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

Presenter: Shijia Wang

Juyong Kim*[1] Yookoon Park*[1] Gunhee Kim[1] Sung Ju Hwang[2,3]

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

PMLR 70, 2017

Outline

Outline

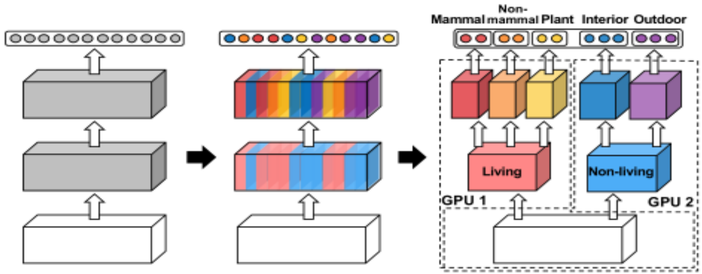
Introduction

- 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

Concept



(a) Base Network

(b) Splitting Deep Neural Network

(c) Our Network(*SplitNet*)

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

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

Network Parameters

- Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$ where $x_i \in R^d$ 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. $Z_2 = \{0, 1\}$
- 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
- $\mathbf{p}_g \in Z_2^D$ feature group assignment vector for group g. D is dimension of features
- $\mathbf{q}_g \in Z_2^K$ class group assignment vector for group g. K is number of classes
- $\mathbf{p}_g$ and $\mathbf{q}_g$ define group g

Restrictions

- No overlaps between groups in features or classes
- $\sum_g \mathbf{p}_g = 1_D$ and $\sum_g \mathbf{q}_g = 1_K$
- Allows weight matrix $\mathbf{W} \in R^{DxL}$ to be sorted into a block-diagonal matrix. Fewer parameters and faster multiplication.

Objective

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

- $\mathcal{L}(\omega, \mathbf{X}, \mathbf{y})$ 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 λ
- 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

Three Objectives

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

- All are regularizations

Group Weight Regularization

$$p_{gi} = e^{\alpha_{gi}} / \sum_{g} e^{\alpha_{gi}}, q_{gj} = e^{\beta_{gi}} / \sum_{g} e^{\beta_{gi}} \quad (3)$$

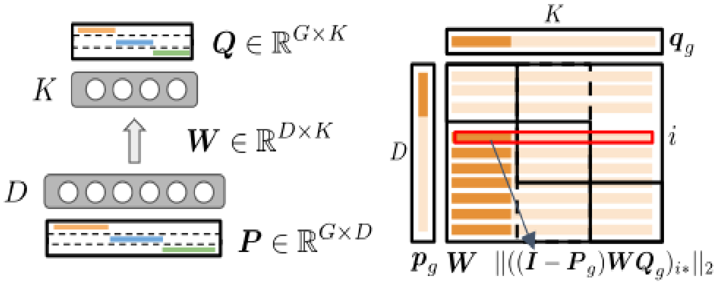
- Reparameterize with unconstrained variables α_{gi} and β_{gj} in the softmax form
- Empirically observed this form results in more semantically meaningful groupings.

Group Weight Regularization

$$R_w(\mathbf{W},\mathbf{P},\mathbf{Q}) = \sum_g \sum_i ||((\mathbf{I}-\mathbf{P}_g)\mathbf{W}\mathbf{Q}_g)_{i*}||_2 + \sum_g \sum_j ||(\mathbf{P}_g\mathbf{W}(\mathbf{I}-\mathbf{Q}_g))_{*j}||_2 \tag{4}$$

- $\mathbf{P}_g = diag(\mathbf{p}_g)$ and $\mathbf{Q}_g = diag(\mathbf{q}_g)$ 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
- $\mathbf{P}_g\mathbf{W}\mathbf{Q}_g$ represents the weights for group g
- Prune out inter-group connections to obtain block-diagonal matrices

Group Weight Regularizatio



$Q \in \mathbb{R}^{G\times K}$
K
$W \in \mathbb{R}^{D\times K}$
D
$P \in \mathbb{R}^{G\times D}$
K
q_g
D
i
p_g
W
$\|((I - P_g)WQ_g)_{i*}\|_2$

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

$$R_D(\mathbf{P}, \mathbf{Q}) = \sum_{i<j} \mathbf{p}_i \cdot \mathbf{p}_j + \sum_{i<j} \mathbf{q}_i \cdot \mathbf{q}_j \tag{5}$$

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

Balanced Group Assignment

$$R_E(\mathbf{P}, \mathbf{Q}) = \sum_g ((\sum_i p_{gi})^2 + (\sum_j q_{gi})^2) \tag{6}$$

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

Outline

Purpose

- Split the input and hidden layers
- Multiple consecutive layers or recursive hierarchical group assignments

Algorithm

Algorithm 1 Splitting Deep Neural Networks

```
Input: Number of groups $G$, layers to split $S \leq L$ and hyper-paramaters $\gamma_1, \gamma_2, \gamma_3$
Initialize weights and group assignments
while groupings have not converged do
    Optimize the objective using SGD with a learning rate $\eta$
    $\mathcal{L}(\omega, \boldsymbol{X}, \boldsymbol{Y}) + \lambda \sum_{l=1}^{L} ||\mathbf{W}^{(l)}||_2^2 + \gamma_1 \sum_{l=S}^{L} R_W(\boldsymbol{W}^{(l)}, \boldsymbol{P}^{(l)}, \boldsymbol{Q}^{(l)})$
    $+\gamma_2 \sum_{l=S}^{L} R_D(\boldsymbol{P}^{(l)}, \boldsymbol{Q}^{(l)}) + \gamma_3 \sum_{l=S}^{L} R_E(\boldsymbol{P}^{(l)}, \boldsymbol{Q}^{(l)})$
end while
Split the network using the obtained group assignments and weight matrices
while validation accuracy improves do
    Optimize $\mathcal{L}(\omega, \boldsymbol{X}, \boldsymbol{Y}) + \lambda \sum_{l=1}^{L} ||\mathbf{W}^{(l)}||_2^2$ 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
- $\mathbf{q}_g^{(l)} = \mathbf{p}_g^{(l+1)}$
- Signal is not passed across different groups of layers
- Allows parallization
- Same objective function

Hierarchical Grouping

- Multi-level hierarchy of categories
- $\mathbf{p}_g^{(l+1)} = \sum_s \mathbf{p}_{gs}^{(l+1)}$
- s subgroups

Outline

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

Outline

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

Parameter Reduction and Accuracies

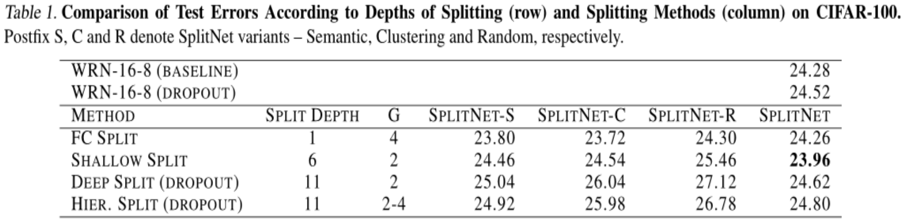
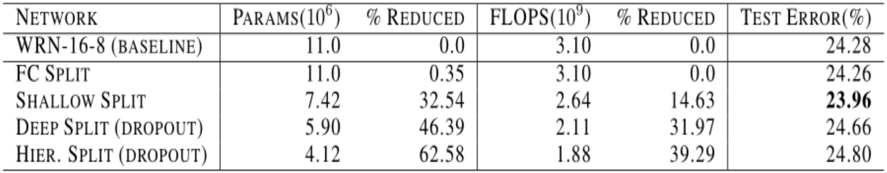


Table 1. **Comparison of Test Errors According to Depths of Splitting (row) and Splitting Methods (column) on CIFAR-100.** Postfix S, C and R denote SplitNet variants – Semantic, Clustering and Random, respectively.

WRN-16-8 (BASELINE)						24.28
WRN-16-8 (DROPOUT)						24.52
METHOD	SPLIT DEPTH	G	SPLITNET-S	SPLITNET-C	SPLITNET-R	SPLITNET
FC SPLIT	1	4	23.80	23.72	24.30	24.26
SHALLOW SPLIT	6	2	24.46	24.54	25.46	**23.96**
DEEP SPLIT (DROPOUT)	11	2	25.04	26.04	27.12	24.62
HIER. SPLIT (DROPOUT)	11	2-4	24.92	25.98	26.78	24.80

Parameter Reduction and Accuracies

Table 2. **Comparison of Parameter/Computation Reduction and Test Errors on CIFAR-100.**



NETWORK	PARAMS(10^6)	% REDUCED	FLOPS(10^9)	% REDUCED	TEST ERROR(%)
WRN-16-8 (BASELINE)	11.0	0.0	3.10	0.0	24.28
FC SPLIT	11.0	0.35	3.10	0.0	24.26
SHALLOW SPLIT	7.42	32.54	2.64	14.63	**23.96**
DEEP SPLIT (DROPOUT)	5.90	46.39	2.11	31.97	24.66
HIER. SPLIT (DROPOUT)	4.12	62.58	1.88	39.29	24.80

Parameter Reduction and Accuracies

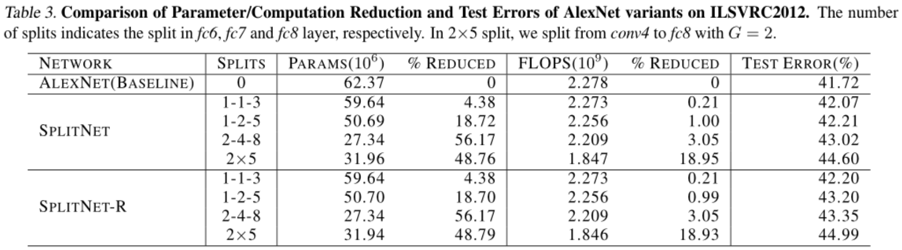


Table 3. **Comparison of Parameter/Computation Reduction and Test Errors of AlexNet variants on ILSVRC2012.** The number of splits indicates the split in *fc6*, *fc7* and *fc8* layer, respectively. In 2×5 split, we split from *conv4* to *fc8* with $G = 2$.

NETWORK	SPLITS	PARAMS(10^6)	% REDUCED	FLOPS(10^9)	% REDUCED	TEST ERROR(%)
ALEXNET(BASELINE)	0	62.37	0	2.278	0	41.72
SPLITNET	1-1-3	59.64	4.38	2.273	0.21	42.07
	1-2-5	50.69	18.72	2.256	1.00	42.21
	2-4-8	27.34	56.17	2.209	3.05	43.02
	2×5	31.96	48.76	1.847	18.95	44.60
SPLITNET-R	1-1-3	59.64	4.38	2.273	0.21	42.20
	1-2-5	50.70	18.70	2.256	0.99	43.20
	2-4-8	27.34	56.17	2.209	3.05	43.35
	2×5	31.94	48.79	1.846	18.93	44.99

Parameter Reduction and Accuracies

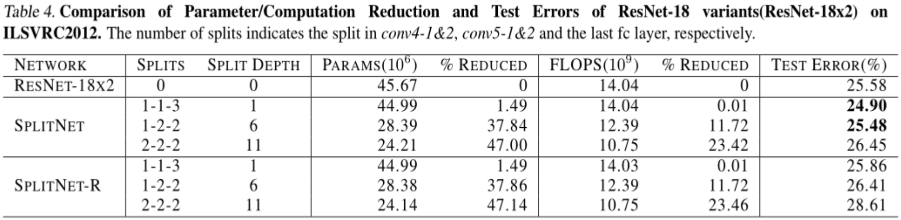


Table 4. **Comparison of Parameter/Computation Reduction and Test Errors of ResNet-18 variants(ResNet-18x2) on ILSVRC2012.** The number of splits indicates the split in *conv4-1&2*, *conv5-1&2* and the last fc layer, respectively.

NETWORK	SPLITS	SPLIT DEPTH	PARAMS(10^6)	% REDUCED	FLOPS(10^9)	% REDUCED	TEST ERROR(%)
RESNET-18X2	0	0	45.67	0	14.04	0	25.58
SPLITNET	1-1-3	1	44.99	1.49	14.04	0.01	**24.90**
	1-2-2	6	28.39	37.84	12.39	11.72	**25.48**
	2-2-2	11	24.21	47.00	10.75	23.42	26.45
SPLITNET-R	1-1-3	1	44.99	1.49	14.03	0.01	25.86
	1-2-2	6	28.38	37.86	12.39	11.72	26.41
	2-2-2	11	24.14	47.14	10.75	23.46	28.61

Test-time Model Parallelization

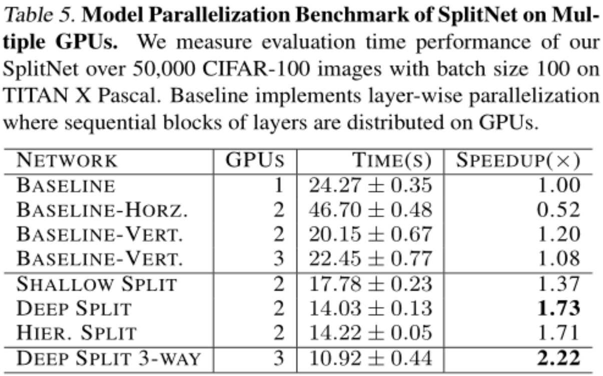


Table 5. **Model Parallelization Benchmark of SplitNet on Multiple GPUs.** We measure evaluation time performance of our SplitNet over 50,000 CIFAR-100 images with batch size 100 on TITAN X Pascal. Baseline implements layer-wise parallelization where sequential blocks of layers are distributed on GPUs.

NETWORK	GPUS	TIME(S)	SPEEDUP(×)
BASELINE	1	24.27 ± 0.35	1.00
BASELINE-HORZ.	2	46.70 ± 0.48	0.52
BASELINE-VERT.	2	20.15 ± 0.67	1.20
BASELINE-VERT.	3	22.45 ± 0.77	1.08
SHALLOW SPLIT	2	17.78 ± 0.23	1.37
DEEP SPLIT	2	14.03 ± 0.13	**1.73**
HIER. SPLIT	2	14.22 ± 0.05	1.71
DEEP SPLIT 3-WAY	3	10.92 ± 0.44	**2.22**

Qualitative Analysis

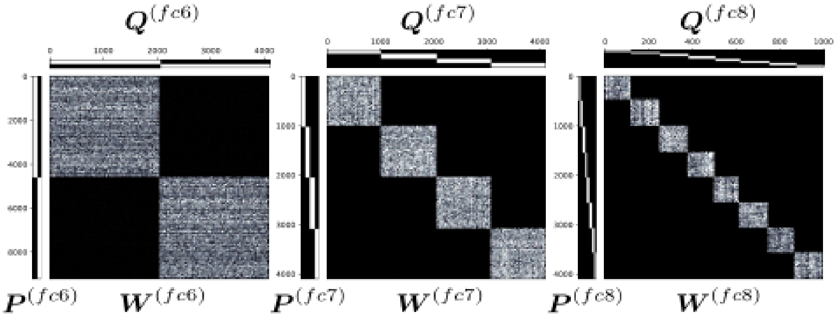


Figure 3. **Learned Groups and Block-diagonal Weight Matrices.** Visualization of the weight matrices along with corresponding group assignments learned in in AlexNet 2-4-8 split. Obtained block-diagonal weights are *split* for faster multiplication. Note the hierarchical grouping structure.

Qualitative Analysis

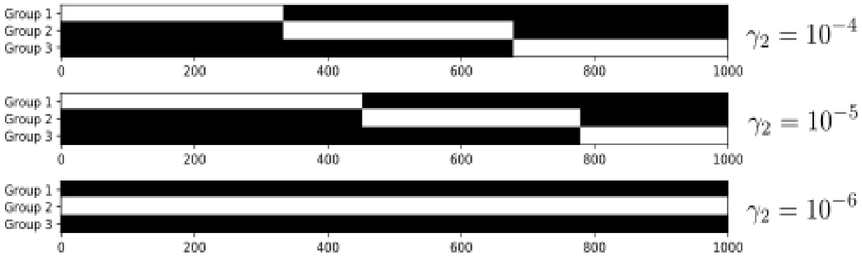


Figure 4. **Effect of Balanced Group Regularization** $R_E(\boldsymbol{P}, \boldsymbol{Q})$. The above figures show group-to-class assignment matrix $\boldsymbol{Q}^{(fc8)}$ for different values of γ_3 on ImageNet-1K with $G = 3$

Qualitative Analysis

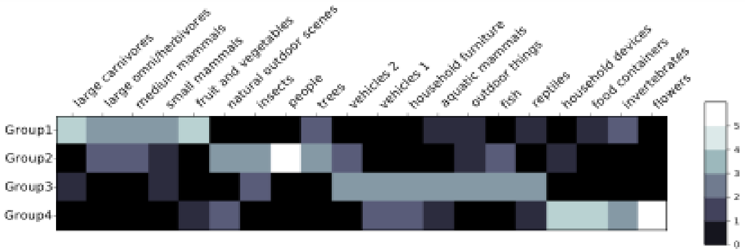


Figure 5. **Learned Groups**. Visualization of grouping learned in FC SplitNet on CIFAR-100. Rows denote learned groups, while columns denote semantic groups provided by CIFAR-100 dataset, each of which includes 5 classes. The brightness of a cell shows the agreement between learned groups and semantic 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