%**)	**39.9**±0.3 (**-2.78%**)
ResNet-164 (bottleneck, pre-act)	1.66M	Baseline	4.23±0.15	21.3±0.2	4.53±0.04	37.7±0.4
		SelectConv	**3.92**±0.14 (**-7.33%**)	**20.9**±0.2 (**-1.97%**)	**4.37**±0.03 (**-3.53%**)	**37.5**±0.2 (**-0.56%**)
ResNeXt-29 ($8 \times 64d$)	33.8M	Baseline	3.62±0.12	18.1±0.1	4.40±0.07	31.7±0.3
		SelectConv	**3.39**±0.14 (**-6.36%**)	**17.6**±0.1 (**-2.92%**)	**4.27**±0.06 (**-2.95%**)	**31.4**±0.3 (**-0.88%**)

- (top) Results on CIFAR-10/100, FMNIST and Tiny-ImageNet
- (right) Results on ImageNet dataset

Model	Params	Method	Error (%)
DenseNet-121 ($k = 32$)	7.98M	Baseline	24.7
		SelectConv	**24.4**
ResNet-50 (bottleneck)	22.8M	Baseline	23.9
		SelectConv	**23.4**

Experiments: Improving Modern CNNs

- Selective convolution can be readily applied to various existing CNNs
- Reduction in error rates across all the tested architectures

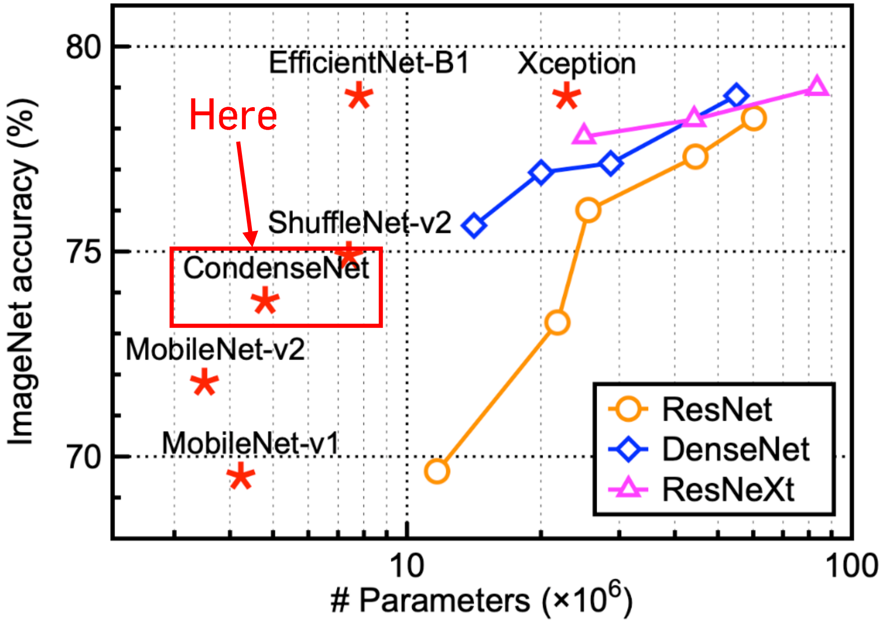
Model	Params	Method	Error rates (%) CIFAR-10	CIFAR-100	Fashion-MNIST	Tiny-ImageNet
DenseNet-40 (bottleneck, $k = 12$)	0.21M	Baseline	6.62±0.15	29.9±0.1	5.03±0.07	45.8±0.2
		SelectConv	**6.09**±0.10 (-8.01%)	**28.8**±0.1 (-3.42%)	**4.73**±0.06 (-5.96%)	**44.4**±0.2 (-3.03%)
DenseNet-100 (bottleneck, $k = 12$)	1.00M	Baseline	4.51±0.04	22.8±0.3	4.70±0.06	41.0±0.1
		SelectConv	**4.29**±0.08 (-4.88%)	**22.2**±0.1 (-2.64%)	**4.58**±0.05 (-2.55%)	**39.9**±0.3 (-2.78%)
ResNet-164 (bottleneck, pre-act)	1.66M	Baseline	4.23±0.15	21.3±0.2	4.53±0.04	37.7±0.4
		SelectConv	**3.92**±0.14 (-7.33%)	**20.9**±0.2 (-1.97%)	**4.37**±0.03 (-3.53%)	**37.5**±0.2 (-0.56%)
ResNeXt-29 ($8 \times 64d$)	33.8M	Baseline	3.62±0.12	18.1±0.1	4.40±0.07	31.7±0.3
		SelectConv	**3.39**±0.14 (-6.36%)	**17.6**±0.1 (-2.92%)	**4.27**±0.06 (-2.95%)	**31.4**±0.3 (-0.88%)

- (top) Results on CIFAR-10/100, FMNIST and Tiny-ImageNet
- (right) Results on ImageNet dataset

Model	Params	Method	Error (%)
DenseNet-121 ($k = 32$)	7.98M	Baseline	24.7
		SelectConv	**24.4**
ResNet-50 (bottleneck)	22.8M	Baseline	23.9
		SelectConv	**23.4**

Experiments: Mobile-targeted Architectures

- Selective convolution can further improve the "already-efficient" CondenseNet-182
- Training with `dealloc` → model compression



Model	Params	FLOPs	Error (%)
ResNet-1001	16.1M	2,357M	4.62
WideResNet-28-10	36.5M	5,248M	4.17
ResNeXt-29 (16 × 64d)	68.1M	10,704M	3.58
VGGNet-Slim [2]	2.30M	391M	6.20
ResNet-164-Slim [2]	1.10M	275M	5.27
CondenseNet-182 [1]	4.20M	513M	3.76
CondenseNet-SConv-182	**3.24M**	**422M**	**3.50**

[1] Gao Huang et al. Condensenet: An efficient densenet using learned group convolutions. CVPR 2018.
[2] Zhuang Liu et al. Learning efficient convolutional networks through network slimming. ICCV 2017.

Summary

- We propose selective convolution = convolution + channel-selectivity
  1. **Generic, easy to use**: applicable to any kind of CNN
  2. **Single-pass**: no post-processing/re-training
  3. **On-demand**: accuracy improvement ⟷ model compression
- We define a new metric of channel importance: expected channel damage

Poster #17

Wed Jun 12th 6:30 – 9:00 PM

@ Pacific Ballroom