SplitNet: Learning to Semantically Split Deep Networks for Parameter Reduction and Model Parallelization

Presenter: Shijia Wang

Juyong Kim*1 Yookoon Park*1 Gunhee Kim1 Sung Ju Hwang2,3

*Equal contribution 1Seoul National University, Seoul, South Korea, 2UNIST, Ulsan, South Korea, 3AITrics, Seoul, South Korea.

PMLR 70, 2017
1 Introduction
   - Network Splitting
   - Related Works

2 SplitNet
   - Problem Statement
   - Learning to Split Network Weights into Disjoint Groups
   - Splitting Deep Neural Networks
   - Parallelization of SplitNet

3 Conclusion
   - Experiments
1 Introduction
   - Network Splitting
   - Related Works

2 SplitNet
   - Problem Statement
   - Learning to Split Network Weights into Disjoint Groups
   - Splitting Deep Neural Networks
   - Parallelization of SplitNet

3 Conclusion
   - Experiments
Introduction

- Reduces number of parameters and effectively structured for parallelization
- Observes classes do not share same features
- Learns to split a network into a set or a hierarchy of multiple groups that use disjoint sets of features
- Learns class-to-group and feature-to-group assignment matrices
Contributions

- Tree of disjoint subnetworks with reduced number of parameters
- Algorithm for training the network which learns assignment matrices and the weights
- Model-parallelizable with more processors
Figure 1. Concept. Our network automatically learns to split the classes and associated features into multiple groups at multiple network layers, obtaining a tree-structured network. Given a base network in (a), (b) our algorithm optimizes network weights as well as class-to-group and feature-to-group assignments. Colors indicate the group assignments of classes and features. (c) After learning to split the network, the model can be distributed to multiple GPUs to accelerate training and inference.
Outline

1 Introduction
   - Network Splitting
   - Related Works

2 SplitNet
   - Problem Statement
   - Learning to Split Network Weights into Disjoint Groups
   - Splitting Deep Neural Networks
   - Parallelization of SplitNet

3 Conclusion
   - Experiments
Related Works

- Parameter reduction for deep neural networks - $l_1$-norm, $l_{2,1}$-norm assumes classes share the same set of features
- Parallel and distributed deep learning - data parallelism and model parallelism assumes model structure is given and fixed
- Tree-structured deep networks - hierarchical class structures improves model accuracy but increased computational complexity
Outline

1 Introduction
   - Network Splitting
   - Related Works

2 SplitNet
   - Problem Statement
     - Learning to Split Network Weights into Disjoint Groups
     - Splitting Deep Neural Networks
     - Parallelization of SplitNet

3 Conclusion
   - Experiments
Network Parameters

- Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$ where $x_i \in \mathbb{R}^d$ is an input and $y_i \in \{1, \ldots, K\}$ is a label for $K$ classes.
- Goal is to learn weights at each layer $l$ is a block-diagonal matrix $\mathbf{W}^{(l)}$.
- Each block $\mathbf{W}_g^{(l)}$ is associated with a class group $g \in \mathcal{G}$ where $\mathcal{G}$ is the set of all groups.
Group Assignment Vectors

- Number of groups $G$ is given. $\mathbb{Z}_2 = \{0, 1\}$
- $p_{gi}$ binary variable whether feature $i$ is assigned to group $g$
- $q_{gj}$ binary variable indicates whether class $j$ is assigned to group $g$
- $p_g \in \mathbb{Z}_2^D$ feature group assignment vector for group $g$. $D$ is dimension of features
- $q_g \in \mathbb{Z}_2^K$ class group assignment vector for group $g$. $K$ is number of classes
- $p_g$ and $q_g$ define group $g$
Restrictions

- No overlaps between groups in features or classes
- $\sum_g p_g = 1_D$ and $\sum_g q_g = 1_K$
- Allows weight matrix $\mathbf{W} \in \mathbb{R}^{D \times L}$ to be sorted into a block-diagonal matrix. Fewer parameters and faster multiplication.
Objective

\[
\min_{\omega, \mathbf{P}, \mathbf{Q}} \mathcal{L}(\omega, \mathbf{X}, \mathbf{y}) + \sum_{l=1}^{L} \lambda \| \mathbf{W}^{(l)} \|_2^2 + \sum_{l=S}^{L} \Omega(\mathbf{W}^{(l)}, \mathbf{P}^{(l)}, \mathbf{Q}^{(l)})
\]

- \(\mathcal{L}(\omega, \mathbf{X}, \mathbf{y})\) is cross entropy loss on training data
- \(\omega = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}\}\) is the set of network weights at all layers
- \(\| \mathbf{W}^{(l)} \|_2^2\) is the weight decay regularizer with hyperparameter \(\lambda\)
- \(S\) is the layer where splitting starts
- \(\Omega(\mathbf{W}^{(l)}, \mathbf{P}^{(l)}, \mathbf{Q}^{(l)})\) is the regularizer for splitting the network
- \(\mathbf{P}^{(l)}, \mathbf{Q}^{(l)}\) are the set of feature-to-group and class-to-group assignment vectors respectively for each layer \(l\)
Objective

- Jointly optimized using stochastic gradient descent
- Obtains grouping and prunes out inter-group connections
- Once grouping is learned, the weight matrix can be split into block-diagonal matrices
Outline

1 Introduction
   - Network Splitting
   - Related Works

2 SplitNet
   - Problem Statement
   - Learning to Split Network Weights into Disjoint Groups
   - Splitting Deep Neural Networks
   - Parallelization of SplitNet

3 Conclusion
   - Experiments
Three Objectives

\[
\Omega(W, P, Q) = \gamma_1 R_w(W, P, Q) + \gamma_2 R_D(P, Q) + \gamma_3 R_E(P, Q) \tag{2}
\]

- All are regularizations
Group Weight Regularization

\[ p_{gi} = \frac{e^{\alpha_{gi}}}{\sum_g e^{\alpha_{gi}}}, \quad q_{gj} = \frac{e^{\beta_{gi}}}{\sum_g e^{\beta_{gi}}} \]  

- Reparameterize with unconstrained variables \( \alpha_{gi} \) and \( \beta_{gj} \) in the \textit{softmax} form
- Empirically observed this form results in more semantically meaningful groupings.
Group Weight Regularization

\[ R_w(W, P, Q) = \sum_{g} \sum_{i} \|((I - P_g)WQ_g)_i\|_2 + \sum_{g} \sum_{j} \|(P_g W(I - Q_g))_j\|_2 \]  \hspace{1cm} (4)

- \( P_g = \text{diag}(p_g) \) and \( Q_g = \text{diag}(q_g) \) are the feature and class group assignments for group \( g \)
- \((M)_i\) and \((M)_j\) denote \( i\)-th row and \( j\)-th column of matrix \( M \)
- Row/column-wise \( l_{2,1}\)-norm
- \( P_g WQ_g \) represents the weights for group \( g \)
- Prune out inter-group connections to obtain block-diagonal matrices
Figure 2. **Group Assignment and Group Weight Regularization.** (Left) An example of group assignment with $G = 3$. Colors indicate groups. Each row of matrix $P, Q$ is group assignment vectors for group $g$: $p_g, q_g$. (Right) Visualization of matrix $(I - P_g)WQ_g$. The group assignment vectors work as soft indicators for inter-group connections. As the groupings converge, $\ell_{2,1}$-norm is concentrated on inter-group connections.
Disjoint Group Assignment

\[ R_D(P, Q) = \sum_{i < j} p_i \cdot p_j + \sum_{i < j} q_i \cdot q_j \] (5)

- To make the group assignment vectors to be completely mutually exclusive
- Orthogonal so their products equal to 0
Balanced Group Assignment

\[ R_E(P, Q) = \sum_g (\sum_i p_{gi})^2 + (\sum_j q_{gi})^2 \]  

- Balance the group assignments
- Minimized when each group has equal number of elements
Outline

1. Introduction
   - Network Splitting
   - Related Works

2. SplitNet
   - Problem Statement
   - Learning to Split Network Weights into Disjoint Groups
   - Splitting Deep Neural Networks
   - Parallelization of SplitNet

3. Conclusion
   - Experiments
Purpose

- Split the input and hidden layers
- Multiple consecutive layers or recursive hierarchical group assignments
Algorithm 1 Splitting Deep Neural Networks

Input: Number of groups $G$, layers to split $S \leq L$ and hyperparameters $\gamma_1, \gamma_2, \gamma_3$

Initialize weights and group assignments

while groupings have not converged do

  Optimize the objective using SGD with a learning rate $\eta$

  $$\mathcal{L}(\omega, X, Y) + \lambda \sum_{l=1}^{L} \|W^{(l)}\|_2^2 + \gamma_1 \sum_{l=S}^{L} R_W(W^{(l)}, P^{(l)}, Q^{(l)})$$

  $$+ \gamma_2 \sum_{l=S}^{L} R_D(P^{(l)}, Q^{(l)}) + \gamma_3 \sum_{l=S}^{L} R_E(P^{(l)}, Q^{(l)})$$

end while

Split the network using the obtained group assignments and weight matrices

while validation accuracy improves do

  Optimize $\mathcal{L}(\omega, X, Y) + \lambda \sum_{l=1}^{L} \|W^{(l)}\|_2^2$ using SGD

end while
Deep Split

- Lower layers learn generic representations
- Higher layers learn features more specific to the classes
- Only need to split the layers down to the $S$-th layer
Deep Split

- The output nodes of each layer corresponds to the input nodes of the next layer
- \( q_g^{(l)} = p_g^{(l+1)} \)
- Signal is not passed across different groups of layers
- Allows parallelization
- Same objective function
Hierarchical Grouping

- Multi-level hierarchy of categories
- \( p_g^{(l+1)} = \sum_s p_g^{(l+1)} \)
- \( s \) subgroups
Outline

1 Introduction
   - Network Splitting
   - Related Works

2 SplitNet
   - Problem Statement
   - Learning to Split Network Weights into Disjoint Groups
   - Splitting Deep Neural Networks
   - Parallelization of SplitNet

3 Conclusion
   - Experiments
Parallelization

- Two approaches
  - Assigning both the lower-layers and group-specific upper layers to each node
    - Redundant computation
  - Assigning the lower layer to a separate processor
    - Communication overhead
1 Introduction
   - Network Splitting
   - Related Works

2 SplitNet
   - Problem Statement
   - Learning to Split Network Weights into Disjoint Groups
   - Splitting Deep Neural Networks
   - Parallelization of SplitNet

3 Conclusion
   - Experiments
Datasets and Baselines

- **Datasets:** CIFAR-100 (45000 images and 100 classes) and ImageNet-1K (1.2M images and 1000 classes)
- **Base Network:** Wide Residual Network (WRN), AlexNet, and ResNet-18
- **Testing Networks:** SplitNet-Semantic (semantic taxonomy), SplitNet-Clustering (spectral clustering), SplitNet-Random, SplitNet
Table 1. Comparison of Test Errors According to Depths of Splitting (row) and Splitting Methods (column) on CIFAR-100.
Postfix S, C and R denote SplitNet variants – Semantic, Clustering and Random, respectively.

<table>
<thead>
<tr>
<th>METHOD</th>
<th>SPLIT DEPTH</th>
<th>G</th>
<th>SPLITNet-S</th>
<th>SPLITNet-C</th>
<th>SPLITNet-R</th>
<th>SPLITNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRN-16-8 (BASELINE)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24.28</td>
</tr>
<tr>
<td>WRN-16-8 (DROPOUT)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24.52</td>
</tr>
<tr>
<td>FC Split</td>
<td>1</td>
<td>4</td>
<td>23.80</td>
<td>23.72</td>
<td>24.30</td>
<td>24.26</td>
</tr>
<tr>
<td>Shallow Split</td>
<td>6</td>
<td>2</td>
<td>24.46</td>
<td>24.54</td>
<td>25.46</td>
<td>23.96</td>
</tr>
<tr>
<td>Deep Split (dropout)</td>
<td>11</td>
<td>2</td>
<td>25.04</td>
<td>26.04</td>
<td>27.12</td>
<td>24.62</td>
</tr>
<tr>
<td>Hier. Split (dropout)</td>
<td>11</td>
<td>2-4</td>
<td>24.92</td>
<td>25.98</td>
<td>26.78</td>
<td>24.80</td>
</tr>
</tbody>
</table>
### Table 2. Comparison of Parameter/Computation Reduction and Test Errors on CIFAR-100.

<table>
<thead>
<tr>
<th>Network</th>
<th>Params($10^6$)</th>
<th>% Reduced</th>
<th>FLOPS($10^9$)</th>
<th>% Reduced</th>
<th>Test Error(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRN-16-8 (BASELINE)</td>
<td>11.0</td>
<td>0.0</td>
<td>3.10</td>
<td>0.0</td>
<td>24.28</td>
</tr>
<tr>
<td>FC Split</td>
<td>11.0</td>
<td>0.35</td>
<td>3.10</td>
<td>0.0</td>
<td>24.26</td>
</tr>
<tr>
<td>SHALLOW Split</td>
<td>7.42</td>
<td>32.54</td>
<td>2.64</td>
<td>14.63</td>
<td><strong>23.96</strong></td>
</tr>
<tr>
<td>DEEP Split (DROPOUT)</td>
<td>5.90</td>
<td>46.39</td>
<td>2.11</td>
<td>31.97</td>
<td>24.66</td>
</tr>
<tr>
<td>HIER. Split (DROPOUT)</td>
<td>4.12</td>
<td>62.58</td>
<td>1.88</td>
<td>39.29</td>
<td>24.80</td>
</tr>
</tbody>
</table>
### Table 3. Comparison of Parameter/Computation Reduction and Test Errors of AlexNet variants on ILSVRC2012.
The number of splits indicates the split in fc6, fc7 and fc8 layer, respectively. In 2×5 split, we split from $conv_4$ to fc8 with $G = 2$.

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>SPLITS</th>
<th>PARMS($10^6$)</th>
<th>% REDUCED</th>
<th>FLOPS($10^9$)</th>
<th>% REDUCED</th>
<th>TEST ERROR(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet(Baseline)</td>
<td>0</td>
<td>62.37</td>
<td>0</td>
<td>2.278</td>
<td>0</td>
<td>41.72</td>
</tr>
<tr>
<td>SplitNet</td>
<td>1-1-3</td>
<td>59.64</td>
<td>4.38</td>
<td>2.273</td>
<td>0.21</td>
<td>42.07</td>
</tr>
<tr>
<td></td>
<td>1-2-5</td>
<td>50.69</td>
<td>18.72</td>
<td>2.256</td>
<td>1.00</td>
<td>42.21</td>
</tr>
<tr>
<td></td>
<td>2-4-8</td>
<td>27.34</td>
<td>56.17</td>
<td>2.209</td>
<td>3.05</td>
<td>43.02</td>
</tr>
<tr>
<td></td>
<td>2×5</td>
<td>31.96</td>
<td>48.76</td>
<td>1.847</td>
<td>18.95</td>
<td>44.60</td>
</tr>
<tr>
<td>SplitNet-R</td>
<td>1-1-3</td>
<td>59.64</td>
<td>4.38</td>
<td>2.273</td>
<td>0.21</td>
<td>42.20</td>
</tr>
<tr>
<td></td>
<td>1-2-5</td>
<td>50.70</td>
<td>18.70</td>
<td>2.256</td>
<td>0.99</td>
<td>43.20</td>
</tr>
<tr>
<td></td>
<td>2-4-8</td>
<td>27.34</td>
<td>56.17</td>
<td>2.209</td>
<td>3.05</td>
<td>43.35</td>
</tr>
<tr>
<td></td>
<td>2×5</td>
<td>31.94</td>
<td>48.79</td>
<td>1.846</td>
<td>18.93</td>
<td>44.99</td>
</tr>
</tbody>
</table>
Table 4. Comparison of Parameter/Computation Reduction and Test Errors of ResNet-18 variants (ResNet-18x2) on ILSVRC2012. The number of splits indicates the split in $conv4\text{-}1\&2$, $conv5\text{-}1\&2$ and the last fc layer, respectively.

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>SPLITS</th>
<th>SPLIT DEPTH</th>
<th>PARAMS($10^6$)</th>
<th>% REDUCED</th>
<th>FLOPS($10^9$)</th>
<th>% REDUCED</th>
<th>TEST ERROR(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18x2</td>
<td>0</td>
<td>0</td>
<td>45.67</td>
<td>0</td>
<td>14.04</td>
<td>0</td>
<td>25.58</td>
</tr>
<tr>
<td>SplitNet</td>
<td>1-1-3</td>
<td>1</td>
<td>44.99</td>
<td>1.49</td>
<td>14.04</td>
<td>0.01</td>
<td>24.90</td>
</tr>
<tr>
<td></td>
<td>1-2-2</td>
<td>6</td>
<td>28.39</td>
<td>37.84</td>
<td>12.39</td>
<td>11.72</td>
<td>25.48</td>
</tr>
<tr>
<td></td>
<td>2-2-2</td>
<td>11</td>
<td>24.21</td>
<td>47.00</td>
<td>10.75</td>
<td>23.42</td>
<td>26.45</td>
</tr>
<tr>
<td>SplitNet-R</td>
<td>1-1-3</td>
<td>1</td>
<td>44.99</td>
<td>1.49</td>
<td>14.03</td>
<td>0.01</td>
<td>25.86</td>
</tr>
<tr>
<td></td>
<td>1-2-2</td>
<td>6</td>
<td>28.38</td>
<td>37.86</td>
<td>12.39</td>
<td>11.72</td>
<td>26.41</td>
</tr>
<tr>
<td></td>
<td>2-2-2</td>
<td>11</td>
<td>24.14</td>
<td>47.14</td>
<td>10.75</td>
<td>23.46</td>
<td>28.61</td>
</tr>
</tbody>
</table>
Table 5. Model Parallelization Benchmark of SplitNet on Multiple GPUs. We measure evaluation time performance of our SplitNet over 50,000 CIFAR-100 images with batch size 100 on TITAN X Pascal. Baseline implements layer-wise parallelization where sequential blocks of layers are distributed on GPUs.

<table>
<thead>
<tr>
<th>NETWORK</th>
<th>GPUs</th>
<th>TIME(s)</th>
<th>SPEEDUP (×)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASELINE</td>
<td>1</td>
<td>24.27 ± 0.35</td>
<td>1.00</td>
</tr>
<tr>
<td>BASELINE-HORZ.</td>
<td>2</td>
<td>46.70 ± 0.48</td>
<td>0.52</td>
</tr>
<tr>
<td>BASELINE-VERT.</td>
<td>2</td>
<td>20.15 ± 0.67</td>
<td>1.20</td>
</tr>
<tr>
<td>BASELINE-VERT.</td>
<td>3</td>
<td>22.45 ± 0.77</td>
<td>1.08</td>
</tr>
<tr>
<td>SHALLOW SPLIT</td>
<td>2</td>
<td>17.78 ± 0.23</td>
<td>1.37</td>
</tr>
<tr>
<td>DEEP SPLIT</td>
<td>2</td>
<td>14.03 ± 0.13</td>
<td><strong>1.73</strong></td>
</tr>
<tr>
<td>HIER. SPLIT</td>
<td>2</td>
<td>14.22 ± 0.05</td>
<td>1.71</td>
</tr>
<tr>
<td>DEEP SPLIT 3-WAY</td>
<td>3</td>
<td>10.92 ± 0.44</td>
<td><strong>2.22</strong></td>
</tr>
</tbody>
</table>
Qualitative Analysis

Figure 3. Learned Groups and Block-diagonal Weight Matrices. Visualization of the weight matrices along with corresponding group assignments learned in AlexNet 2-4-8 split. Obtained block-diagonal weights are *split* for faster multiplication. Note the hierarchical grouping structure.
Figure 4. Effect of Balanced Group Regularization $R_E(P, Q)$. The above figures show group-to-class assignment matrix $Q^{(fc8)}$ for different values of $\gamma_3$ on ImageNet-1K with $G = 3$. 
Figure 5. **Learned Groups.** Visualization of grouping learned in FC SplitNet on CIFAR-100. Rows denote learned groups, while columns denote semantic groups provided by CIFAR-100 dataset, each of which includes 5 classes. The brightness of a cell shows the agreement between learned groups and semantic groups.
Summary

- Reduced number of parameters and make parallelization easier
- Algorithm that created block-diagonal matrices
- Allows weights and splitting to be trained at the same time
- For future work, plan to find way to efficiently train on multi-GPU or multiprocessor environments from the initial training stage