ClickTrain: Efficient and Accurate End-to-End Deep Learning Training via Fine-Grained Architecture-Preserving Pruning

Chengming Zhang  Washington State University
Geng Yuan  Northeastern University
Wei Niu  College of William and Mary
Jiannan Tian  Washington State University
Sian Jin  Washington State University
Donglin Zhuang  University of Sydney
Zhe Jiang  University of Alabama
Yanzhi Wang  Northeastern University
Bin Ren  College of William and Mary
Shuaiwen Leon Song  University of Sydney
Dingwen Tao  Washington State University

June 14, 2021  ICS’21 Worldwide online event
Outline

- Introduction
  - prune neural networks during training
  - Fined-Grained Pattern-Based Pruning
  - Contribution

- Background
  - Patterns
  - Impact of patterns

- Designs
  - Modeling framework
  - Algorithm-level design
  - System-level design

- Experimental Evaluation
  - Model Accuracy and Ratio Evaluation
  - Single-GPU Performance Evaluation
  - Multi-GPU Performance Evaluation
Outline

- Introduction
  - prune neural networks during training
  - Fined-Grained Pattern-Based Pruning
  - Contribution

- Background
  - Patterns
  - Impact of patterns

- Designs
  - Modeling framework
  - Algorithm-level design
  - System-level design

- Experimental Evaluation
  - Model Accuracy and Ratio Evaluation
  - Single-GPU Performance Evaluation
  - Multi-GPU Performance Evaluation
1 Introduction

- **What is neural network pruning?**
  - Pruning is to reduce the number of DNN weights.
  - Pruning reduces the computation complexity.

- **Why prune neural networks during training?**
  - Ever-increasing scale and complexity of the networks with large-scale training datasets, leading to challenges to the cost of DNN training.
  - Backward phase can consume more than 70% of the overall training FLOPs.

![Figure 1: Percentage of FLOPs in forward and backward.](image-url)

Figure 1: Percentage of FLOPs in forward and backward.
1 Introduction

- **Typical Training Procedure**
  - Training a model to high accuracy.
  - Pruning the well-trained model.
  - Fine-tuning the pruned model.

- **Non-Structured Pruning and Structured Pruning**
  - The non-structured pruning: heuristically prune the redundant weights on arbitrary locations.
  - Structured Pruning: prune the entire filters, channels to maintain the structural regularity.

- **Fined-Grained Pattern-Based Pruning**
  - As shown in Fig 2, fined-grained pattern-based Pruning: intermediate sparsity type between non-structured pruning and structured pruning.

Fig 2. Fined grained pattern-based pruning. Gray parts are pruned.
1 Introduction

➢ What we did?

• Incorporate a weight importance estimation approach to select the desired patterns from a generated candidate pattern pool.
• Propose methods to gradually generate the candidate patterns.
• Propose a modified group-lasso regularization.
• Propose multiple system-level optimizations including fast sparse matrix format conversion, pattern-accelerated sparse convolution, pattern-based communication optimization, and compiler-assisted optimized code generation.
• Use pruning during training (PDT)-based method to significantly reduce the end-to-end time.
• Maintain the network architecture for high accuracy.
Introduction
- prune neural networks during training
- Finer-Grained Pattern-Based Pruning
- Contribution

Background
- Patterns
- Impact of patterns

Designs
- Modeling framework
- Algorithm-level design
- System-level design

Experimental Evaluation
- Model Accuracy and Ratio Evaluation
- Single-GPU Performance Evaluation
- Multi-GPU Performance Evaluation
2 Background

What is pattern?
- As shown in Figure 3, weights with higher absolute values form some specific shapes (named pattern).
- Repeatedly appears in the model.

![Fig 3. Heat map of convolutional layer of a VGG-16 [1].](image)

What is the impact of pattern on performance?
- Transform patterns to Gaussian filter

\[
\begin{bmatrix}
1 & 1 & 0 \\
1 & 1 & 0 \\
0 & 0 & 0 \\
\end{bmatrix} \ldots \begin{bmatrix}
0 & 1 & 1 \\
0 & 1 & 1 \\
0 & 0 & 0 \\
\end{bmatrix} \ldots \begin{bmatrix}
1 & 1 & 0 \\
1 & 1 & 0 \\
0 & 0 & 0 \\
\end{bmatrix} \ldots \begin{bmatrix}
0 & 0 & 1 \\
0 & 0 & 1 \\
0 & 1 & 1 \\
\end{bmatrix} = \left[ \begin{array}{ccc}
p & 2p & p^n \\
p & 2p & 4p & 2p \\
p & 2p & 4p & 2p \\
p & 2p & p \\
\end{array} \right] = \left[ \begin{array}{ccc}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1 \\
\end{array} \right] \]

![Fig 4. Gaussian filter [1].](image)
2 Background

- Transform patterns to Laplacian of Gaussian filter

\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & 1 & 1 \\
0 & 0 & 0
\end{bmatrix} \ldots \begin{bmatrix}
0 & 1 & 0 \\
1 & 1 & 0 \\
0 & 1 & 0
\end{bmatrix} \ldots \begin{bmatrix}
0 & 0 & 0 \\
1 & 1 & 1 \\
0 & 0 & 0
\end{bmatrix} \ldots \begin{bmatrix}
0 & 1 & 0 \\
1 & 1 & 1 \\
0 & 0 & 0
\end{bmatrix}
\]

\(=\left[\begin{array}{ccc}
0 & p & 0 \\
p & 1 & p \\
0 & p & 0
\end{array}\right]^n = \left[\begin{array}{ccc}
0 & 1 & 0 \\
p & 1/p & 1 \\
0 & 1 & 0
\end{array}\right]^n\)

Fig 5. Laplacian of Gaussian filter[1].

Outline

- Introduction
  - prune neural networks during training
  - Fined-Grained Pattern-Based Pruning
  - Contribution

- Background
  - Patterns
  - Impact of patterns

- Designs
  - Modeling framework
  - Algorithm-level design
  - System-level design

- Experimental Evaluation
  - Model Accuracy and Ratio Evaluation
  - Single-GPU Performance Evaluation
  - Multi-GPU Performance Evaluation
3 Framework

- As shown is fig 6. Stage 1, 2, 3 and 4 are algorithm-level design, which focus on high compress ratio and high accuracy.
- Stage 5 is system-level supports, which focus on improve computation efficiency.

Fig 6. Overview of framework.

- Introduced Hyper-parameters:
  Basic training hyperparameters (learning rate, etc.), compression ratio, regularized training epochs, regularization penalty coefficient, pre-training/warm-up epochs, and hard pruning epochs.
4 Algorithm-level design

- **Stage 1**: Train network for N epochs.
  - N is hyper-parameter.

- **Stage 2**: Dynamic pattern pool generate.
  - Generic pattern pool and dynamic pattern pool.
  - First select one weight position.
  - Select the second weight position.
  - Create a candidate pattern pool.
  - Calculate important score for each pattern and finalize patterns.

- **Stage 3**: Adaptive choose pattern for each kernel.
  - Calculate important score for each pattern using importance formula.
    \[ t_{i,:} = G_{f_{\ell}, c_{\ell}, :,:} \odot W_{f_{\ell}, c_{\ell}, :,:} \odot p_i, \quad I_{p_i} = \sum_{h_{\ell}} \sum_{s_{\ell}} (t_{h_{\ell}} \times s_{\ell})^2, \]

- **Stage 4**: Penalize unimportant weights using modified group lasso.
  \[ Z^{(\ell)} = W^{(\ell)} \odot (-P^{(\ell)}), \quad U^{(\ell)} = W^{(\ell)} \odot (-I^{(\ell)}) \]
  \[ E(W, D) = E(W, D) + \lambda P \sum_{l=1}^{L} \left( \sum_{f_{\ell}=1}^{F_{\ell}} \sum_{k_{\ell}=1}^{K_{\ell}} \left\| Z_{f_{\ell}, k_{\ell}; :,:}^{(\ell)} \right\|_g \right) \]
5 System-level design

- Modern GPUs are more suitable for Matrix-Matrix multiplication.
- SpMM requires first converting dense input matrix to a sparse format such as Compressed Sparse Row (CSR).
- Pattern sparsity facilitate the fast conversion.
- Limit all the filters in the same layer to have the same number of un-pruned (non-zero) weights.
- 1D tiling strategy and map each thread block to a 1D row tile of the output matrix.
Outline

- Introduction
  - prune neural networks during training
  - Fine-grained Pattern-Based Pruning
  - Contribution

- Background
  - Patterns
  - Impact of patterns

- Designs
  - Modeling framework
  - Algorithm-level design
  - System-level design

- Experimental Evaluation
  - Model Accuracy and Ratio Evaluation
  - Single-GPU Performance Evaluation
  - Multi-GPU Performance Evaluation
6 Results

- Setup and Dataset
  - Neural networks:
    ResNet18/32/50/101 and VGG11/13/16
  - Dataset:
    CIFAR10/100 [6] and ImageNet-2012
  - Experiment platform
    Pytorch
    Frontera supercomputer at TACC
    CUDA 10.1 and its default profiler
6 Results

<table>
<thead>
<tr>
<th>Method</th>
<th>PDT</th>
<th>Base. Acc.</th>
<th>Valid. Acc. Δ</th>
<th>Comp. Ratio</th>
<th>Train./Inf. FLOPs</th>
<th>Hard Pr. Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet32</td>
<td>PRT</td>
<td>93.6%</td>
<td>-2.1%</td>
<td>2.2X</td>
<td>53% / 66%</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>93.6%</td>
<td>0±0.05%</td>
<td>8.6X</td>
<td>41.3% / 85.1%</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>93.6%</td>
<td>0±0.07%</td>
<td>10.7X</td>
<td>43.9% / 85.7%</td>
<td>95</td>
</tr>
<tr>
<td>ResNet50</td>
<td>PRT</td>
<td>94.2%</td>
<td>-1.1%</td>
<td>2.3X</td>
<td>50% / 70%</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>94.1%</td>
<td>0±0.04%</td>
<td>8.5X</td>
<td>37.5% / 74.3%</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>94.1%</td>
<td>-0.2±0.05%</td>
<td>10.8X</td>
<td>41.2% / 77.6%</td>
<td>90</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>VGG11</td>
<td>PRT</td>
<td>92.1%</td>
<td>-0.7%</td>
<td>8.1X</td>
<td>57% / 65%</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>92.1%</td>
<td>-0.1±0.04%</td>
<td>8.7X</td>
<td>41.2% / 81.5%</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>92.1%</td>
<td>-0.3±0.06%</td>
<td>11.5X</td>
<td>43.9% / 85.3%</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>VGG13</td>
<td>PRT</td>
<td>93.9%</td>
<td>-0.6%</td>
<td>8.0X</td>
<td>56% / 63%</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>93.8%</td>
<td>0±0.08%</td>
<td>8.6X</td>
<td>41.3% / 81.3%</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>93.8%</td>
<td>-0.2±0.04%</td>
<td>10.9X</td>
<td>42.5% / 84.9%</td>
<td>96</td>
</tr>
<tr>
<td>ResNet32</td>
<td>PRT</td>
<td>71.0%</td>
<td>-1.4%</td>
<td>2.1X</td>
<td>32% / 46%</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>71.0%</td>
<td>0±0.05%</td>
<td>8.3X</td>
<td>41.7% / 82.9%</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>71.0%</td>
<td>-0.2±0.05%</td>
<td>10.4X</td>
<td>45.2% / 85.6%</td>
<td>90</td>
</tr>
<tr>
<td>CIFAR100</td>
<td>ResNet50</td>
<td>PRT</td>
<td>73.1%</td>
<td>-0.7%</td>
<td>1.9X</td>
<td>53% / 69%</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>73.1%</td>
<td>0±0.04%</td>
<td>8.2X</td>
<td>36.7% / 73.6%</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>73.1%</td>
<td>-0.2±0.07%</td>
<td>9.7X</td>
<td>38.9% / 77.3%</td>
<td>95</td>
</tr>
<tr>
<td>VGG11</td>
<td>PRT</td>
<td>70.6%</td>
<td>-1.3%</td>
<td>3.0X</td>
<td>47% / 57%</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>70.6%</td>
<td>0±0.1%</td>
<td>6.7X</td>
<td>40.1% / 78.6%</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>70.6%</td>
<td>-0.2±0.06%</td>
<td>8.4X</td>
<td>43.1% / 82.0%</td>
<td>92</td>
</tr>
<tr>
<td>VGG13</td>
<td>PRT</td>
<td>74.1%</td>
<td>-1.4%</td>
<td>2.9X</td>
<td>42% / 52%</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>74.1%</td>
<td>-0.1±0.05%</td>
<td>7.4X</td>
<td>40.5% / 79.7%</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>CLK</td>
<td>74.1%</td>
<td>-0.2±0.08%</td>
<td>9.2X</td>
<td>41.7% / 83.3%</td>
<td>96</td>
</tr>
</tbody>
</table>

Fig 11. Comparison between ClickTrain (CLK) and PDT-based method PruneTrain (PRT). FLOPs are the saved FLOPs.

Fig 12. Comparison between ClickTrain and PAT-based methods on ImageNet. Well-train costs about 90 epochs.
6 Results

Fig 9. CSR format conversion time.

Fig 10. Convolution time with different methods.
6 Results

Fig 13. Average forward and backward time per iteration.

Fig 14. Total training time on CIFAR and ImageNet (single-GPU).
Fig 15. Total time of distributed training.

Fig 16. Total time of communication time.
6 Results

Fig 17. Comparison of PruneTrain and ClickTrain.
7 Conclusion & Future Work

- **Conclusion**
  
  - Implement both algorithm-level and system-level optimizations with four stages.  
    i) **accurate** weight importance estimation to select the pattern,  
    ii) **dynamic** pattern generation and finalization,  
    iii) regularized training for fine-tuning with an enhanced group-lasso,  
    iv) **compiler-assisted** optimized training.  
  - Reduce the cost of PAT-based method by up to $2.3 \times$ with comparable accuracy and compression ratio.  
  - Improve the pruned accuracy by up to 1.8% and the compression ratio by up to $4.9 \times$ on the tested CNNs and datasets.  
  - We plan to extend ClickTrain to more types of DNNs in the future
Thank you!

Any questions are welcome!

Contact  Dingwen Tao: dingwen.tao@wsu.edu
          Chengming Zhang: chengming.zhang@wsu.edu