Combined Group and Exclusive Sparsity for Deep Neural Networks

Jaehong Yoon [1] **Sung Ju Hwang** [1 2]

Abstract

The number of parameters in a deep neural network is usually very large, which helps with its learning capacity but also hinders its scalability and practicality due to memory/time inefficiency and overfitting. To resolve this issue, we propose a sparsity regularization method that exploits both positive and negative correlations among the features to enforce the network to be sparse, and at the same time remove any redundancies among the features to fully utilize the capacity of the network. Specifically, we propose to use an exclusive sparsity regularization based on $(1, 2)$-norm, which promotes competition for features between different weights, thus enforcing them to fit to disjoint sets of features. We further combine the exclusive sparsity with the group sparsity based on $(2, 1)$-norm, to promote both sharing and competition for features in training of a deep neural network. We validate our method on multiple public datasets, and the results show that our method can obtain more compact and efficient networks while also improving the performance over the base networks with full weights, as opposed to existing sparsity regularizations that often obtain efficiency at the expense of prediction accuracy.

1. Introduction

Deep neural networks have shown tremendous success in recent years, achieving near-human performances on tasks such as visual recognition (Krizhevsky et al., 2012; Szegedy et al., 2015; He et al., 2016). One of the key factors in this success of deep network is its expressive power, which is made possible by multiple layers of non-linear transformations. However, this expressive power comes at a cost: increased number of parameters. Due to large number of parameters, deep networks require large amount of memory and computation power to train. Further, large number of parameters also mean that the model is highly susceptible to overfitting as well, if trained with insufficient data. To resolve such issues, researchers have sought ways to make the model more compact and lightweight by parameter reduction, via model compression (Ba & Caruana, 2014; Hinton et al., 2014), or removing unnecessary weights either by pruning (Reed, 1993; Han et al., 2015) and $\ell 1$-regularization (Collins & Kohli, 2014). However, one of the main problems of these methods is that they often achieve such efficiency at the expense of accuracy.

How can we then obtain a compact deep network without sacrificing the prediction accuracy? One way to achieve this goal is better utilizing the capacity of the network, by reducing redundancies in the model parameters. In the optimal case, the weights at each layer will be fully orthogonal to each other, and thus forming an orthogonal basis set. However, since this is a difficult constraint to satisfy, in practice, such constraint is given only at the initialization stage (Saxe et al., 2014), or enforced implicitly through regularizations such as dropout (Srivastava et al., 2014) that prevents feature co-adaption. Contrary to these existing approaches, we propose to impose an explicit regularization to reduce redundancies. Our idea is to enforce network weights at each layer to fit to different sets of input features as much as possible. This exclusive feature learning is implemented by the *exclusive sparsity* regularization based on $(1, 2)$-norm (Zhou et al., 2010; Kong et al., 2014), which basically promotes network weights at each layer to compete for few meaningful features from the lower layer.

However, it is not practical nor desirable to restrict each weight to be completely disjoint from others as some features still need to be shared. For example, if the lower-layer feature is a *wheel*, and the upper layer weights are features describing *car* and *bicycle* respectively, then the two upper layer weights should share the common feature that describes the wheel. Thus, we also allow for sharing of some important features, by introducing an additional group sparsity regularizer based on $(2, 1)$-norm and combine the two regularization terms, balancing their effect at each layer of the network to adjust the degree of feature sharing and competition.

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

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

Our combined regularizer can be applied to all layers of a generic deep neural network, including plain fully-connected feedforward networks and convolutional networks. We validate our regularized network on four public datasets with different base networks, on which it achieves a compact, lighter model while achieving superior performance over networks trained with other sparsity-inducing regularizers, sometimes obtaining even better accuracy than the full model. As an example, on CIFAR-10 dataset, our network obtains 2.17% accuracy improvements while using 13.72% less number of parameters and 35.67% less floating point operations. Further empirical analysis shows that exclusive sparsity helps the network to converge faster to a given error rate, and learn less redundant features.

2. Related Work

Sparsity for deep neural networks Obtaining compact deep networks by removing unnecessary weights, is a long-studied topic in deep learning research. The most simplest yet popular weight removal method is to prune out weak weights by simple thresholding (Reed, 1993; Han et al., 2015). Another way to induce sparsity on weights is by $\ell 1$-regularization (Tibshirani, 1994). Collins & Kohli (2014) applied the $\ell 1$-regularization to convolutional neural networks, demonstrating that it can obtain a compact, memory-efficient network at the expense of small reduction in the prediction accuracy. Few recent work applied group sparsity (Yuan & Lin, 2006) regularization to deep networks, as it has a number of nice properties. By removing an entire feature group, group sparsity can automatically decide the number of neurons (Alvarez & Salzmann, 2016). Further, if applied between the weights at different layers, it can also be used to decide optimal number of layers to use for the given network (Wen et al., 2016). In terms of efficiency, structured sparsity using $(2,1)$-norm exhibits better data locality than the regular sparsity, and results in larger speedups (Wen et al., 2016; Alvarez & Salzmann, 2016). We also employ the group sparsity in our combined regularizer, but we mainly group the features across multiple filters, to promote feature sharing among the filters. While all the previously introduced models do help reduce number of parameters and result in certain amount of speedups, such memory and time efficiency is mostly obtained at the expense of reduced accuracy. Our combined group and exclusive sparsity regularization, on the other hand, do not degenerate performance, since its aim in learning sparse weights/features is in removing redundancy to better utilize the network capacity.

Exclusive feature learning There exists quite a number of work on imposing exclusivity among the learned model parameters/features. One popular way is to enforce orthogonality, as this will minimize the dependency and redundancy among the variables that are being regularized. Orthogonality at initialization stage has been much studied in the deep learning context (Saxe et al., 2014), as in such a non-convex optimization setting this can lead to convergence to a better local optimum. Zhou et al. (2011) enforced orthogonality via explicit dot product regularization to make the parameters for parent-level and child-level classifiers in a hierarchical classifier to be orthogonal. However, the orthogonal regularizer is non-convex and does not scale well, since it scales quadratically to the number of participating vectors. Another way to enforce exclusivity is through $(1,2)$-norm, which is basically the 2-norm over 1-norm groups, that results in promoting sparsity across different vectors. The $(1,2)$-norm is first proposed in (Zhou et al., 2010), where it is used to promote competitions among the models jointly learned in a multi-task learning framework. A similar regularizer was used in (Hwang et al., 2011) in a metric learning setting, with an additional $\ell 1$-regularization that helps learn discriminative features for each metric. Kong et al. (2014) generalized the $(1,2)$-norm to be used with arbitrary objective and handle overlapping groups. In deep learning context, Goo et al. (2016) proposed a difference pooling technique that has a similar motivation to exclusive lasso, which subtracts the common superclass level feature map from the class-specific feature maps to learn class-exclusive features for fine-grained classification. In all existing models, exclusivity is applied only at the class-level, and application of the exclusivity regularization to weights at any layers of deep networks through $(1,2)$-norm, has not yet been explored. Further, our regularizer is a combined term of both group and exclusive lasso which allows sharing of important features while making each weight to be as different as possible, rather than purely exclusive feature learning that is impractical. The regularizer proposed in (Kim & Xing, 2010) is similar to ours, which proposes a weighted $(2,1)$-norm that has a similar effect of varying the degree of competition and grouping, although our regularizer is more explicit in its effect and optimization.

3. Approach

Our main objective is to implement a sparse deep neural network with significantly less number of parameters than what the original non-sparse network has, which at the same time obtains comparable or even better performance to the original model. The training objective for a generic (deep) neural network for classification[1] is given as follows:

$$\min_{\{\boldsymbol{W}^{(l)}\}} \mathcal{L}(\{\boldsymbol{W}^{(l)}\}, \mathcal{D}) + \lambda \sum_{l=1}^{L} \Omega(\boldsymbol{W}^{(l)}) \qquad (1)$$

[1]While our method is generic and can be also applied to regression, we only consider the classification task for simplicity.

Here, $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ is a training dataset with N instances where $\boldsymbol{x}_i \in \mathbb{R}^d$ is a d-dimensional input feature and $y_i \in \{1, \ldots, K\}$ is its class label which is one of the K classes, $\{\boldsymbol{W}^{(l)}\}$ is the set of weights across all layers, $\mathcal{L}(\boldsymbol{W})$ is the loss parameterized by $\boldsymbol{W}$, L is the total number of layers, $\boldsymbol{W}^{(l)}$ is the weight matrix (or tensor) for layer l, $\Omega(\boldsymbol{W}^{(l)})$ is some regularization term on the network weights at layer l, and λ is the regularization parameter that balances the loss with the regularization.

The usual and the most often used regularization term is the 2-norm: $\Omega(\boldsymbol{W}^{(l)}) = \|\boldsymbol{W}^{(l)}\|_2^2$, which is also called as the ℓ_2-regularizer. The regularization has an effect of adding a bias term to reduce variance of the model, which in turn results in a lower generalization error.

However, since our goal is in obtaining a sparse model where large portion of $\boldsymbol{W}^{(l)}$ is zeroed out, we want $\Omega(\boldsymbol{W}^{(l)})$ to be a sparsity-inducing regularizer. The most common regularizer for promoting sparsity is the 1-norm:

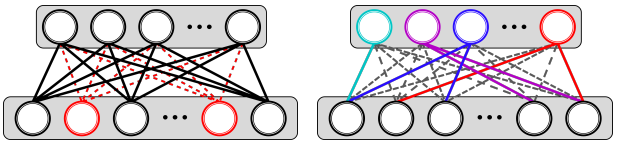
$$\Omega(\boldsymbol{W}^{(l)}) = \|\boldsymbol{W}^{(l)}\|_1 \tag{2}$$

This 1-norm regularization results in obtaining a sparse weight matrix, since it requires the solution to be found at the corner of the 1-norm ball, thus eliminating unnecessary elements. The element-wise sparsity can be helpful when most of the features are irrelevant to the learning objective, as in the data-driven approaches. However, as aforementioned, when applied to a deep network it usually results in slight accuracy reduction. Further, element-wise sparsity, while achieving a memory-efficient model, usually do not result in meaningful speedups in practical network architectures such as CNNs, since the bottleneck is in the convolutional operations that do not reduce much when the number of filters stays the same (Wen et al., 2016).

Group sparsity, on the other hand, can help reduce the intrinsic complexity of the model by eliminating a neuron or a convolutional filter as a whole, and thus can help obtain practical speedups in deep neural networks (Wen et al., 2016; Alvarez & Salzmann, 2016). The group sparsity regularization is defined as follows:

$$\Omega(\boldsymbol{W}^{(l)}) = \sum_g \|\boldsymbol{W}_g^{(l)}\|_2 = \sum_g \sqrt{\sum_i {w_{g,i}^{(l)}}^2} \tag{3}$$

where $g \in \mathcal{G}$ is a weight group, $\boldsymbol{W}_g^{(l)}$ is the weight matrix (or a vector) for group g that is defined on $\boldsymbol{W}^{(l)}$, and $w_{g,i}$ is a weight at index i, for group g. Since ℓ_2-norm has the grouping effect that results in similar weights for correlated features, this will result in complete elimination of some groups, thus removing some input neurons (See Figure 3, (a)). This has an effect of automatically deciding how many neurons to use at each layer.



(a) Group Sparsity (b) Exclusive Sparsity

Figure 1. **Illustration of the regularizers:** (a) When grouping weights from the the same input neuron into each group, the group sparsity has an effect of completely removing some neurons that are not shared across different weights (highlighted in red). (b) Exclusive sparsity, on the other hand, does not result in removal of any input neurons, but rather it makes each upper layer unit to select from a set of lower-layer units, that is disjoint from the sets used by other units.

Still, this group sparsity does not maximally utilize the capacity of the network since there still could be redundancy among the features that are selected. Thus, we propose to apply a sparsity-inducing regularization that obtains a sparse network weight matrix, while also minimizing the redundancy among network weights for better utilization of the network capacity.

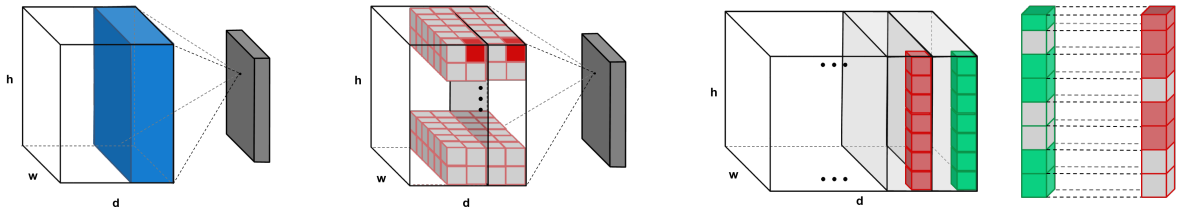
3.1. Exclusive Sparsity Regularization for Deep Neural Networks

Exclusive sparsity, or exclusive lasso was first introduced in (Zhou et al., 2010) in multi-task learning context. The main idea in the work is to enforce the model parameters for different tasks to compete for features, instead of sharing features as suggested by previous work on multi-task learning that leverages group lasso. When the task is a classification task, this makes sense since the objective is to differentiate between classes which can be achieved by identifying discriminative feature for each class.

The generic exclusive sparsity regularization is defined as follows:

$$\Omega(\boldsymbol{W}^{(l)}) = \frac{1}{2}\sum_g \|\boldsymbol{W}_g^{(l)}\|_1^2 = \frac{1}{2}\sum_g \left(\sum_i |w_{g,i}^{(l)}|\right)^2 \tag{4}$$

where $w_{g,i}^{(l)}$ is the i_{th} instance of the submatrix (or the vector) $\boldsymbol{W}_g^{(l)}$. This norm is often called as $(1,2)$-norm, and is basically the 2-norm over 1-norm groups. The sparsity is now enforced within each group, as opposed to the group sparsity regularizer which promotes inter-group sparsity. Applying 2-norm over these 1-norm groups will result in even weights among the groups; that is, all groups should have similar number of non-sparse weights, and thus no group can have large number of non-sparse weight. In (Zhou et al., 2010), $\boldsymbol{W}_g$ is defined to be the model parameter for multiple tasks on the same feature, in which case the (1,2)-norm enforces each task predictor to fit to



(a) Group Sparsity - Filter (b) Group Sparsity - Feature (c) Exclusive Sparsity

Figure 2. **Illustration of the effect of each regularizer on convoultional filters.** (a) Group sparsity, when each group is defined as a filter, can result in complete elimination of some filters that are not shared among multiple high-level filters (Wen et al., 2016; Alvarez & Salzmann, 2016). (b) Group sparsity, when applied across filters for the same feature, will remove certain spatial features as a whole. (c) Exclusive sparsity enforces each convolutional filter to learn features that are as different as possible, by promoting competition among the filters for the same spatial feature.

few features that are most useful for it. Exclusive sparsity can be straightforwardly applied to fully connected layers of a deep network, by grouping network weights from the same neuron at each layer into one group and applying $(1, 2)$-norm on these groups (See Figure 1(b)). This will enforce each output neuron to compete for input neurons, which will result in learning largely disparate network weights at each layer.

Exclusive sparsity on convolutional filters For convolutional layers of a convolutional neural network, exclusive sparsity can be applied in the same manner as in fully connected layers, where we apply Eq. 4 on the convolutional filters, while defining each group g as the same feature across multiple convolutional filters. Figure 2(c) illustrates the feature groups and effect of exclusive sparsity on the convolutional filters. This will enforce the convolutional filters to be as different as possible from each other, removing any redundancies between them.

3.2. Combined Group & Exclusive Sparsity Regularization

As mentioned earlier, our main intuition is that there are varying degree of sharing and exclusivity among different features. Exclusivity alone cannot result in learning an optimal set of features, since some features need to be shared across multiple higher-level features. Thus we need to allow for some degree of sharing across the features, while still making each weight to be sufficiently different in order for each feature to be meaningful. How can we then come up with a regularizer that can achieve the two seemingly conflicting goals?

In tree guided group lasso (Kim & Xing, 2010), each pair of weights are given different degree of sharing and competition based on the similarity between the tasks given by a taxonomy, which can be either semantically defined or obtained through clustering, through a regularization similar to an elastic-net formulation. While this model can be applied at the final softmax layer, on the softmax weight for each class, such taxonomy does not exist for the intermediate level network weights, and it is also not efficient to obtain them through clustering or other means.

Thus we propose to simply combine the group sparsity and the exclusive sparsity together, which will result in a similar effect, where network weights exhibit certain degree of sharing if they are correlated, but are learned to be different on other parts that are not shared. Our combined group and exclusive lasso regularizer is given as follows:

$$\Omega(\boldsymbol{W}^{(l)}) = \sum_g \left((1-\mu_l) \|\boldsymbol{W}_g^{(l)}\|_2 + \frac{\mu_l}{2} \|\boldsymbol{W}_g^{(l)}\|_1^2 \right) \quad (5)$$

where λ is the parameter that decides the entire regularization effect, $\boldsymbol{W}^l$ is the weight matrix for l_{th} layer, and μ_l is the parameter for balancing the sharing and competition term at each layer.

Then how should we set the balancing term μ_l at each layer? One simple solution is to set all μ_l to be a single constant, but a better way is to set them differently at each layer, based on the degree of sharing and competition required at each layer. At lower layers, features will be quite generic and might need to be shared across all high-level neurons for accurate expression of the input data, whereas at the top layer softmax weights, it would be better to have the weights to select features as disjoint as possible for better discriminativity. Thus, we set $\mu_l = m + (1-2m)\frac{l}{L-1}$, to reflect such intuition, where L is a total number of all layers, $l \in \{0, ..., L-1\}$ is an index of each layer, and $0 \leq m \leq 1$ is the lowest parameter value for the exclusive sparsity term. If $m = 0$, the regularizer reduces to $(2, 1)$-norm regularizer with $\mu_1 = 0$ at the lowest layer, while at the topmost softmax layer, the regularizer is an $(1, 2)$-norm regularizer $\mu_1 = 1$.

3.3. Numerical Optimization

Our regularized learning objective can be solved using proximal gradient descent, which is often used for optimizing objectives formed as a combination of both smooth

and non-smooth terms. The proximal gradient algorithm for regularized objective first obtains the intermediate solution $\boldsymbol{W}_{t+\frac{1}{2}}$ by taking a gradient step using the gradient computed on the loss only, and then optimize for the regularization term while performing Euclidean projection of it to the solution space, as in the following formulation:

$$\min_{\boldsymbol{W}_{t+1}} \Omega(\boldsymbol{W}_{t+1}) + \frac{1}{2\lambda s}\|\boldsymbol{W}_{t+1} - \boldsymbol{W}_{t+\frac{1}{2}}\|_2^2 \tag{6}$$

where $\boldsymbol{W}_{t+1}$ is the variable to obtain after the current iteration, λ is the regularization parameter, and s is the step size. When $\Omega(\boldsymbol{W}_{t+1})$ is a group sparsity regularizer or an exclusive sparsity regularizer, the above problem has a closed-form solution.

The solution, or the proximal operator for the group sparsity regularizer, $prox_{GL}(\boldsymbol{W})$ is given as follows:

$$prox_{GL}(\mathbf{W}) = \left(1 - \frac{\lambda}{\|\mathbf{w}_g\|_2}\right)_+ w_{g,i} \tag{7}$$

for all g and i, where g is each group, and i is an element of in each group. The proximal operator for the exclusive sparsity regularizer, $prox_{EL}(\mathbf{W})$, is obtained as follows:

$$\begin{aligned} prox_{EL}(\mathbf{W}) &= \left(1 - \frac{\lambda\|\mathbf{w_g}\|_1}{|w_{g,i}|}\right)_+ w_{g,i} \\ &= sign(w_{g,i})(\big|w_{g,i}\big| - \lambda\|\mathbf{w_g}\|_1)_+ \end{aligned} \tag{8}$$

for all g and i. The combined regularizer can be optimized simply by applying the two proximal operators in a row at each gradient step, after updating the variable with the loss-based gradient. Algorithm 1 describes the proximal gradient algorithm for optimizing our regularized objective.

Algorithm 1 Stochastic Proximal Gradient Algorithm for Combined (2,1)- and (1,2)- regularization

Input: $\mathbf{W}, \lambda, \mu$, mini-batch size b, learning rate η
  Initialize $\mathbf{W}, t$
  while Some predefined stopping criterion is satisfied **do**
    Randomly select b samples from $p \in \{1, 2, ..., n\}$,
    for each layer l, **do**
      $\mathbf{W}^{(l)}_{t+\frac{1}{2}} := \mathbf{W}^{(l)}_t - \frac{\eta s_t}{b}\sum_p \bigtriangledown f_p(\mathbf{W}^{(l)}_t)$ ▷ Update the parameter with the gradient of a non-regularized objective
      $\mathbf{W}^{(l)}_{t+\frac{1}{2},GL} = prox_{GL}(\mathbf{W}^{(l)}_{t+\frac{1}{2}})$ ▷ Apply $prox_{GL}$ in Eq. 7
      $\mathbf{W}^{(l)}_{t+1} = prox_{EL}(\boldsymbol{W}^{(l)}_{t+\frac{1}{2},GL})$ ▷ Apply $prox_{EL}$ in Eq. 8
    end for
  end while

4. Experiment

We perform all experiments with convolutional neural network as the base network model. The regularization is applied at the network weights for all layers, excluding the bias term. All models are implemented and experimented using Tensorflow (Abadi et al., 2016) framework [2].

Baselines and our models We compare our regularized networks against relevant baselines.

1) ℓ_2. The network trained with ℓ_2-regularization.

2) ℓ_1. The network trained with ℓ_1-regularization, which has elementwise sparse network weights.

3) Group Sparsity-Filter. The network regularized with $\ell_{2,1}$-norm on the weights, which groups each convolutional filter as a group at convolutional layers. This network is an implementation of the model in (Wen et al., 2016).

4) Group Sparsity-Feature. The network that uses the same $\ell_{2,1}$-regularization as in 3), but with each group defined as the same feature at different filters.

5) Exclusive Sparsity. This is the network whose weights at each layer are regularized with $\ell_{1,2}$-norm only.

6) Combined Group and Exclusive Sparsity. The network regularized with our combined structured sparsity on the weights. The combination weight that balances both regularizations are dynamically set at each layer.

Datasets and base networks We validate our method on four public datasets for classification, with four different convolutional networks.

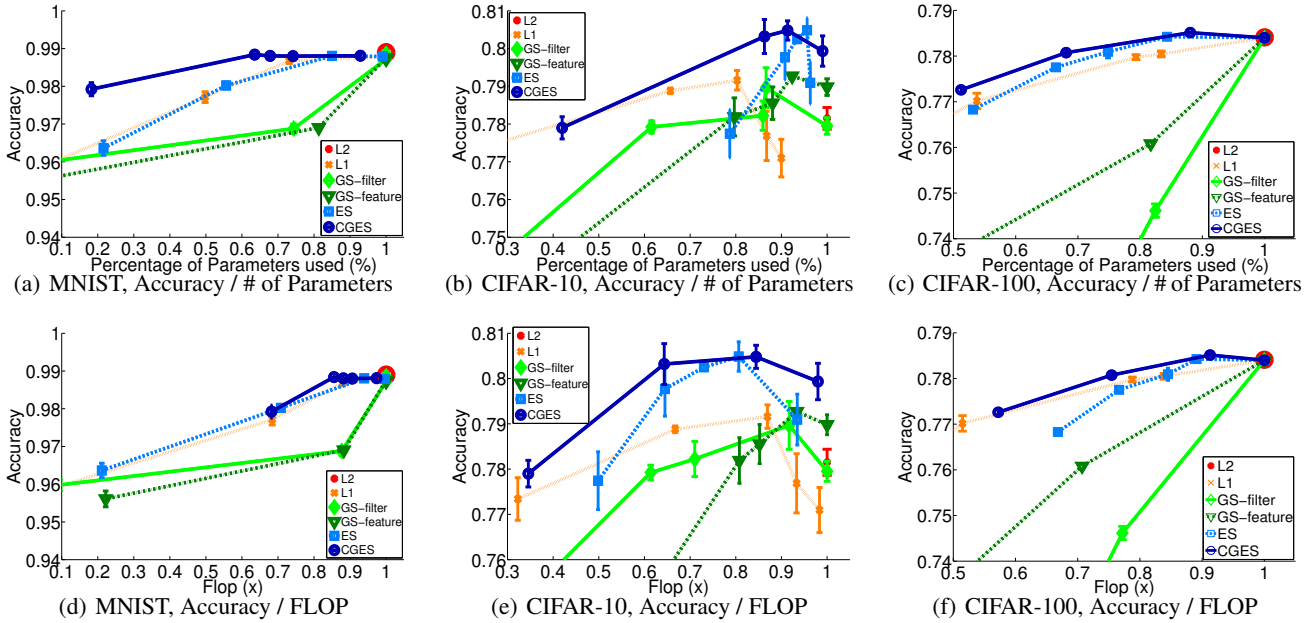
1) MNIST. This dataset contains $70,000$ 28×28 grayscale images of handwritten digits for training example images, where there is $6,000$ training instances and $1,000$ test instances per class. As for the base network, we use a simple convolutional neural network with two convolutional layers and two fully connected layers.

2) CIFAR-10. This dataset consists of $60,000$ images sized 32×32, from ten animal and vehicle classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck). For each class, there are $5,000$ images for training and $1,000$ images for test. For the base network, we use LeNet (Lecun et al., 1998), that has two convolutional layers followed by three fully connected layers.

3) CIFAR-100. This dataset also consists of $60,000$ images of 32×32 pixels as in CIFAR-10, but has 100 generic object classes instead of 10. For each class, 500 images are used for training and 100 images are used for test. For the base network, we use a variant of Wide Residual Network (Zagoruyko & Komodakis, 2016), which has 16 layers with the widening factor of $k = 10$.

4) ImageNet-1K. This is the dataset for 2012 ImageNet

[2] Codes available at https://github.com/jaehong-yoon93/CGES



(a) MNIST, Accuracy / # of Parameters (b) CIFAR-10, Accuracy / # of Parameters (c) CIFAR-100, Accuracy / # of Parameters

(d) MNIST, Accuracy / FLOP (e) CIFAR-10, Accuracy / FLOP (f) CIFAR-100, Accuracy / FLOP

Figure 3. **Accuracy-efficiency trade-off.** We report the accuracy over number of parameters, and accuracy over FLOP to see how each sparsity-inducing regularization at various sparsity range affect the model accuracy. The reported results are average model accuracy over three runs (with random weight initialization), and the errorbars denote standard errors for 95% confidence interval. L2 and L1 are the networks trained with ℓ_2-regularization, ℓ_1-regularization, GS-filter and GS-feature are filter-wise and feature-wise group sparsity respectively, ES is our exclusive sparsity regularizer, and CGES is our proposed combined group and exclusive sparsity regularizer.

Large Scale Visual Recognition Challange (Deng et al., 2009) that consists of $1,281,167$ images from $1,000$ generic object categories. For evaluation, we used the validation set that consists of $50,000$ images, following the standard procedure. For the base network, we used an implementation of AlexNet (Krizhevsky et al., 2012).

For MNIST and CIFAR-10 experiment, we train all networks from the scratch; for CIFAR-100, and ImageNet-1K experiment where we use larger networks (WRN and AlexNet) we fine-tune the network from the ℓ_2- regularized networks, since training them from scratch takes prohibitively long time.

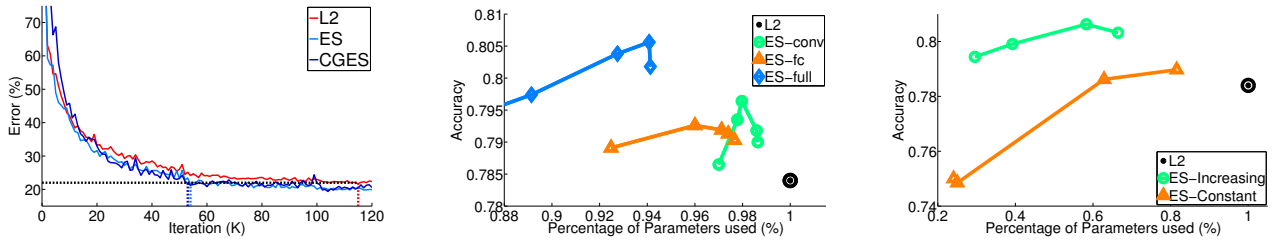
4.1. Quantitative analysis

We first validate whether our sparsity-inducing regularizations result in better accuracy-efficiency trade-off compared to baseline methods, by measuring the prediction accuracy over number of parameters, and number of floating point operations (FLOP) for each method.

Figure 3 shows the prediction accuracy of the different models over number of parameters/FLOP, obtained by differentiating the sparsity-inducing regularization parameter for each method. As expected, ℓ_1-regularization greatly reduces the number of parameters, while maintaining a similar performance to the original model. The group sparsity regularization in general performs worse than ℓ_1, but achieves better accuracy in certain sparsity ranges. The exclusive sparsity improves the performance over the base ℓ_2-regularization model in low-sparsity range which is especially well shown in CIFAR-10 result, but degenerates performance as the sparsity increases. We attribute this to the fact that exclusive sparsity aims to make each weight/filter to fit to completely disjoint sets of low-level features, which is unrealistic as features may need to fit to the same set of low-level features for accurate representation.

Finally, our combined group and exclusive sparsity, which allows for certain degree of sharing between the weights/features while enforcing exclusivity, achieves the best accuracy/parameter trade-off, achieving similar or better performance gain to the exclusive sparsity while also greatly reducing the number of parameters. Fig 3(a) shows the results on the MNIST dataset, on which our CGES obtains no accuracy reduction, using 36.48% less number of parameters and 14.46% less computation. On CIFAR-10 dataset, CGES improves the classification accuracy over the ℓ_2 baselines by 2.17%, using 13.72% less number of parameters using 35.67% less FLOP. CGES obtains slight accuracy reduction of 1.15% on CIFAR-100 dataset, using only 51.22% of parameters and 42.77% less FLOP.

On ImageNet (Table 1), CGES obtains similar or slightly worse performance to the full network while using $60\% - 68\%$ of its parameters, while ℓ_1 shows noticeable performance degeneration at the same sparsity level.



(a) Convergence speed (b) Conv vs. FC layers (c) Effect of μ_l

Figure 4. Further Analysis of the exclusive sparsity on CIFAR-10 dataset. **(a) Convergence speed:** Networks regularized with ES (Light Blue) or CGES (Dark Blue) converge fastest to a given error rate, compared to ℓ_2. **(b) Effect of exclusive sparsity at different types of layers:** The network regularized with exclusive sparsity at all layers performed better with higher sparsity, compared to models that used ES only at convolutional, or fully connected layers. **(c) Effect of μ_l:** ES-increasing is our combined regularizer, where exclusivity increases with network layer l. For ES-constant, we set $\mu_l = 0.5$ at all layers.

Table 1. Accuracy-efficiency trade-off on the Imagenet dataset.

Model	Accuracy	% Params.	Accuracy	% Params.
L2	59.89%	100.0%	-	-
L1	57.55%	60.39%	58.45%	66.75%
ES	57.99%	60.41%	58.89%	67.37%
CGES	**58.56%**	60.66%	**59.25%**	67.24%

Table 2. Performance of CGES coupled with iterative pruning. The reported results are averages over 3 runs and standard errors for 95% confidence interval.

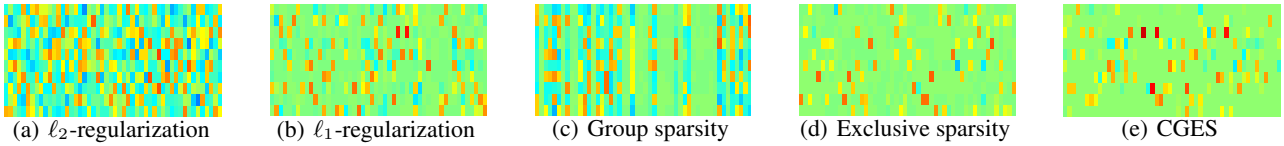
Model	MNIST	CIFAR-10
L2 (Full Network)	99.20%	78.15%
Han et al. (2015)	$98.71 \pm 0.03\%$	$76.37 \pm 0.42\%$
CGES	$\mathbf{99.16 \pm 0.03\%}$	$\mathbf{78.97 \pm 0.41\%}$

Iterative pruning Iterative pruning (Han et al., 2015) is another effective method for obtaining a sparse network while maintaining high accuracy. As iterative pruning is orthogonal to our method, we can couple the two methods to obtain even better performance per number of parameters used; specifically, we replace the usual weight decay regularizer used in (Han et al., 2015) with our CGES regularizer. We report the accuracy of this combined model on MNIST and CIFAR-10 dataset, when using 10% of the parameters of the full network (Table 2). The results show that CGES coupled with iterative pruning obtains similar or even better results to the original model using only a fraction of the parameters, significantly outperforming the base pruning model which suffers substantial accuracy loss.

Convergence speed We further analyze the empirical convergence rate of our regularized network, since it will be impractical if the regularized network requires much longer iterations to reach the same accuracy. Interestingly, we empirically found that our exclusive sparsity regularizer also helps network achieve the same error using much fewer iterations (Figure 4(a)), compared to base ℓ_2-regularization. This faster convergence agrees with the observations in (Saxe et al., 2014), where networks whose weights are initialized as random orthgonal matrices converged faster than networks with random Gaussian initialization.

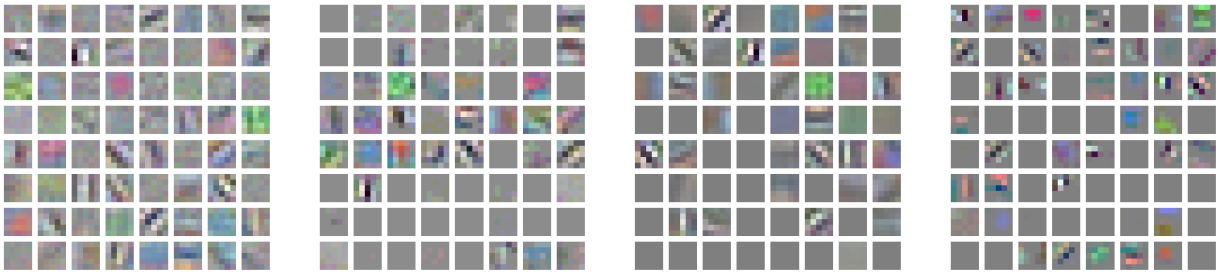
Convolutional vs. fully connected layers To see how much effect our combined regularizer has on different types of layers, we experiment applying the model only to the fully connected layer, or convolutional layers, while applying usual ℓ_2-regularizer to other layers. Figure 4(b) shows the result of this experiment, where we plot the accuracy over percentage of parameters used, for models that applies ES only to fully connected layers, only to convolutional layers, and both. We observe improvements on all models, which shows the effectiveness of the exclusive sparsity regularizer to all types of network weights. Further, ES results in larger improvements on convolutional layers, which makes sense since lower-layer features are more important as they are more generic across different classes, than the features learned at fully connected layers. However, conv layers obtained the best accuracy at low-sparsity range, since strict enforcement of exclusivity hurts the representational power of the features, whereas FC layers obtained improvements even on high-sparsity range; this may be because loss of expressiveness could be compensated by better discriminativity of the features at high level.

Sharing vs. Competing for Features We further explore how varying the degree of sharing and competition affect the accuracy and efficiency of the model, by experimenting with different configurations of μ_l in Eq. 5 at each layer. We report the results in Figure 4(c). Specifically, we test two different approaches to balance the degree of sharing and competition at each layer. The first model, ES-Increasing, is the actual combination we have used in our method which increases the effect of exclusive sparsity with increasing l. This model reflects our intuition that competition will help at high layers, while sharing will help more at lower layers. The second model, ES-Constant combines the two terms with $\mu_l = 0.5$ throughout all layers. We observe that ES-Increasing works better than ES-



(a) ℓ_2-regularization (b) ℓ_1-regularization (c) Group sparsity (d) Exclusive sparsity (e) CGES

Figure 5. Visualizations of the last fully connected layer weights on the CIFAR-10 dataset. These figures show the weight of first 50 weights out of 192 weights. The rows are output units for each class and the columns are features.



(a) ℓ_2, Sparsity: 0.00 % (b) ℓ_1, Sparsity: 71.3 % (c) GS-Filter, Sparsity: 54.7 % (d) CGES, Sparsity: 73.5 %

Figure 6. Visualizaiton of the 1^{st} convolution layer filters from the network trained on CIFAR-10 dataset. (a) ℓ_2-regularization results in smooth non-sparse filters. (b) ℓ_1-regualrization results in filters that are elementwise sparse. (c) GS-Filter results in complete removal of some filters. (d) CGES obtains sharper filters with some spatial features completely zeroed out, from competition among the filters.

Constant across all sparsity ranges, which shows that our scheme of increasing exclusivity at higher layers indeed helps improve the model performance.

4.2. Qualitative analysis

For further qualitative analysis, we visualize the weights and convolutional filters obtained using the baselines and our methods.

Figure 5 visualizes the weights of the softmax layer for different regularization methods, from the network trained on the CIFAR-10 dataset. Each row is the softmax parameter for each class. ℓ_2 and ℓ_1 work as expected, resulting in non-sparse and elementwise sparse weights. The group sparsity regularizer results in the total elimination of certain features that are not shared across multiple classes. The exclusive sparsity regularizer, when used on its own, results in disjoint feature selection for each class. However, when combined with the group lasso, it allows certain degree of feature reuse, while still obtaining parameters that are largely disparate across classes.

To show that such effect is not confined to the fully connected layer, we also visualize the convolutional filters in the first convolutional layer of the network trained on the CIFAR-10 dataset, in Figure 6. We observe that the combined group and exclusive sparsity regularizer results in filters that are much sharper than the ones that are obtained by ℓ_1 or group sparsity regularization, with some spatial features dropped altogether from the competition with other filters. Further, there is less redundancy among the filters, unlike the filters learned by other regularization methods. Note that we set the exclusivity factor $\mu_1 = 0.8$ just for visualization purpose, since our weighting scheme will set μ_1 as a low value in the first convolutional layer.

5. Conclusion

In this work, we proposed a novel regularizer for generic deep neural networks that effectively utilizes the capacity of the network, by exploiting the sharing and competing relationships among different network weights. Specifically, we propose to use an exclusive sparsity regularization based on $(1, 2)$-norm on the network weights, along with group sparsity regularization using $(2, 1)$-norm, such that exclusive sparsity enforces the network weights to use input neurons that are as different as possible from the other weights, while the group sparsity allows for some degree of sharing among them, as it is impossible to make the network weights to fit to completely disjoint set of features. We validate our method on some public datasets for both the accuracy and efficiency against other sparsity-inducing regularizers, and the results show that our combined regularizer helps obtain even better performance than the original full network, while significantly reducing the memory and computation requirements.

Acknowledgements This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2016M3C4A7952634), and UAV-technology development

program through the National Research Foundation of Korea (NRF) funded by the Korea Aerospace Research Institute (NRF-2016M1B3A1A01937742).

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, et al. Tensorflow: Large-scale Machine Learning on Heterogeneous Distributed Systems. *arXiv:1603.04467*, 2016.

Alvarez, Jose M and Salzmann, Mathieu. Learning the number of neurons in deep networks. In *NIPS*. 2016.

Ba, Jimmy and Caruana, Rich. Do deep nets really need to be deep? In *NIPS*, 2014.

Collins, Maxwell D and Kohli, Pushmeet. Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*, 2014.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and ei, L. Fei-F˙ Imagenet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

Goo, Wonjoon, Kim, Juyong, Kim, Gunhee, and Hwang, Sung Ju. Taxonomy-Regularized Semantic Deep Convolutional Neural Networks. In *ECCV*, 2016.

Han, Song, Pool, Jeff, Tran, John, and Dally, William. Learning both weights and connections for efficient neural network. In *NIPS*. 2015.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *CVPR*, 2016.

Hinton, Geoffrey, Vinyals, Oriol, and Dean, Jeff. Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*, 2014.

Hwang, Sung Ju, Grauman, Kristen, and Sha, Fei. Learning a tree of metrics with disjoint visual features. In *NIPS*, 2011.

Kim, S. and Xing, E. P. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*, pp. 543–550, 2010.

Kong, Deguang, Fujimaki, Ryohei, Liu, Ji, Nie, Feiping, and Ding, Chris. Exclusive feature learning on arbitrary structures via ℓ_1, 2-norm. In *NIPS*. 2014.

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

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Reed, R. Pruning algorithms-a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747, Sep 1993. ISSN 1045-9227. doi: 10.1109/72.248452.

Saxe, Andrew M., McClelland, James L., and Ganguli, Surya. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*, 2014.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going Deeper with Convolutions. In *CVPR*, 2015.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

Wen, Wei, Wu, Chunpeng, Wang, Yandan, Chen, Yiran, and Li, Hai. Learning structured sparsity in deep neural networks. In *NIPS*, pp. 2074–2082. 2016.

Yuan, Ming and Lin, Yi. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006.

Zagoruyko, Sergey and Komodakis, Nikos. Wide residual networks. In *BMVC*, 2016.

Zhou, D., Xiao, L., and Wu, M. Hierarchical Classification via Orthogonal Transfer. In *ICML*, 2011.

Zhou, Yang, Jin, Rong, and Hoi, Steven C. H. Exclusive lasso for multi-task feature selection. *Journal of Machine Learning Research*, 9:988–995, 2010.