SNIP: Single-shot Network Pruning based on Connection Sensitivity
Namhoon Lee, Thalaiyasingam Ajanthan, Philip H. S. Torr
University of Oxford

Network Pruning: Background and Motivation

- Issues with overparameterization in large neural networks
  - memory and time complexity
  - energy consumption
  - generalization capability

- Drawbacks of existing methods
  - hyperparameters with heuristics
  - architectural specific requirements
  - optimization difficulties
  - pretraining

What we want

- No hyperparameters
- No pretraining
- No iterative prune–retrain cycle
- Not require the whole training set

Single-shot pruning prior to training

Single-shot Network Pruning based on Connection Sensitivity

Neural Network Pruning

Write pruning as constrained optimization:

\[ \min_w \ell(w; D) = \min_w \frac{1}{n} \sum_{i=1}^{n} \ell(w; x_i, y_i), \]

subject to \( w \in \mathbb{R}^m, ||w|| \leq \kappa \).

Here, \( D = \{ (x_i, y_i) \}_{i=1}^{n} \) is a dataset, \( \kappa \) is a desired sparsity level, \( \ell(\cdot) \) is the loss function, \( w \) is the set of parameters of the network.

Connection Sensitivity: Architectural Perspective

Introduce auxiliary indicator variables \( e \in \{ 0, 1 \}^m \):

\[ \min_{c, w} \ell(c \circ w; D) = \min_{c, w} \frac{1}{n} \sum_{i=1}^{n} \ell(c \circ w; x_i, y_i), \]

subject to \( w \in \mathbb{R}^m, c \in \{ 0, 1 \}^m, ||c||_1 \leq \kappa \).

The idea is to measure the effect of removing a parameter on the loss by separating \( w \) from \( c \):

- The effect of removing parameter \( j \):
  \[ \Delta L_j(w; D) = L(1 \circ w; D) - L(1 \circ w_j; D), \]

where \( e_j \) is the indicator vector of element \( j \).

- Infinitesimal version of \( \Delta L_j \):
  \[ \Delta L_j(w; D) \approx \frac{\partial \ell(c \circ w; D)}{\partial c_j} \bigg|_{c_j=1} = \lim_{\delta \to 0} \frac{\ell(c \circ w; D) - \ell(c - \delta e_j \circ w; D)}{\delta}, \]

which, denoted as \( q_j(w; D) \), measures the rate of change of \( L \) with respect to an infinitesimal change in \( c_j \) from 1 to 1 – \( \delta \).

Define connection sensitivity as the saliency criterion by taking the normalized magnitude of \( q_j \):

\[ s_j = \frac{|q_j(w; D)|}{\sum_{k=1}^{m} |q_k(w; D)|} \]

Single-shot Pruning at Initialization

- Use variance scaling methods to initialize weights so that the impact of weights on \( s_j \) is minimized while making it robust to architectural variations.
- Using a reasonable number of training examples in one mini-batch can lead to effective pruning.

Algorithm 1 SNIP

Require: Loss function \( L \), training dataset \( D \), sparsity level \( \kappa \)
Ensure: \( ||w||_0 \leq \kappa \)
1. \( w \leftarrow \text{VarianceScalingInitialization} \)
2. \( D_b = \{ (x_i, y_i) \}_{i=1}^{n}, D \)
3. \( s_j \leftarrow \frac{|q_j(w; D)|}{\sum_{k=1}^{m} |q_k(w; D)|}, \quad \forall j \in \{1, \ldots, m \} \)
4. \( c_j \leftarrow \begin{cases} \{ j \mid s_j \geq \bar{\kappa} \}, & \forall j \in \{1, \ldots, m \} \end{cases} \)
5. \( w^* \leftarrow \arg \min_w \Delta L_j(c \circ w; D) \)
6. \( w^* \leftarrow c \circ w^* \)

SNIP is capable of pruning extreme sparsity levels (e.g., 90% for LeNet-5-Caffe), while being significantly simpler than other approaches.

Various modern architectures

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Model</th>
<th>Sparsity (%)</th>
<th># Parameters</th>
<th>Error (%)</th>
<th>( \Delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet-s</td>
<td>90.0</td>
<td>5.1m \to 600k</td>
<td>14.14 \to 14.99</td>
<td>0.87</td>
<td></td>
</tr>
<tr>
<td>AlexNet-s</td>
<td>90.0</td>
<td>8.5m \to 600k</td>
<td>14.12 \to 14.59</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>Conflational</td>
<td>VGG-C</td>
<td>95.0</td>
<td>8.62 \to 7.27</td>
<td>0.45</td>
<td></td>
</tr>
<tr>
<td>VGG-D</td>
<td>95.0</td>
<td>15.2m \to 76k</td>
<td>6.76 \to 6.70</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>VGG-Like</td>
<td>95.0</td>
<td>43.8m \to 8.38</td>
<td>6.38 \to 6.09</td>
<td>0.26</td>
<td></td>
</tr>
<tr>
<td>Residual</td>
<td>WRN-16-10</td>
<td>95.0</td>
<td>8.54 \to 6.43</td>
<td>0.52</td>
<td></td>
</tr>
<tr>
<td>WRN-22-10</td>
<td>95.0</td>
<td>7.6m \to 61k</td>
<td>6.45 \to 6.15</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>Recurrent</td>
<td>LSTM-a</td>
<td>95.0</td>
<td>10.0k \to 6.36</td>
<td>2.14 \to 1.32</td>
<td></td>
</tr>
<tr>
<td>LSTM-b</td>
<td>95.0</td>
<td>26.1k \to 1.15</td>
<td>2.35 \to 0.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRU-a</td>
<td>95.0</td>
<td>3.3k \to 2.14</td>
<td>4.04 \to 0.54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRU-b</td>
<td>95.0</td>
<td>20.3k \to 1.71</td>
<td>4.05 \to 0.13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SNIP is generally applicable to various architectures and models and reduces a significant amount of parameters with minimal loss in performance.

Visualizing pruned/retained parameters

(a) MNIST
(b) Fashion-MNIST

The parameters connected to the discriminative part of image survive and the irrelevant parts get pruned.

Survived parameters and resulting performance for different batch sizes

For \( |D| = 1 \), the sample was \( \kappa \); SNIP precisely retains valid connections. As \( |D| \) increases, connections get close to the train average, and the error decreases.

Fitting random labels

(left) The SNIP-pruned model does not fit the random labels. (right) The effect of varying sparsity \( \bar{\kappa} \). This indicates that the pruned network does not have sufficient capacity to fit the random labels, but is capable of performing the task.

Conclusion

SNIP: a new pruning algorithm that is simple, versatile and interpretable
- Pruning at single-shot prior to training
- Applicable to a variety of neural network models without modifications

https://github.com/namhoonlee/snip-public

{namhoon, ajanthan, phst}@robots.ox.ac.uk