Supplementary File for Paper *SplitNet*: Learning to Semantically Split Deep Networks for Parameter Reduction and Model Parallelization

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

1. Reduced Gradient Method

Given the following optimization J with sum-to-one constraint:

$$\begin{aligned} &\min_{\boldsymbol{d}\in\mathbb{R}^M} J(\boldsymbol{d}) \\ &\text{subject to } \boldsymbol{d} \geqslant 0, \sum_m d_m = 1, \end{aligned} \tag{1}$$

The reduced gradient of J with respect to $\boldsymbol{d}$ is defined as follows(Rakotomamonjy et al., 2008):

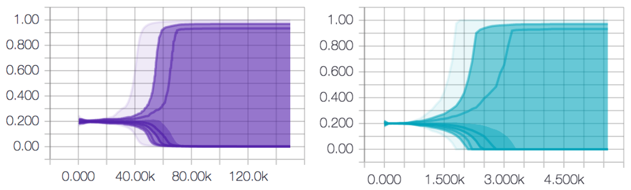
$$(\boldsymbol{\nabla}_{\text{red}} J)_m = \begin{cases} 0 & \text{if } d_m = 0 \text{ and } \dfrac{\partial J}{\partial d_m} - \dfrac{\partial J}{\partial d_\mu} > 0 \\ \dfrac{\partial J}{\partial d_m} - \dfrac{\partial J}{\partial d_\mu} & \text{if } d_m > 0 \text{ and } m \neq \mu \\ -\displaystyle\sum_{\substack{\nu\neq\mu \\ d_\nu>0}} (\dfrac{\partial J}{\partial d_\nu} - \dfrac{\partial J}{\partial d_\mu}) & \text{if } m = \mu \end{cases}, \tag{2}$$

where μ is the index of the largest component of vector $\boldsymbol{d}$. Then we can apply (stochastic) gradient descent to optimize J with $\boldsymbol{d} \leftarrow \boldsymbol{d} - \gamma\boldsymbol{\nabla}_{\text{red}} J$, where γ is the step size. The maximal admissible size of γ is the value that make any component of $\boldsymbol{d}$ zero.

Since the constraint is given as $\sum_g p_{gi} = 1$ in our problem, the reduced gradients method is applied to each group of $\{p_{*i}\}$ and $\{q_{*i}\}$ separately.

1.1. Convergence of Group Assignments

As mentioned in the paper, the direct optimization of group assignment variables with reduced gradients yields faster convergence than optimization via softmax reparametrization. Figure 1 shows the distribution plots, which are provided by TensorFlow, of class-to-group assignments using two methods. Despite starting with lower variance, when the distribution of group assignment variables diverged to 0 and 1. The experiments are done on CIFAR-100 and use WRN-16-8 as the base network. $K = 5$ and all other hyperparameters are same.



(a) Softmax Reparameterization (b) Reduced Gradient Method

Figure 1. **Comparison of Convergence of Group Assignments**. Distribution plot of class to group assignments of SplitNet (a) with softmax reparametrization and (b) reduced gradient method. Note that the x-axis scale of two graphs is different.

2. Group Weight Regularization

Here we attach the full version of the Figure 2 in the paper(Figure 2), illustrating each term of the group weight regularization. As illustrated in the right of the figure, two terms of the regularization $R_W(\boldsymbol{W}, \boldsymbol{P}, \boldsymbol{Q})$ are $\ell_{2,1}$-norm of inter group connections in row and column direction respectively.

3. Deep Split in Residual Networks

Our method can be further extended to residual networks (He et al., 2016) by sharing group assignments over the nodes connected with shortcut connections. Consider a residual building block where a shortcut connection bypasses two convolutional layers. Let $\boldsymbol{W}^{(l_1)}, \boldsymbol{W}^{(l_2)}$ denote the weights of the convolutional layer and $\boldsymbol{p}_{\boldsymbol{g}}^{l_1}, \boldsymbol{q}_{\boldsymbol{g}}^{l_1}, \boldsymbol{p}_{\boldsymbol{g}}^{l_2}, \boldsymbol{q}_{\boldsymbol{g}}^{l_2}$ denote corresponding group assignment vectors for each layer with $\boldsymbol{q}_{\boldsymbol{g}}^{(l_1)} = \boldsymbol{p}_{\boldsymbol{g}}^{(l_2)}$. Since the shortcut identity mapping connects the input nodes of the first convolutional layer to the output nodes of the second convolutional layer, the grouping of these nodes should be shared: $\boldsymbol{p}_{\boldsymbol{g}}^{(l_1)} = \boldsymbol{q}_{\boldsymbol{g}}^{(l_2)}$.

[*]Equal contribution [1]Seoul National University, Seoul, South Korea [2]UNIST, Ulsan, South Korea [3]AITrics, Seoul, South Korea. Correspondence to: Sung Ju Hwang <sjhwang@unist.ac.kr>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

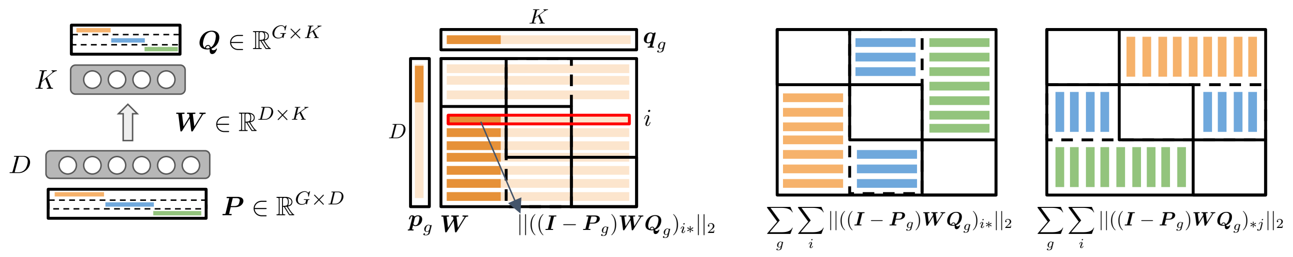

$Q \in \mathbb{R}^{G\times K}$
K
$W \in \mathbb{R}^{D\times K}$
D
$P \in \mathbb{R}^{G\times D}$
K
q_g
D
i
p_g
W
$||((I - P_g)WQ_g)_{i*}||_2$
$\sum_g \sum_i ||((I - P_g)WQ_g)_{i*}||_2$
$\sum_g \sum_i ||((I - P_g)WQ_g)_{*j}||_2$

Figure 2. **Group Assignment and Group Weight Regularization(Full Figure)**. (**Left**) An example of group assignment with $G = 3$. Colors indicate groups. Each row of matrix $\boldsymbol{P}, \boldsymbol{Q}$ is group assignment vectors for group g: $\boldsymbol{p}_g$, $\boldsymbol{q}_g$. (**Center**) Visualization of matrix $(\boldsymbol{I} - \boldsymbol{P}_g)\boldsymbol{W}\boldsymbol{Q}_g$. 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. (**Right**) Group weight regularization $R_W(\boldsymbol{W}, \boldsymbol{P}, \boldsymbol{Q})$ when group assignments are well partitioned. Horizontal and vertical bars painted in each color indicate the parts of $\boldsymbol{W}$ that represent inter-group connections from/to the corresponding group. These colored parts of $\boldsymbol{W}$ are penalized to be near to zero.

4. Experimental Details

4.1. Models Description

Models for CIFAR-100. We use the WRN-16-8(*i.e.* a depth of 16 and a widening factor $k = 8$), as our base network. This network consists of initial conv layer with 16 filters, followed by 6 residual blocks each including 2∼3 conv layers and last fc layer. The number of filters in residual blocks are 128, 256, 512, respectively. Refer to (Zagoruyko & Komodakis, 2016) for details of the WRN structure. We explore four split settings on the structure of WRN-16-8: *FC Split*, *Shallow Split*, *Deep Split*, and *Hierarchical Split*.

We experiment with three split depth settings: 1, 6, 11. The depth 6 and 11 roughly correspond to one-third and two-thirds depth of the WRN-16-8. *FC Split* splits the last softmax fc layer with setting the number of groups to $G = 4$. *Shallow Split* has the depth of 6, including 5 conv layers and the fc layer with $G = 2$. *Deep Split* has the split depth of 11, including 10 conv layers and the fc layer with $G = 2$. We use $G = 2$ for Shallow and Deep Split, mainly because using a large G may substantially cut down the network capacity when recursively splitting multiple layers. Finally, *Hierarchical Split* is a hierarchical version of Deep Split: first 5 layers are split into two supergroups and then the network branches into 4 subgroups for later 6 layers. This results in tree-structured layer networks.

We use weight decay 0.0005 in FC Split and Shallow Split as in baseline. For Deep Split and Hierarchical Split we observe that using the weight decay 0.0001 produces better groupings. However, this causes the network to overfit and we apply dropout to prevent overfitting. In addition, we expect that dropout will help the splitting process as dropout prevents co-adaptation of neurons(Srivastava et al., 2014). We report the test error of baseline and SplitNet variants using the same setting. For data augmentation, we apply random croping and horizontal flipping. Batch normalization is included in all networks. For the CIFAR-100 experiments, we reparameterized group assignment variables in the softmax form.

We hypothesize that the degradation of the deep split is due to insufficient capacity caused by splitting layers, and experiment increasing the capacity of the splitted layers by allowing overlap of the nodes in each layers. Specifically, we allow some nodes to be duplicated and assigned to multiple groups when the network is splitted. Here we regard group assignment for a node as fuzzy if there are multiple groups that the node is assigned to, with values greather than some threshold, say 0.1. During training, we freeze group assignment matrix of a layer if only less than 20% of nodes in the layer remains fuzzy. Then, when splitting the network, these fuzzy nodes are duplicated and distributed to corresponding groups. The resulting SplitNet with overlap has a larger capacity than a standard SplitNet.

The results of the overlap experiments for three SplitNets on WRN-16-8 are shown in the table 1(marked as overlap). In the case of all three splits, the error rate is reduced while increasing the parameter and FLOPs. Note that although the parameter reductions are smaller, the networks are still parallelizable easily.

For comparison with other works in the direction of parameter reduction, we conduct an experiment on WRN-16-8 using SSL(Wen et al., 2016) which uses the (2,1)-norm to induce structured sparsity. Our Shallow Split (23.96%) outperforms SSL(1e-3) (24.15%), while significantly reducing the number of parameters (32.54%) as opposed to 9.40% of SSL.

Models for ImageNet-1K. We test two baseline networks, AlexNet(Krizhevsky et al., 2012) and ResNet-18x2, a variant of ResNet-18(He et al., 2016). First, for AlexNet, we evaluate various split settings from *conv4* to *fc8* layer, and finetune SplitNets from those grouping. The number of splits in the result table indicates the split in *fc6*, *fc7* and *fc8* layer, respectively. For instance, the 2-4-8 split means that *fc6* layer is split into 2 groups and *fc7*, *fc8* layers are split in 4, 8 groups, respectively. All supergroups at each

Table 1. **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	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
FC Split (overlap)	11.0	0.32	3.10	0.0	24.04
Shallow Split (overlap)	8.62	21.67	2.79	9.77	**23.74**
Deep Split (overlap)	8.03	27.02	2.49	19.52	24.10
SSL(1e-3) (Wen et al., 2016)	9.97	9.40	2.70	12.64	24.15
SSL(3e-3) (Wen et al., 2016)	7.44	32.38	2.07	33.14	24.49
SSL(5e-3) (Wen et al., 2016)	4.48	59.31	1.60	48.71	24.65

layer include 2$\sim$3 subgroups at the consecutive layer, so that the numbers of branches in the hierarchical tree are as even as possible. For deeper split, we split 5 layers from *conv4* to the last *fc8* with $G = 2$ in 2$\times$5 split experiment.

ResNet-18x2 is a over-parameterized network that doubles the number of convolutional filters of ResNet-18. It consists of the initial conv layer with 128 filters, 8 residual blocks, each including 2$\sim$3 conv layers with 128$\sim$1024 channels, and the last fc layer. We optimize the grouping in various levels of splits, from *conv4-1* layer at the deepest split, and finetune the SplitNet. The number of splits in the result table indicates the split in *conv4-1&2*, *conv5-1&2*, and the last fc layer, each of which has 5, 5, and 1 split depth, respectively.

We also test SplitNet-Random, where we split fc layers randomly, and remove inter-group connections while keeping the parameters of remaining intra-group connections. We then finetune the SplitNet-Random models for 10$\sim$20 epochs. For these two ImageNet-1K experiments, we directly optimized group assignment variables, p_{gi} and q_{gi}, using reduced gradient algorithm (Rakotomamonjy et al., 2008).

5. Effect of Number of Groups

We experiment how setting the number of groups G affects Shallow SplitNet on CIFAR-100, and report the results in Table 2. Setting $G = 2$ produces the lowest test error. Interestingly, the test error is lower with $G = 4$ than $G = 3$, even with less parameters. Moreover, Shallow SplitNet using $G = 4$ still outperforms the baseline with only 51.36% of parameters. This result strongly supports the motivation of this research, in that often it is important to find the optimal split structure for a given network, in terms of both test accuracy and memory/computation.

6. Training Time Parallelization of SplitNet

Training SplitNet mainly consists of two stages: 1) learning groupings and block-diagonal weight matrices, 2) Splitting the network and finetuning. The second stage can be easily model-parallelized as in test time parallelization. For shared lower layers, gradients computed at each processor may be averaged for weight update. Training time parallelization of the first stage is also implementable, although the networks are not as clean-cut as at test time. We can simply start with an arbitrary assignment of the weight submatrices in the early stage, and as the network is trained we can rearrange the network weights to be disjoint across multiple nodes. However, training-time parallelization is not yet implemented, which requires further research.

Table 2. **Effect of G on Shallow SplitNet on CIFAR-100**

Model	G	Params Reduced (%)	Test error (%)
Baseline		0.0	24.28
Shallow Split	2	32.44	23.94
Shallow Split	3	43.09	25.14
Shallow Split	4	48.66	24.10

References

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

Rakotomamonjy, Alain, Bach, Francis R, Canu, Stéphane, and Grandvalet, Yves. SimpleMKL. *JMLR*, 9:2491–2521, 2008.

Srivastava, Nitish, Hinton, Geoffrey E, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Wen, Wei, Wu, Chunpeng, Wang, Yandan, Chen, Yiran, and Li, Hai. Learning Structured Sparsity in Deep Neural Networks. In *NIPS*. 2016.

Zagoruyko, Sergey and Komodakis, Nikos. Wide Residual Networks. In *BMVC*, 2016.