CUP: Cluster Pruning for Compressing Deep Neural Networks

Rahul Duggal  
Amplitude  
Georgina Tech  

Cao Xiao  
Georgina Tech  

Richard Vuduc  
Georgina Tech  

Polo Chau  
Georgina Tech  

Jimeng Sun  
UIUC
Goal

Reduce the *storage* and *computation* cost of a DNN

\[
F(x; W) \approx F(x; W_{compressed})
\]

Such that \(|W_{compressed}| \ll |W|\)
Filter Pruning

Layer L

Activation Map Of layer L

Layer L+1

Activation Map Of layer L+1
Filter Pruning

Layer L

Prune 2 filters

Activation Map Of layer L

Layer L+1

Activation Map Of layer L+1

Prune 2 channels
Filter Pruning

Which filters to prune?
Our method

CUP: Cluster Pruning

Our Idea: Prune similar filters
Our method

CUP: Cluster Pruning

Our Idea: Prune similar filters
Our method

CUP: Cluster Pruning

STEP 1
Compute per filter features

INPUT
Layer /

Fully Connected Layer

Convolutional Layer
Our method

**CUP: Cluster Pruning**

INPUT
Layer $l$

Compute per filter features

STEP 1

**Fully Connected Layer**

\[ \tilde{W}_{i,:,:}^{(l)}, \tilde{B}_{i}^{(l)} \]

**Convolutional Layer**

\[ \tilde{W}_{i,:}^{(l+1)} \]

\[ \tilde{F}_{i,:}^{(l)} = [ \tilde{W}_{i,:,:}^{(l)}, \tilde{B}_{i}^{(l)} ], \tilde{W}_{i,:}^{(l+1)} \]

Incoming features  Outgoing features
Our method

**CUP: Cluster Pruning**

**STEP 1**

**Fully Connected Layer**

\[
\begin{align*}
\tilde{W}^{(l)}_{i,:}, & \quad \tilde{B}^{(l)}_{i,:}, \\
\tilde{F}^{(l)}_{i,:} & = \begin{bmatrix}
\tilde{W}_{i,:}^{(l)}, \\ \tilde{B}^{(l)}_{i,:}
\end{bmatrix}, \\
\tilde{W}_{i, :}^{(l)} & \text{Incoming features}, \\
\tilde{W}_{i}^{(l+1)} & \text{Outgoing features}
\end{align*}
\]

**Convolutional Layer**

\[
\begin{align*}
\tilde{W}^{(l)}_{i,:,:}, & \quad \tilde{B}^{(l)}_{i,:,:}, \\
\tilde{F}^{(l)}_{i,:,:} & = \begin{bmatrix}
g\left(\tilde{W}_{i,:,:}^{(l)}\right), \\ \tilde{B}^{(l)}_{i,:,:}
\end{bmatrix}, \\
\tilde{W}_{i,:,:}^{(l+1)} & \text{Input features}, \\
\tilde{W}_{i}^{(l+1)} & \text{Output features}
\end{align*}
\]
Our method

**CUP: Cluster Pruning**

1. Compute per filter features
2. Cluster filters in each layer

$\tilde{F}^{(l)}$
Our method

**CUP: Cluster Pruning**

INPUT

Layer / $l$

1. Compute per filter features

2. Cluster filters in each layer

- How many clusters?

$\tilde{F}^{(l)}$

$m$

$n + p + 1$
Our method

CUP: Cluster Pruning

How many clusters?
Our method

**CUP: Cluster Pruning**

INPUT

Layer / 

```
<table>
<thead>
<tr>
<th>Compute per filter features</th>
<th>Cluster filters in each layer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STEP 1</strong></td>
<td><strong>STEP 2</strong></td>
</tr>
</tbody>
</table>
```

How many clusters?

\[ \text{INPUT} \quad \text{Layer} / \]

\[ \mathcal{F}(l) \]

\[ m \]

\[ n + p + 1 \]

\[ \text{Cluster} \]

\[ \text{Global Threshold } t \]

\[ \mathcal{C} \]

\[ \mathcal{T} \]
Our method

**CUP: Cluster Pruning**

**INPUT**
Layer $l$

Compute per filter features

Cluster filters in each layer

**STEP 1**

**STEP 2**

$n + p + 1$

$m$

Cluster filters $F^{(l)}$

Clip Dendogram

Global Threshold $t$

Pruned layer $l$

Input

Output

How many clusters?
Our method

**CUP: Cluster Pruning**

![Diagram of CUP process]

1. **INPUT**
   - Layer $l$
   - $n + p + 1$

2. **STEP 1**
   - Compute per filter features

3. **STEP 2**
   - Cluster filters in each layer

4. **Cluster**
   - $\tilde{F}^{(l)}$
   - $m$

5. **Global Threshold $t$**

6. **Dendogram**
   - $C_{11}^{(l)}$
   - $C_{12}^{(l)}$
   - $C_{13}^{(l)}$

**How many clusters?**

$t$ parameterizes the number of clusters

---

16
Our method

**CUP: Cluster Pruning**

**INPUT**

Layer \( l \)

**STEP 1**

Compute per filter features

**STEP 2**

Cluster filters in each layer

**STEP 3**

Prune filters from clusters

**OUTPUT**

Pruned layer \( l \)

\[
S_r^{(l)} = \arg\max_{i \in \mathcal{C}_r^{(l)}} \| \widetilde{F}_i^{(l)} \|_2
\]
Our method

CUP: Cluster Pruning

Input:
Layer $l$

Compute per filter features
STEP 1

Cluster filters in each layer
STEP 2

Prune filters from clusters
STEP 3

Output:
Pruned layer $l$

Equation:

$S_r^{(l)} = \underset{i \in C_r^{(l)}}{\text{argmax}} \| \tilde{F}_{i,:} \|_2$

#clusters = # remaining filters

$t$ parameterizes pruning amount
Results

**Benefit 1:** Single hyper parameter control over pruning amount

**Flops Reduction** *(Higher is better)*

**Parameter Reduction** *(Higher is better)*

**Test Accuracy** *(Higher is better)*

1.45% drop
Result

Benefit 2: Non uniform pruning with a single hyper-parameter $t$
Results

Benefit 3: Training time reduction through train time pruning.

<table>
<thead>
<tr>
<th>Method</th>
<th>Retrain?</th>
<th>Top-1 (%)</th>
<th>FR (×)</th>
<th>Training Time (GPU Hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resnet-50</td>
<td>-</td>
<td>75.86</td>
<td>1.00</td>
<td>66.0</td>
</tr>
<tr>
<td>SFP [14]</td>
<td>x</td>
<td>74.01</td>
<td>1.73</td>
<td>61.8</td>
</tr>
<tr>
<td>GM [15]</td>
<td>x</td>
<td>74.13</td>
<td>2.15</td>
<td>62.2</td>
</tr>
<tr>
<td>CUP-RF (ours)</td>
<td>x</td>
<td>74.34</td>
<td>2.21</td>
<td>51.6</td>
</tr>
</tbody>
</table>

~15 hours saving with 2x compression
## Results

### Benefit 4: State-of-the-art compression

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>Retrain?</th>
<th>FR (×)</th>
<th>Acc. (Δ%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Top-1</td>
<td>Top-5</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>GM [15]</td>
<td>✓</td>
<td>1.71</td>
<td>-1.87 -1.15</td>
</tr>
<tr>
<td></td>
<td>COP [29]</td>
<td>✓</td>
<td>1.75</td>
<td>-2.48 -</td>
</tr>
<tr>
<td></td>
<td><strong>CUP (Our)</strong></td>
<td>✓</td>
<td><strong>1.75</strong></td>
<td><strong>-1.00 -0.79</strong></td>
</tr>
<tr>
<td></td>
<td>SFP [14]</td>
<td>×</td>
<td>1.71</td>
<td>-3.18 -1.85</td>
</tr>
<tr>
<td></td>
<td>GM [15]</td>
<td>×</td>
<td>1.71</td>
<td>-2.47 -1.52</td>
</tr>
<tr>
<td></td>
<td><strong>CUP-RF (ours)</strong></td>
<td>×</td>
<td><strong>1.75</strong></td>
<td><strong>-2.37 -1.40</strong></td>
</tr>
<tr>
<td>ResNet-34</td>
<td>L1 [2]</td>
<td>✓</td>
<td>1.31</td>
<td>-1.06 -</td>
</tr>
<tr>
<td></td>
<td>GM [15]</td>
<td>✓</td>
<td>1.69</td>
<td>-1.29 -0.54</td>
</tr>
<tr>
<td></td>
<td><strong>CUP (ours)</strong></td>
<td>✓</td>
<td><strong>1.78</strong></td>
<td><strong>-0.86 -0.53</strong></td>
</tr>
<tr>
<td></td>
<td>SFP [14]</td>
<td>×</td>
<td>1.69</td>
<td>-2.09 -1.29</td>
</tr>
<tr>
<td></td>
<td>GM [15]</td>
<td>×</td>
<td>1.69</td>
<td>-2.13 -0.92</td>
</tr>
<tr>
<td></td>
<td><strong>CUP-RF (ours)</strong></td>
<td>×</td>
<td><strong>1.71</strong></td>
<td><strong>-1.61 -0.89</strong></td>
</tr>
<tr>
<td></td>
<td>MP [30]</td>
<td>✓</td>
<td>2.05</td>
<td>-1.20 -</td>
</tr>
<tr>
<td></td>
<td><strong>CUP (ours)</strong></td>
<td>✓</td>
<td><strong>2.47</strong></td>
<td><strong>-1.17 -0.81</strong></td>
</tr>
<tr>
<td></td>
<td>SFP [14]</td>
<td>×</td>
<td>1.71</td>
<td>-1.54 -0.81</td>
</tr>
<tr>
<td></td>
<td>GM [15]</td>
<td>×</td>
<td>2.15</td>
<td>-2.02 -0.93</td>
</tr>
<tr>
<td></td>
<td><strong>CUP-RF (ours)</strong></td>
<td>×</td>
<td><strong>2.20</strong></td>
<td><strong>-1.47 -0.88</strong></td>
</tr>
</tbody>
</table>
Conclusion

CUP: Cluster pruning framework
• Prunes a DNN by clustering similar filters.

Benefits of CUP
• Single hyper-parameter control over pruning amount.
• Enables non uniform pruning across layers.
• Train time savings.

Extensive evaluation on large DNNs & datasets