Training CNNs with Selective Allocation of Channels

Jongheon Jeong\textsuperscript{1} \hspace{1cm} Jinwoo Shin\textsuperscript{1,2}

\textsuperscript{1}Korea Advanced Institute of Science and Technology (KAIST) \\
\textsuperscript{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?**

\[ \text{X} \times \left\{ \begin{array}{c} \text{W} \\ \text{unnecessary} \\ \text{more computation needed} \end{array} \right\} = \text{Conv}(\text{X}; \text{W}) \]
Key Points

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

\[
\text{Conv}(f(X); W) \approx \text{Conv}(X; W)
\]
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

\[ \Delta_{-i} := \frac{1}{HW} \sum_{h,w} \mathbb{E}_{X} [\text{Conv}(X; W) - \text{Conv}(X; \underline{W_{-i}})]; \quad h, w \in \mathbb{R}^{O} \]

- **We measure expected channel damage** for channel importance

\[ \text{W} \in \mathbb{R}^{O \times I \times K^2} \]
Selective Convolutional Layer

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

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

- **We measure expected channel damage** for channel importance
Selective Convolutional Layer

- We measure **expected channel damage** for channel importance:

\[
\Delta_{-i} := \frac{1}{HW} \sum_{h, w} \mathbb{E}_x [\text{Conv}(X; W) - \text{Conv}(X; W_{-i})]; h, w \in \mathbb{R}^O
\]

- Difference after pruning channel \( i \) → **function-preserving property**
Selective Convolutional Layer

- We measure **expected channel damage** for channel importance

\[
\Delta_{-i} := \frac{1}{HW} \sum_{h,w} \mathbb{E}_X [\text{Conv}(X; W) - \text{Conv}(X; W_{-i})], h, w \in \mathbb{R}^O
\]

- Difference after pruning channel \( i \) → **function-preserving property**

- **Challenge:** Computing \( \Delta_{-i} \) requires a marginalization over \( X \) 😞
Selective Convolutional Layer

- We measure **expected channel damage** for channel importance

\[
\Delta_{-i} := \frac{1}{HW} \sum_{h,w} \mathbb{E}_X [\text{Conv}(X; W) - \text{Conv}(X; W_{-i})]; h,w \in \mathbb{R}^O
\]

- Difference after pruning channel \(i\) \(\rightarrow\) **function-preserving property**

- **Challenge:** Computing \(\Delta_{-i}\) requires a marginalization over \(X\) 😞

- **Idea:** Use BatchNorm statistics to approximate \(\Delta_{-i}\)
  - Assuming \(\text{BN}(x) \sim \mathcal{N}(\beta, \gamma^2)\) and \(X = \text{ReLU} (\text{BN}(Y))\), we get:

\[
\Delta_{-i} \approx \left( |\gamma_i| \phi_N \left( \frac{\beta_i}{|\gamma_i|} \right) + \beta_i \Phi_N \left( \frac{\beta_i}{|\gamma_i|} \right) \right) \cdot \sum_{k=1}^{K^2} W_{i,:,k}
\]

- **Common design**

- **BN parameters**
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** $b = (b^h, b^w) \in \mathbb{R}^2$ learnable

\[
\text{shift} \ (X, b)_{x,y} := \sum_{n=1}^{H} \sum_{m=1}^{W} X_{n,m} \times \max \left(0, 1 - |x - n + b^h| \right) \times \max \left(0, 1 - |y - m + b^w| \right)
\]
Selective Convolutional Layer

- Two operations during training: deallocate & reallocate

  2. Channel re-allocation (realloc): Replicate “important” channels into the released area
Selective Convolutional Layer

- **Two operations** during training: **deallocate** & **reallocate**
  2. **Channel re-allocation (reallocate):** 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

![Graph showing the comparison between dealloc-only and dealloc+realloc]

- **On-demand**

![Graph showing the comparison between baseline and SelectConv]

- **Baseline** vs. **SelectConv**

16
Experiments: Improving Modern CNNs

- Selective convolution can be readily applied to various existing CNNs

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

- (top) Results on CIFAR-10/100, FMNIST and Tiny-ImageNet
- (right) Results on ImageNet dataset
Experiments: Improving Modern CNNs

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

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

- (top) Results on CIFAR-10/100, FMNIST and Tiny-ImageNet
- (right) Results on ImageNet dataset
Experiments: Mobile-targeted Architectures

- Selective convolution can further improve the “already-efficient” CondenseNet-182
- Training with \texttt{dealloc} → model compression

<table>
<thead>
<tr>
<th>Model</th>
<th>Params</th>
<th>FLOPs</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-1001</td>
<td>16.1M</td>
<td>2,357M</td>
<td>4.62</td>
</tr>
<tr>
<td>WideResNet-28-10</td>
<td>36.5M</td>
<td>5,248M</td>
<td>4.17</td>
</tr>
<tr>
<td>ResNeXt-29 (16 × 64d)</td>
<td>68.1M</td>
<td>10,704M</td>
<td>3.58</td>
</tr>
<tr>
<td>VGGNet-Slim [2]</td>
<td>2.30M</td>
<td>391M</td>
<td>6.20</td>
</tr>
<tr>
<td>ResNet-164-Slim [2]</td>
<td>1.10M</td>
<td>275M</td>
<td>5.27</td>
</tr>
<tr>
<td>CondenseNet-SConv-182</td>
<td>3.24M</td>
<td>422M</td>
<td>3.50</td>
</tr>
</tbody>
</table>

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 12\textsuperscript{th} 6:30 – 9:00 PM
@ Pacific Ballroom