Progressive Gradient Pruning for Classification, Detection and Domain Adaptation

ICPR 2020

Le Thanh Nguyen-Meidine, Eric Granger, Madhu Kiran, Marco Pedersoli, Louis-Antoine Blais-Morin
1) Introduction

Channel pruning methods can be seen as:

- **as a post processing step (L1, Entropy):** train a model -> prune -> fine-tune
- **as an iterative process on trained model (Taylor[2]):** train a model -> iteratively prune and fine-tune
- **as a reconstruction error (ThiNet, DCP):** train a model -> minimize the error between the pruned model and the original model
- **from scratch using constraints (DCP, Slimming):** add pruning constraints to original model -> optimize the model and the pruning constraints
- **as progressive pruning (PSFP, ours):** pruning while training the model from scratch without any constraints
1) Introduction

Current challenges with SOA methods:

- Existing methods for pruning from scratch is difficult to optimize
- Existing progressive pruning method does not take advantage of pruning during training
- It does not handle back-propagation pruning properly
1) Introduction

Contributions:

- A pruning technique and criterion suitable for training.
- Momentum pruning for backward pass
- The technique can be easily adapted to other tasks such as object detection or unsupervised domain adaptation
2) Progressive Gradient Pruning

Illustration of the overall working of our method for pruning during training.
2) Progressive Gradient Pruning

Forward Backward Pruning:

- Pruning of forward pass follows the same procedure as traditional pruning procedure

- Pruning of momentum tensor for the backward pass use the same indexes as forward pass
2) Progressive Gradient Pruning

Gradient Norm criterion:

- Inspired by Taylor(Molchanov)[2] equation:

\[
\left| \Delta \mathcal{L}(H_i) \right| = \left| \mathcal{L}(D|H_i = 0) - \mathcal{L}(D|H_i) \right| \approx \left| \frac{\partial \mathcal{L}}{\partial H_i} H_i \right|
\]

- With L any loss function, D a labeled dataset, H_i the feature map of layer i

- Experimentally, we found that, by changing the formulation to Weight (W_i) instead of feature map yield better result:

\[
TW = \left| \mathcal{L}(D|W_i = 0) - \mathcal{L}(D|W_i) \right| \approx \left| \frac{\partial \mathcal{L}}{\partial W_i} W_i \right|
\]
3) Experimental Results

Tasks And Datasets:
- Cifar, Mnist for classification
- PascalVOC for object detection
- Office31 for domain adaptation

Backbones:
- For classification:
  - LeNet and ResNet20 for MNIST
  - VGG19 and ResNet56 for CIFAR
- For object detection:
  - Faster R-CNN with VGG16 Backbone
- For domain adaptation:
  - MMD based domain adaptation with VGG16
3) Experimental Results

Baselines:
- For classification and object detection:
  - L1 pruning
  - Taylor
  - DCP
  - PSFP
- For domain adaptation:
  - TCP

Our Techniques:
- RPGP: Pruning is done at each training iteration
- PGP: Pruning is done at the end of each epoch
3) Experimental Results

Results of our algorithm (red) on MNIST (left) and CIFAR (right) on Resnet

- Our method can outperform baselines in some cases while having competitive results with state-of-the-art
3) Experimental Results

Results of our algorithm(red) for object detection with Faster R-CNN VGG16 on PascalVOC

<table>
<thead>
<tr>
<th>Methods</th>
<th>No. Params</th>
<th>FLOPS</th>
<th>mAP</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline VGG16</td>
<td>137M</td>
<td>250G</td>
<td>69.6%</td>
<td>428 min</td>
</tr>
<tr>
<td>LI [8]</td>
<td>125M</td>
<td>174G</td>
<td>62.3%</td>
<td>(428) + 31 min</td>
</tr>
<tr>
<td>PSFP [16]</td>
<td>125M</td>
<td>174G</td>
<td>63.5%</td>
<td>428 min</td>
</tr>
<tr>
<td>PGP_GN₇ (ours)</td>
<td>125M</td>
<td>174G</td>
<td>65.5%</td>
<td>769 min</td>
</tr>
<tr>
<td>RPGP_GNS (ours)</td>
<td>125M</td>
<td>174G</td>
<td>66.0%</td>
<td>281 min</td>
</tr>
</tbody>
</table>

- Our method outperforms baselines in both accuracy and training time
3) Experimental Results

Results of our algorithm (red) for domain adaptation with VGG16 on Office31

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Source-only</th>
<th>Baseline VGG</th>
<th>TCP</th>
<th>RPGP_GN₈ (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduction (% FLOPS)</td>
<td>0%</td>
<td>0%</td>
<td>26%</td>
<td>35%</td>
</tr>
<tr>
<td>A → W</td>
<td>68.5</td>
<td>74.0</td>
<td>76.1</td>
<td>78.2</td>
</tr>
<tr>
<td>A → D</td>
<td>61.1</td>
<td>72.3</td>
<td>76.2</td>
<td>77.7</td>
</tr>
<tr>
<td>W → A</td>
<td>41.6</td>
<td>55.2</td>
<td>51.2</td>
<td>51.6</td>
</tr>
<tr>
<td>W → D</td>
<td>94.3</td>
<td>97.5</td>
<td>99.8</td>
<td>99.4</td>
</tr>
<tr>
<td>D → W</td>
<td>94.5</td>
<td>94.0</td>
<td>96.1</td>
<td>96.5</td>
</tr>
<tr>
<td>D → A</td>
<td>50.3</td>
<td>54.1</td>
<td>47.9</td>
<td>48.0</td>
</tr>
<tr>
<td>Average</td>
<td>68.3</td>
<td>74.5</td>
<td>74.5</td>
<td>75.2</td>
</tr>
</tbody>
</table>

- Our method outperforms baselines both in compression and accuracy.
4) Conclusion

- A new progressive pruning method that’s suitable for pruning during training
- The proposed approach can work on classification as well as object detection and unsupervised domain adaptation
- It provides faster training and pruning time compared to state-of-the-art

Thank you for listening