Lost in Pruning:
The Effects of Pruning Neural Networks beyond Test Accuracy

Lucas Liebenwein, lucasl@mit.edu, http://www.mit.edu/~lucasl/
Distributed Robotics Lab, CSAIL, MIT
Neural networks are SOTA

Natural Language Processing

Computer Vision

Robotics
Larger size, better performance
Smaller size, same performance?
Smaller size, same performance?

- Less storage requirements
- Faster inference time
- Improved Interpretability
Objective: what are the effects of pruning?

1. How we prune?

2. What is preserved?

3. What is lost?

MLSys 2021: Lost in Pruning
Lucas Liebenwein, lucasl@mit.edu
Prune pipeline

Training → Prune weights or filters → Iteratively → Retrain (hyperparameters from training) → Desired Prune Ratio
Weight pruning

Filter pruning

$\ell - 1 \quad \ell \quad \ell + 1$
Weight pruning

Filter pruning

\( \ell - 1 \quad \ell \quad \ell + 1 \)

Remove individual edges
Weight pruning

Remove individual edges

Filter pruning

Remove neurons (all incoming and outgoing edges)
Weight pruning

Remove individual edges

Filter pruning

Remove neurons (all incoming and outgoing edges)
Prune methods

- Unstructured (Weights)
- Structured (Filters/Neurons)
# Prune methods

<table>
<thead>
<tr>
<th>Unstructured (Weights)</th>
<th>Relative importance within layer</th>
<th>Structured (Filters/Neurons)</th>
<th>Budget allocation across layers</th>
</tr>
</thead>
</table>
# Prune methods

<table>
<thead>
<tr>
<th>Unstructured (Weights)</th>
<th>Structural (Filters/Neurons)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WT: Weight Thresholding</strong> (Renda et al. 2020)</td>
<td>$\alpha</td>
</tr>
<tr>
<td><strong>SiPP: Sensitivity Pruning</strong> (Baykal et al. 2019)</td>
<td>$\alpha</td>
</tr>
</tbody>
</table>

Relative importance within layer  
Budget allocation across layers

---

$i$...filter,  
$j$...channel,  
$W_{i}^{\ell}$...filter weights,  
$W_{ij}^{\ell}$...channel weights,  
$x$...input,  
$a(x)$...activation (layer input)
## Prune methods

<table>
<thead>
<tr>
<th>Unstructured (Weights)</th>
<th>Structured (Filters/Neurons)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WT</strong>: Weight Thresholding (Renda et al. 2020)</td>
<td><strong>FT</strong>: Filter Thresholding (Renda et al. 2020)</td>
</tr>
<tr>
<td>$\propto</td>
<td>W_{ij}</td>
</tr>
<tr>
<td><strong>SiPP</strong>: Sensitivity Pruning (Baykal et al. 2019)</td>
<td><strong>PFP</strong>: Provable Filter Pruning (Lieberwein et al. 2020)</td>
</tr>
<tr>
<td>$\propto</td>
<td>W_{ij}a_j(x)</td>
</tr>
</tbody>
</table>

- $i, j, \ldots$ filter, channel
- $W^\ell_i, \ldots$ filter weights
- $W^\ell_j, \ldots$ channel weights
- $x, \ldots$ input
- $a(x), \ldots$ activation (layer input)
Prune results – how much can be pruned?

WT (Renda et al. 2020)
SiPP (Baykal et al. 2019)
FT (Renda et al. 2020)
PFP (Lieberwein et al. 2020)
Prune results – how much can be pruned?

ResNet20, CIFAR10

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>WT  (Renda et al. 2020)</td>
<td>84.9%</td>
</tr>
<tr>
<td>SiPP (Baykal et al. 2019)</td>
<td>84.9%</td>
</tr>
<tr>
<td>FT  (Renda et al. 2020)</td>
<td>52.1%</td>
</tr>
<tr>
<td>PFP (Lieberwein et al. 2020)</td>
<td>44.9%</td>
</tr>
</tbody>
</table>
Prune results – how much can be pruned?

<table>
<thead>
<tr>
<th></th>
<th>ResNet20, CIFAR10</th>
<th>ResNet101, ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>WT (Renda et al. 2020)</td>
<td>84.9%</td>
<td>81.6%</td>
</tr>
<tr>
<td>SiPP (Baykal et al. 2019)</td>
<td>84.9%</td>
<td>81.6%</td>
</tr>
<tr>
<td>FT (Renda et al. 2020)</td>
<td>52.1%</td>
<td>53.1%</td>
</tr>
<tr>
<td>PFP (Lieberwein et al. 2020)</td>
<td>44.9%</td>
<td>50.3%</td>
</tr>
</tbody>
</table>
## Prune results – how much can be pruned?

<table>
<thead>
<tr>
<th>Model</th>
<th>CIFAR10</th>
<th>ImageNet</th>
<th>Pascal VOC2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet20, CIFAR10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delta Test Accuracy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WT ([Renda et al. 2020])</td>
<td>84.9%</td>
<td>81.6%</td>
<td>58.9%</td>
</tr>
<tr>
<td>SiPP ([Baykal et al. 2019])</td>
<td>84.9%</td>
<td>81.6%</td>
<td>42.3%</td>
</tr>
<tr>
<td>FT ([Renda et al. 2020])</td>
<td>52.1%</td>
<td>53.1%</td>
<td>0.0%</td>
</tr>
<tr>
<td>PFP ([Liebenwein et al. 2020])</td>
<td>44.9%</td>
<td>50.3%</td>
<td>20.2%</td>
</tr>
<tr>
<td>DeeplabV3, Pascal VOC2011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>deelabv3_resnet50, VOCSegmentation2011</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is preserved during pruning?

- Test accuracy
- Informative features
- Functional similarities
What is preserved during pruning?

- Test accuracy
- Informative features
- Functional similarities

MLSys 2021: Lost in Pruning
Lucas Liebenwein, lucasl@mit.edu
Informative features

For a given network $f_\theta$ and data distribution $x \sim D$ the 10% of most informative features are defined as

$$m(\theta, D) := "the mask of input features that change the output of the network the least"$$

Captures the parts of the input that drive the network’s decision

Can be compared between pruned and unpruned networks
Functional (noise) similarity

For a given network $f_\theta$ and data distribution $x \sim \mathcal{D}$ noise similarity measures

"how much the output of the pruned and unpruned network differ under noisy input"

Captures Local Lipschitz behavior of networks

Direct comparison between pruned and unpruned networks
What is preserved during pruning?

- Test accuracy
- Informative features
- Functional similarities

MLSys 2021: Lost in Pruning

Lucas Liebenwein, lucasl@mit.edu
What is preserved during pruning?

- Test accuracy
- Informative features
- Functional similarities
What is **lost** during pruning?

- Prune potential for noise
- Prune potential for corruptions
- Adequate excess error
Prune potential

For $\delta \in [0,1)$, given network $f_\theta$, and data distribution $\mathcal{D}$ the prune potential $P(\theta, \mathcal{D})$ is defined as

$$P(\theta, \mathcal{D}) := "\text{maximum prune ratio with accuracy loss at most } \delta"$$

- Quantifies “overparameterization” of network
- Approximated using many prune-retrain cycles

Prune potential for $\delta = 0$
Prune potential for noise

- Prune potential after small noise injections in input
- Pruned networks are more affected by out-of-distribution
- "Robust overparameterization" vs. "nominal overparameterization"
Prune potential for corruptions

- Significantly reduced prune potential under distribution changes (!)
Excess error

For a network $f_{\theta}$, train distribution $\mathcal{D}$, and test distribution $\mathcal{D}'$ the excess error is defined as

$$e(\theta, \mathcal{D}') := \text{"additional error incurred on test distribution"}$$

- Quantifies performance drop for distribution changes
- Difference in excess error for pruned and unpruned network indicates performance drop
Excess error averaged over multiple corruptions

Pruned networks exhibit higher excess error

Nominal prune curve is not indicative of out-of-distribution performance!
What is *lost* during pruning?

- Prune potential for noise
- Prune potential for corruptions
- Adequate excess error
What is *lost* during pruning?

- **Prune potential for noise**: 
  
- **Prune potential for corruptions**: 
  
- **Adequate excess error**: 
  
---

MLSys 2021: Lost in Pruning

Lucas Liebenwein, lucasl@mit.edu
What is *lost* during pruning?

- Prune potential for noise
- Prune potential for corruptions
- Adequate excess error
What is *lost* during pruning?

- Prune potential for noise
- Prune potential for corruptions
- Adequate excess error
Related work and discussion

Nominal vs. robust overparameterization

- Neyshabur et al. 2019

Implicit regularization via overparameterization

- Belkin et al. 2019

Generalization-aware pruning

- Arora et al. 2018

Ye et al. 2019

Nakkiran et al. 2020

Sehwag et al. 2020
Guidelines for pruning in practice

1. Don’t prune if unexpected shifts in data may occur
2. Prune moderately if you can account for some data shifts during training
3. Prune fully if you can account for all data shifts during training
4. Maximize prune potential by considering data shifts during training
Lost in Pruning: The Effects of Pruning Neural Networks beyond Test Accuracy

Lucas Liebenwein, lucasl@mit.edu, http://www.mit.edu/~lucasl/
Distributed Robotics Lab, CSAIL, MIT

Code: https://github.com/lucaslie/torchprune
Contact: lucasl@mit.edu

Collaborators (Thank you!)
Cenk Baykal
Brandon Carter
David Gifford
Daniela Rus