Structured Compression of Deep Neural Networks with Debiased Elastic Group LASSO

Oyebade K. Oyedotun, Djamila Aouada, Björn Ottersten
Interdisciplinary Centre for Security, Reliability and Trust (SnT),
University of Luxembourg, L-1855 Luxembourg
{oyebade.oyedotun, djamila.aouada, bjorn.ottersten}@uni.lu

Abstract

State-of-the-art Deep Neural Networks (DNNs) are typically too cumbersome to be practically useful in portable electronic devices. As such, several works pursue model compression that seeks to drastically reduce computational memory footprints, FLOPS and memory for storage. Many of these works achieve unstructured compression, where the compressed models are not directly useful since dedicated hardware and specialized algorithms are required for storage of sparse weights and fast sparse matrix-vector multiplication respectively. In this paper, we propose structured compression of large DNNs using debiased elastic group LASSO (DEGL), which is motivated by different interesting characteristics of the individual components. That is, where group LASSO penalty enforces structured sparsity, l2-norm penalty promotes features grouping, and debiasing disentangles sparsity and shrinkage effects of group LASSO. We perform extensive experiments by applying DEGL to different DNN architectures including LeNet, VGG, AlexNet and ResNet on MNIST, CIFAR-10, CIFAR-100 and ImageNet datasets. Furthermore, we validate the effectiveness of our proposal on domain adaptation using Oxford-102 flower species and Food-5K datasets. Results show that DEGL can compress DNNs by several folds with small or no loss of performance. Particularly, DEGL outperforms conventional group LASSO and several other state-of-the-art methods that perform structured compression.

1. Introduction

Computer vision tasks benefit from the success of DNNs, as several interesting results have been reported for different tasks [12, 25, 22]. Importantly, exceptional results on challenging datasets have been obtained using large DNN models [28, 14]. On one hand, there seems to be a positive correlation between model size and performance for many tasks [41, 23]; many works posit that large models allow the expoloration of extensive solution configurations, and thus reduces the possibly of getting stuck in poor local minima. On the other hand, these high-performance models are usually too cumbersome[1] for deployment in real-life applications. A solution to the aforementioned problems of large DNNs is to train from scratch smaller models with acceptable computational memory footprint, FLOPS and memory for storage. However, such small models typically incur noticeable generalization loss in comparison to large models. Consequently, model compression [29, 5] seeks to address the problems associated with cumbersome models, but incur minimal or even no performance loss in comparison to the reference large models.

Model compression results can be classified as unstructured or structured. In unstructured compression [32], the outcome is implicitly compressed, but the overall architecture of the compressed model is the same as the original (or reference) large model. Consequently, the aforementioned benefits of compression are not directly obtainable; dedicated hardware to store model weights with unstructured sparsity are required to realize a reduction in memory size, specialized algorithms are required for fast matrix-vector multiplications to realize a reduction in computational footprint and faster inference in comparison to the original large model [18]. In contrast, structured compression [44, 1] leads to an explicitly smaller model that requires no ad-hoc algorithms or specialized hardware for operation.

Model compression works [32, 18] have directly taken inspiration from *sparse signal processing* [46, 33]. The convergence and transferability of key concepts in signal processing to neural networks (NNs) is not surprising, since one can consider NNs as directed graphs with information flowing through them. Several works [18, 27] on unstructured and structured model compression have relied on enforcing some form of sparsity in model parameters by incorporating various penalty terms in the model's cost function.

[1]The term 'cumbersome'is used to refer to the computer memory size and computational footprint

Depending on the specific characteristics of the penalty term used, different forms of sparsity can be achieved for model parameters. It is noteworthy that these sparsity-enforcing penalty terms that are now used for model compression are essentially not new; they have existed in classical signal processing for several years [33, 24]. Very good results have been reported using such penalty terms for compressing DNNs, and thus their characteristics deserve to be investigated even further for improved results. For enforcing unstructured and structured sparsity, LASSO [33] and group LASSO [39] penalties, respectively, have been reported to yield interesting results. Motivated by the aforementioned drawbacks of unstructured model compression, we focus in this paper on structured model compression based on group LASSO.

We start by examining the drawbacks of group LASSO (which simply reduces to LASSO in the case where each group is considered as one feature [10]) in the context of model compression. First is the problem of *LASSO saturation* [46, 4] that occurs when the number of model parameters is considerably larger than the number of training data points; here, the number of features that LASSO selects is at most the number of training data points. This poses a huge limitation for application in DNNs where it is customary that the number of model parameters are exponentially higher than available training data points. Second is that group LASSO does not work well with correlated features [46]; the work [40] discusses the problems of ordinary least squares (OLS), LASSO and group LASSO with correlated variables. If, for instance, sets of features form groups, LASSO 'randomly' selects one set from each group, and discards the rest [34, 37, 42]. Interestingly, features correlation (especially among hidden units or convolution filters) is highly prevalent in DNNs [13, 2]. Third is that group LASSO simultaneously enforces both feature selection (which is desirable for model compression) and feature shrinkage (which is not so desirable) [46]; as such, model compression (via features selection) is entangled with model regularization (via feature shrinkage). Obviosuly, this is a concern since our actual interest lies in feature selection which characterizes truly important features (identified by non-zero valued weights), as other features (with zero valued weights) can be discarded without hurting model performance. In fact, the shrinkage effect of group LASSO can be so impactful such that it causes models to underfit training data [42].

Our proposal in this paper is aimed at tackling the aforementioned problems encountered with group LASSO for compressing DNNs. The overall framework of the proposed compression method is shown in Figure 1. Namely, the contributions in this paper can be summarized as follows:

1. Compression of DNNs using $l2$-norm penalty and group LASSO, where group LASSO enforces group sparsity of parameters, and $l2$-norm promotes grouping and selection stability of correlated features.

2. Model debiasing after pruning by eliminating group LASSO prior to the retraining phase. This simple step disentangles the interwoven impact of feature selection and shrinkage, and thus ameliorate interpretability.

3. Experimental validation on various DNN architectures using six benchmarking datasets. Improved results over conventional group LASSO and several state-of-the-art approaches are reported.

The rest of this paper is organized as follows. In Section 2, related works are discussed. Section 3 presents as background conventional group LASSO and subsequntly the problem statement. Section 4 describes the proposed approach that addresses the problems highlighted in Section 3. Section 5 contains experimental results and discussions. The paper is concluded in Section 6.

2. Related work

High performance DNNs are typically cumbersome for real-life applications. Consequently, several works [15, 8] have proposed different methods of compression.

In [38], neuron importance score was defined for assessing the impact of hidden neurons in trained models; the score takes into account the reconstruction of important responses in the layer before the classification layer. It is argued that global pruning methods yield better results as opposed to pruning methods that take into account only adjoining layers. Soft filter pruning was proposed in [7], where pruning is performed dynamically. During training, previously pruned filters can be included again and participate in learning if it becomes necessary.

Sparse convolutional neural network was proposed that in [18]. This work employed sparsity-inducing contraints to impose sparsity in model parameters during training. The original model is trimmed after training using some defined thresholds to determine inconsequential weights. Unfortunately, the compression is unstructured, and thus the reported results, despite being good, are not directly usable as discussed earlier. In [35], group LASSO is used for learning structured sparsity in DNNs; interesting results are reported given that they directly achieve model speedup during inference without recourse to custom algorithms. In a similar work [27], sparse group sparsity was proposed. The idea is similar in spirit with [18]; that is, imposing unstructured sparsity among model parameters. Again, the overall result leads to unstructured sparsity and thus suffers the aforementioned drawbacks. In [6], $l2$-norm penalty was used to enforce model parameters to have small values. Although good results were achieved, the sparsity of resulting models were unstructured. In another work [9], LASSO

penalty is used for channel selection and then subsequent pruning. The work reported compact models with reduced model size and accelerated inference.

3. Background and problem statement

This section discusses the simple premise that the application of conventional group LASSO can be suboptimal for the purpose of compressing DNNs.

3.1. Background

Consider the cost function, $J(W)$, for a DNN parameterized by W; where $W = \{W_1, W_2, ..., W_l, .., W_L\}$, and W_l is the weight matrix at layer l; and L is the total number of weight layers in the DNN. For instance, say $J(W)$ is the Mean Squared Error (MSE) given as

$$J(W) = \underset{W}{\operatorname{argmin}} \frac{1}{2N} \| \mathbf{y_d} - \mathbf{y_o} \|^2, \tag{1}$$

such that

$$\mathbf{y_o} = f(x; W), \tag{2}$$

where x is the input data; f is the mapping function defined by the DNN; $\mathbf{y_d}$ and $\mathbf{y_o}$ are the desired and computed output vectors, respectively; and N is the number of training data points. Given that the group of features at layer l is denoted W_l^g and there are G groups, the new cost for the DNN with group LASSO, $J_{gl}(W)$, can then be written as

$$\left.\begin{aligned} J_{gl}(W) = \underset{W}{\operatorname{argmin}} J(W), \quad \text{s.t.} \\ \| W_l \|_2 = \sum_{g=1}^{G} \| W_l^g \|_2 \leq c : \forall\, 1 \leq l \leq L. \end{aligned}\right\} \tag{3}$$

Note that each convolution filter or all the weights of a hidden unit are taken as a group; that is, g. In Langragian form, (3) can be written explicitly as

$$J_{gl}(W) = \underset{W}{\operatorname{argmin}} \left\{ J(W) + \gamma \sum_{g=1}^{G} \sum_{l=1}^{L} \| W_l^g \|_2 \right\}, \tag{4}$$

where the relationship between γ and c is such that $\gamma \propto 1/c$.

3.2. Problem statement

We herein discuss the problems that can impact the application of group LASSO to the task of DNN compression.

3.2.1 Saturation

For problems where the total number of model parameters, θ_t, is considerably larger than the number of training samples, N, LASSO does not select more than N features prior to saturation [46, 4]. This can severely impact the expressiveness of the resulting compact model. A formal treatment and proof of this scenario can be found in [26]. Interestingly, the case $\theta_t \gg N$ is extremely prevalent in DNNs, and thus deserves to be addressed when LASSO is employed for enforcing parameters sparsity.

3.2.2 Correlated features and grouping

It has been shown that for correlated features, LASSO simply *randomly* selects a feature from the group and discards the rest [46]. This ultimately results in instability of feature selection over different training runs. Coincedentally, DNNs are popular for learning highly correlated features [13, 2]. As such, for model compression techniques that employ LASSO, features selection instability is a concern in view of model interpretability.

3.2.3 Entangled feature selection and shrinkage

An examination of LASSO penalty in [46, 33] reveals that LASSO performs both feature selection and shrinkage. This combined impact could be interesting for specific applications where alleviating model overfitting is the main objective; thus, being unable to identify their individual impact is of no concern. However, in model compression, our main goal is feature selection to facilitate interpretability and therefore compactness. As such, it becomes important to disentangle feature selection and shrinkage when LASSO is employed for model compression. Additionaly, LASSO shrinkage effect can over-penalize model parameters so that model underfits [42].

4. Proposed approach

We present in Figure 1 the proposed approach that addressesthe challenges discussed in Section 3.2. The objective is to obtain compact DNNs by employing a more interesting penalized cost function than the conventional group LASSO.

4.1. Debiased Elastic Group LASSO (DEGL)

We propose Debiased Elastic Group LASSO (DEGL) that aims to separate model selection via elastic group LASSO (EGL) from model estimation via ridge regression.

4.1.1 Elastic group LASSO

Motivated by the success of elastic net [46], the problems of LASSO saturation and erratic selection for correlated features discussed in Sections 3.2.1 and 3.2.2 are addressed in this paper by incorporating $l2$-norm penalty into the group LASSO cost given in (5). The Elastic group LASSO (EGL) cost function that is proposed to enforce group sparsity, promote features grouping and alleviate selection instability is

$$\left.\begin{aligned} J_{egl}(W) = \underset{W}{\operatorname{argmin}} J(W), \ \text{s.t.} \\ \| W_l \|_2 = \sum_{g=1}^{G} \| W_l^g \|_2 \leq c_1 : \forall\, 1 \leq l \leq L, \\ \| W_l \|_2^2 \leq c_2 : \forall\, 1 \leq l \leq L. \end{aligned}\right\} \tag{5}$$

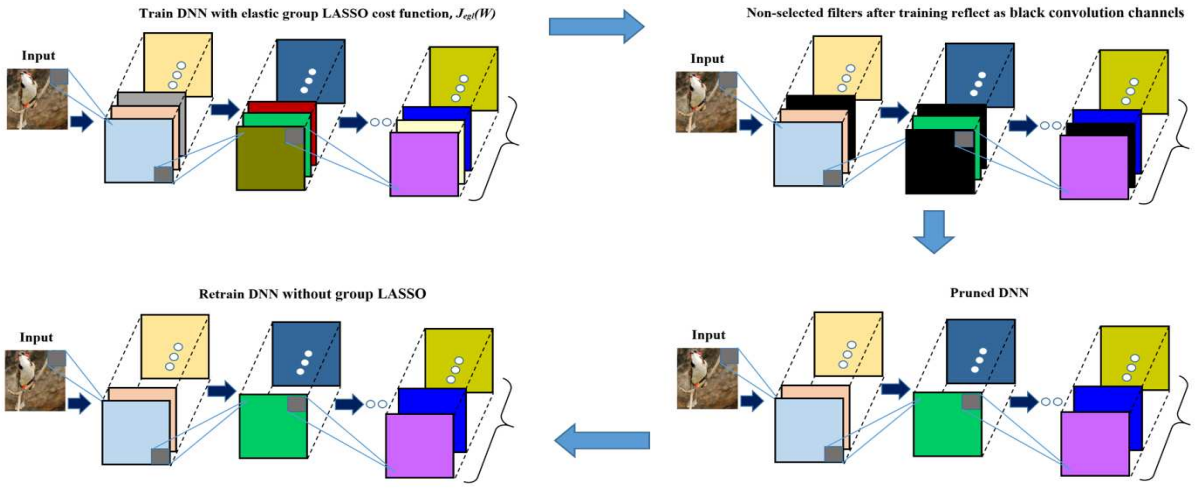

Figure 1: Overall framework for the proposed approach in Section 4. Top left: DNN is trained using the proposed elastic group LASSO cost function, $J_{egl}(W)$, in Section 4.1.1. Top right: the non-selected filters after training translate to the black convolution channels. Bottom right: Non-selected filters are pruned, as in Section 4.1.2. Bottom left: Pruned model is retrained without group LASSO, as in Section 4.1.3

Again, in Langragian form, (5) can be combined and written explicitly as

$$J_{egl}(W) = \underset{W}{\operatorname{argmin}} \left\{ J(W) + \gamma_g \sum_{g=1}^{G} \sum_{l=1}^{L} \| W_l^g \|_2 + \lambda \sum_{l=1}^{L} \| W_l \|_2^2 \right\}, \tag{6}$$

where γ_g addresses over-penalization of small feature groups by taking into account the length of each group, ρ_g, for any given γ as in

$$\gamma_g = \gamma \sqrt{\rho_g}. \tag{7}$$

4.1.2 Pruning unimportant filters

Given a specific filter m out of M filters in layer l denoted W_l^m, we assess its importance by computing the maximum value of its individual absolute weight values, $w_l^{m_k}$ as in

$$max(W_l^m) = max\{ \mid w_l^{m_k} \mid \}_{k=1}^{K}, \tag{8}$$

where k indexes the individual weights in filter W_l^m. The set of unimportant filters in layer l that least contribute to model performance is denoted W_l^p. Given the pruning threshold t_{th}, W_l^p is initialized as an empty set, and then populated from W_l using the condition

$$max(W_l^m) < t_{th} \ : \forall \ 1 \leq m \leq M. \tag{9}$$

Hence, $W_l^p \subseteq W_l$, and the set of remaining filters after pruning layer l is denoted $W_l^r = W \backslash W_l^p$. Similar procedure can be repeated for all weight layers so that the overall resulting trimmed model is now parameterized by W^r as in $W^r = \{W_1^r, ..., W_l^r, ..., W_L^r\}$.

4.1.3 Debiasing elastic group LASSO

To tackle the problem of entangled feature selection and shrinkage discussed in Section 3.2.3, we completely eliminate group LASSO from the cost function given in (10) by setting $\gamma = 0$, and retraining the trimmed model now using the cost function, $J_{degl}(W_r)$, given as

$$J_{degl}(W_r) = \underset{W_r}{\operatorname{argmin}} \left\{ J(W_r) + \lambda_{rt} \sum_{l=1}^{L} \| W_l^r \|_2^2 \right\}, \tag{10}$$

where λ_{rt} denotes the new $l2$-norm penalty weight for retraining; and $\lambda_{rt} = s\lambda : 0 < s \leq 1$, since $W_r \subseteq W$; s is chosen considering the size of W^r. The new cost in (10) is similar in spirit to LARS-OLS [4] and relaxed LASSO [20].

5. Evaluation metrics

The different metrics used for evaluating the performance of the proposed approach are discussed as follows.

5.1. Model parameters

We consider a DNN with L weight layers, out of which L_{cnv} are convolution layers. Given a convolution weight layer $W_l \in \mathbb{R}^{f_l^w \times f_l^h \times c_{l-1}^n \times c_l^n}$ that receives an input $H_{l-1} \in \mathbb{R}^{b_s \times c_{l-1}^w \times c_{l-1}^h \times c_{l-1}^n}$ and output $H_l \in \mathbb{R}^{b_s \times c_l^w \times c_l^h \times c_l^n}$, where w_l^f, h_l^f, c_{l-1}^n, c_l^n are the filter width, filter height, number of input feature channels and output feature channels, respectively; b_s, c_{l-1}^w, c_{l-1}^h are the batch size, incoming feature channel width and height, respectively. The number of parameters in a convolution layer l, θ_l^{cnv}, can be obtained using

$$\theta_l^{cnv} = f_l^w f_l^h c_{l-1}^n c_l^n. \tag{11}$$

For a fully connected layer l, the hyperparameters f_l^w, f_l^h, c_{l-1}^w, c_{l-1}^h, c_{l-1}^n, c_l^w, c_l^h, c_l^n can all be set to 1, so that the number of parameters, θ_l^{fc}, is given as

$$\theta_l^{fc} = c_{l-1}^n c_l^n. \tag{12}$$

Hence, the total number of model parameters, θ_t, is

$$\theta_t = \sum_{l=1}^{L_{cnv}} \theta_l^{cnv} + \sum_{l=1}^{L-L_{cnv}} \theta_l^{fc}. \quad (13)$$

5.2. Float point operations per second (FLOPS)

Required Float point operations per second (FLOPS) is a very useful criterion for assessing how fast a model runs. The number of FLOPS for a convolution layer l, $FLOPS_{cnv}^l$, can be obtained using

$$FLOPS_l^{cnv} = b_s f_l^w f_l^h c_{l-1}^n c_l^w c_l^h c_l^n. \quad (14)$$

Similarly, the number of FLOPS for a fully connected layer l, referred to as $FLOPS_l^{fc}$, can be computed as

$$FLOPS_l^{fc} = b_s c_{l-1}^n c_l^n. \quad (15)$$

Thus, the total number of FLOPS for the model, $FLOPS_t$, is given as

$$FLOPS_t = \sum_{l=1}^{L_{cnv}} FLOPS_l^{cnv} + \sum_{l=1}^{L-L_{cnv}} FLOPS_l^{fc}. \quad (16)$$

5.3. Computational memory footprint (CMF)

Computational memory footprints (CMFs) of compression results are quite critical, as it describes the memory required to hold model parameters and units' activations; small CMFs are generally desirable. At a given layer l, the CMF for a convolution layer and a fully connected layer can be computed as CMF_l^{cnv} and CMF_l^{fc}, respectively,

$$CMF_l^{cnv} = (Q/8)(\theta_l^{cnv} + b_s c_l^w c_l^h c_l^n), \quad (17)$$

and

$$CMF_l^{fc} = (Q/8)(\theta_l^{fc} + b_s c_l^n), \quad (18)$$

where Q is the number of bits of precision as in 'float$Q$ '. The total CMF for the model, CMF^t, is

$$CMF_t = \sum_{l=1}^{L_{cnv}} CMF_l^{cnv} + \sum_{l=1}^{L-L_{cnv}} CMF_l^{fc}. \quad (19)$$

5.4. Model memory size (MMS)

The compression results are also evaluated in view of the required memory storage, since one of the critical objectives of model compression is to obtain models that require considerably less memory for storage. MMS is reported in megabytes (MB) or kilobytes (KB) in this paper.

6. Experiments

6.1. Main experiments

We perform extensive experiments using six benchmarking datasets. The results of our proposed DEGL are compared to conventional group LASSO and several state-of-the-art methods. Note that no comparison is made with $l2$-norm penalty and LASSO since they do not on their own yield structured sparsity for compression.

We train the different models in this paper using mini-batch gradient descent; batch size for all experiments is chosen in the range 256-512. The initial learning rate, l_r for all experiments are chosen in the range 0.001 to 0.1; l_r is reduced by a factor of 0.1 during training whenever the training loss fails to improve for 5 epochs. Different suitable values for γ and λ in (10) are used to enforce the desired sparsity levels and grouping effects respectively in the models. For pruning, different values of t_{th} (as in (9)) are used to obtain different levels of compression; larger values for t_{th} translate to more compact models. Note that DEGL and group LASSO are both retrained after pruning. For all results, 'Error $\uparrow$ ', 'FLOPS $\downarrow$ ' and 'Param. $\downarrow$ ' denote error rate increase, FLOP percentage decrease and Parameters percentage decrease, respectively. Error $\uparrow$ is calculated as in

$$Error\uparrow \ = Error_{ac} - Error_{bc}, \quad (20)$$

where $Error_{ac}$ and $Error_{bc}$ are the error rates after and before compression, respectively. *Thus, a negative value for 'Error $\uparrow$ ' actually shows a decrease in error rate after compression*. The evaluated metrics for the reference (i.e. uncompressed) models are appended at the end of each table.

6.2. Ablation studies

As ablation studies, we observe the performance loss of elastic group LASSO without model debaising. That is, after pruning, we employ the same cost objective in (6) for model retraining. Results of these experiments from Elastic group LASSO without the debiasing step are denoted (EGL), and are given along side DEGL and group LASSO for all experiments. In addition, the impact of pruning threshold values and group LASSO penalty weight hyper-parameter, γ in (6) & (7), on model performance loss are studied; see supplemtary material for results, including the training times for DEGL, group LASSO and EGL.

6.3. Results

6.3.1 LeNet-5 on MNIST

The MNIST[2] dataset contains 60K training and 10K testing samples, respectively. LeNet-5 [6] is a DNN with two convolution layers and three fully connected weight layers including the softmax layer; it has 431K parameters. LeNet-5 is trained for 100 epochs with $\gamma = 5 \times 10^{-3}$ and $\lambda = 10^{-4}$. Table 1 shows the compression results of LetNet-5, and comparison with state-the-of-art methods. DEGL1, DEGL2, DEGL3, DEGL4, DEGL5 are obtained by setting the pruning threshold values, t_{th} : 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2} and 2.5×10^{-2}, respectively. Figure 2 shows the CMF and MMS of compressed models using DEGL. For example, DEGL5 with 98.4% parameters pruned reduces CMF from 57MB to 27.58MB and

[2]http://yann.lecun.com/exdb/mnist/

Models	Error ↑	FLOPS ↓	Param. ↓
LTPWC [6]	-0.03%	83.7%	91.7%
AFP [3]	-0.06%	93.3%	93.0%
TSN [32]	0.01%	–	95.8%
LiM [43]	0.08%	–	98.4%
Group-LASSO [3]	0.00%	87.0%	86.4%
Group-LASSO [3]	0.20%	93.7%	93.4%
EGL	0.05%	96.1%	94.1%
EGL	0.14%	98.5%	98.4%
Ours: DEGL1	**-0.10%**	**96.1%**	**94.1%**
Ours: DEGL2	**-0.03%**	**96.9%**	**95.3%**
Ours: DEGL3	**-0.15%**	**97.9%**	**96.9%**
Ours: DEGL4	**-0.07%**	**98.3%**	**97.5%**
Ours: DEGL5	**0.01%**	**98.5%**	**98.4%**

Reference LeNet-5: Error = 0.80%, Param. = 431K, FLOPS = 4.6M

Table 1: LeNet-5 compression results on MNIST dataset

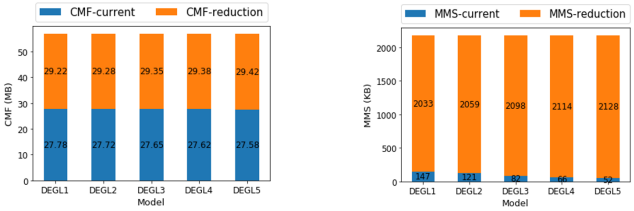

Figure 2: LeNet-5 CMF and MMS results on MNIST

Models	Error ↑	FLOPS ↓	Param. ↓
AFP [3]	0.38%	81.4%	–
AFP [3]	0.31%	79.6%	–
Pruning [16]	-0.15%	34.2%	64.0%
L2PF [11]	1.90%	64.5%	86.5%
Group-LASSO	0.08%	86.9%	87.8%
Group-LASSO	0.23%	88.1%	90.0%
EGL	0.03%	87.0%	88.0%
EGL	0.31%	88.1%	90.0%
Ours: DEGL1	**-0.30%**	**85.6%**	**86.7%**
Ours: DEGL2	**-0.30%**	**87.0%**	**88.0%**
Ours: DEGL3	**-0.18%**	**88.1%**	**90.0%**

Reference VGG-16: Error = 6.75%, Param. = 15M, FLOPS = 313M

Table 2: VGG-16 compression results on CIFAR-10 dataset

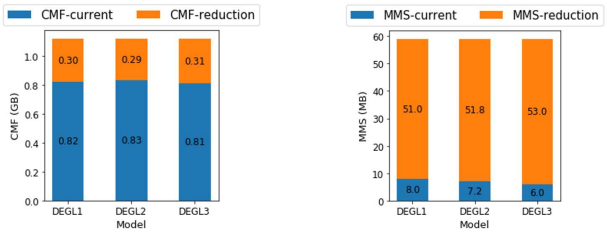

Figure 3: VGG-16 CMF and MMS results on CIFAR-10

MMS from 2180KB to 52KB, while achieving an impressive error of 0.73%. It is seen that EGL performs better than group-LASSO, but worse than DEGL.

6.3.2 VGG-16 and ResNet-56 on CIFAR-10

CIFAR-10[3] dataset contains 50K training and 10K testing images, respectively; there are 10 different classes. VGG-16 [3] model has 15M parametres that consist of 13 convolution layers and 2 fully connected weight layers. We apply DEGL to VGG-16 for 300 epochs using $\gamma = 1 \times 10^{-5}$ and $\lambda = 10^{-6}$. Compression results are given in Table 2, where DEGL1, DEGL2 and DEGL3 are obtained by setting t_{th} : 10^{-4}, 10^{-3}, and 10^{-2}, respectively. Figure 3 shows the CMFs and MMMs for DEGL1, DEGL2 and DEGL3 in Table 2. For example, DEGL3 with 90.0% parameters pruned reduces CMF from 1.12GB to 0.81GB, and MMS from 59MB to 6MB, while even improving generalization. Figure 4 shows the number of current and pruned filters (or units) for DEGL2.

[3]https://www.cs.toronto.edu/ kriz/cifar.html

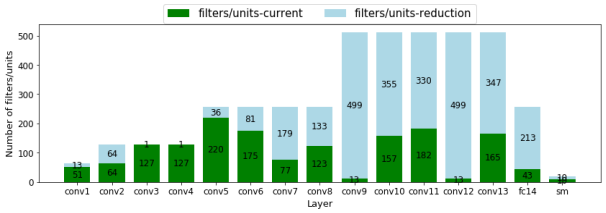

Figure 4: DEGL2 compression results for VGG-16 on CIFAR-10

Models	Error ↑	FLOPS ↓	Param. ↓
Pruning [16]	-0.02%	27.6%	13.7%
NISP [38]	0.03%	43.6%	42.6%
CNN-FCF [17]	-0.24%	42.78%	43.1%
KSE [23]	0.15%	60%	57.6%
Group-LASSO	0.03%	37.9%	44.3%
Group-LASSO	0.24%	47.1%	59.4%
EGL	0.05%	47.8%	44.7%
EGL	0.48%	53.4%	60.0%
Ours: DEGL1	**-0.37%**	**38.2%**	**31.8%**
Ours: DEGL2	**-0.31%**	**47.8%**	**44.7%**
Ours: DEGL3	**0.09%**	**53.4%**	**60.0%**

Reference ResNet-56: Error = 6.96%, Param. = 0.85M, FLOPS = 125M

Table 3: ResNet-56 compression results on CIFAR-10 dataset

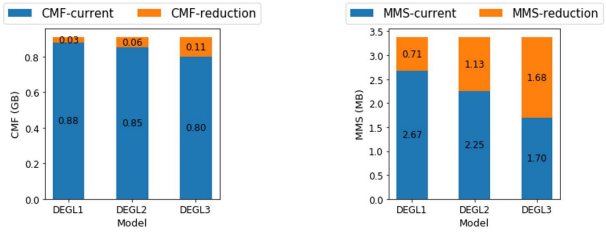

Figure 5: ResNet-56 CMF and MMS results on CIFAR-10

Results of DEGL using ResNet-56 on CIFAR-10 is given in Table 3, where DEGL1, DEGL2 and DEGL3 are obtained using the same t_{th} values as in VGG-16. We particularly observe that DEGL outperforms conventional group LASSO for ResNet architectures, where skip connections can increase features correlations among different layers, and therefore conventional group LASSO is struggle with consistent feature selection; see Section 3.2.2 for details. Figure 5 shows how pruning impacts CMF and MMS for models reported in Table 3. Importantly, for both VGG-16 and ResNet-56, DEGL outperforms both group-LASSO

Models	Error ↑	FLOPS ↓	Param. ↓
Group-LASSO	0.07%	84.0%	83.3%
Group-LASSO	0.11%	84.2%	85.6%
EGL	0.06%	84.5%	84.0%
EGL	0.14%	84.7%	86.0%
Ours: DEGL1	**-0.01%**	**81.7%**	**82.0%**
Ours: DEGL2	**0.00%**	**84.5%**	**84.0%**
Ours: DEGL3	**0.03%**	**84.7%**	**86.0%**

Reference VGG-16: Error = 27.41%, Param. = 15M, FLOPS = 313M

Table 4: VGG-16 compression results on CIFAR-100 dataset

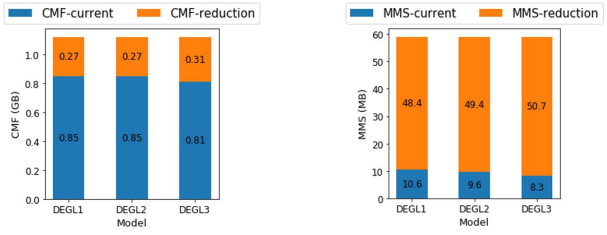

Figure 6: VGG-16 CMF and MMS results on CIFAR-100

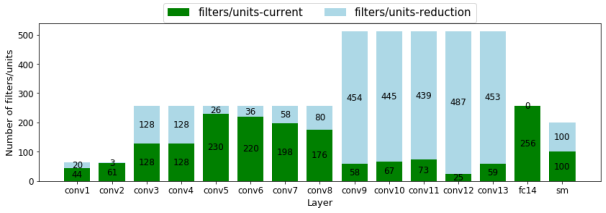

Figure 7: DEGL3 compression results for VGG-16 on CIFAR-100

and EGL.

6.3.3 VGG-16 and ResNet-56 on CIFAR-100

CIFAR-100[3] dataset contains 50K training and 10K testing samples, respectively; the dataset composes 100 different classes. The same VGG-16 architecture used for CIFAR-10 is used for CIFAR-100; the model is trained for 350 epochs with $\gamma = 1 \times 10^{-5}$ and $\lambda = 10^{-4}$. Table 4 shows pruning results, where DEGL1, DEGL2 and DEGL3 are obtained by setting t_{th} : 10^{-4}, 5×10^{-3}, and 10^{-2}, respectively. As CIFAR-100 is a more challenging dataset, and the effectiveness of DEGL compared to group LASSO and EGL clearly reflects. The impact of compression on CMF and MMS is reported in Figure 6. Figure 7 shows the current and pruned filters (or units) for DEGL3.

The compression results for ResNet-56 are given in the supplementary material as Table A1 and Figure A1, along with discussion in Section A1. ResNet-56 results are similar to those obtained for VGG-16; that is, DEGL outperforms both group-LASSO and EGL.

Models	Error ↑	FLOPS ↓	Param. ↓
LiM [43]	0.57%	–	76.8%
NISP [38]	0.00%	40.1%	47.1%
DFP [31]	4.08%	–	45.8%
Group-LASSO	0.81%	31.4%	53.1%
Group-LASSO	1.37%	44.2%	66.7%
Group-LASSO	2.41%	51.6%	78.5%
EGL	1.10%	40.2%	57.4%
EGL	1.68%	44.4%	67.6%
EGL	2.84%	52.3%	79.3%
Ours: DEGL1	**-0.02%**	**40.2%**	**57.4%**
Ours: DEGL2	**0.25%**	**42.1%**	**62.0%**
Ours: DEGL3	**0.31%**	**44.4%**	**67.6%**
Ours: DEGL4	**0.43%**	**52.3%**	**79.3%**

Reference AlexNet: Error = 41.81%, Param. = 61M, FLOPS = 727M

Table 5: AlexNet compression results on ImageNet dataset

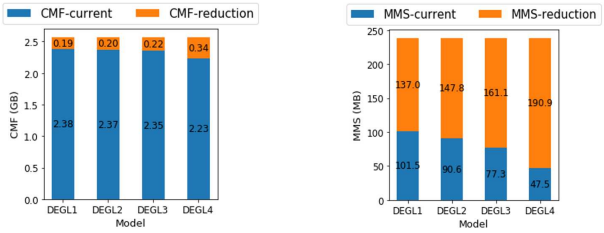

Figure 8: AlexNet CMF and MMS results on ImageNet

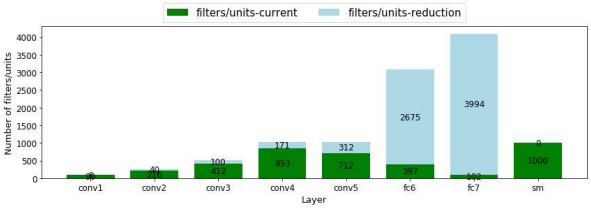

Figure 9: DEGL4 compression results for AlexNet on ImageNet

6.3.4 AlexNet and ResNet-50 on ImageNet

The ImageNet dataset[4] contains about 1.2M training and 50K testing samples, respectively. The dataset has 1000 different categories. AlexNet has 61M parameters that consist of 5 convolution layers and 3 fully connected weight layers including the softmax layer. The model is trained for 70 epochs with $\gamma = 5 \times 10^{-6}$ and $\lambda = 10^{-4}$. ResNet-50 has 25.6M parameters, and the same values of γ and λ used for AlexNet are employed for training. ResNet50 is trained for 90 epochs, since it is much deeper than AlexNet.

The results of using the proposed DEGL for compressing AlexNet are given in Table 5, where DEGL1, DEGL2 and DEGL3 are obtained by setting t_{th} : 10^{-4}, 5×10^{-3}, and 5×10^{-2}, respectively; similar values are used for ResNet experiments. It is seen that based on achieved error rates, DEGL4 with 67.6% of model parameters pruned incurs no performance loss, while with group LASSO with 66.7% of model parameters pruned and EGL with 67.6% of model parameters pruned incur performance loss of 0.17%

[4]http://image-net.org/challenges/LSVRC/2012/index

Models	Error ↑	FLOPS ↓	Param. ↓
ThiNet-50 [19]	1.15%	36.8%	33.8%
ThiNet-30 [19]	6.12%	71.5%	66.1%
NIPS [38]	0.89%	44.0%	43.8%
DCP [45]	1.06%	55.6%	51.5%
CNN-FCF [17]	0.47%	46.1%	42.4%
CNN-FCF [17]	2.62%	66.2%	61.0%
KSE [23]	0.84%	78.5%	65.8%
Group-LASSO	1.18%	44.3%	43.5%
Group-LASSO	1.55%	59.2%	54.1%
EGL	1.06%	47.6%	44.4%
EGL	1.79%	59.3%	54.4%
Ours: DEGL1	**0.11%**	**47.6%**	**44.4%**
Ours: DEGL2	**0.23%**	**59.3%**	**54.4%**
Ours: DEGL3	**0.52%**	**79.2%**	**67.0%**

Reference ResNet-50: Error = 24.02%, Param. = 25.06M, FLOPS = 4.1B

Table 6: ResNet-50 compression results on ImageNet dataset

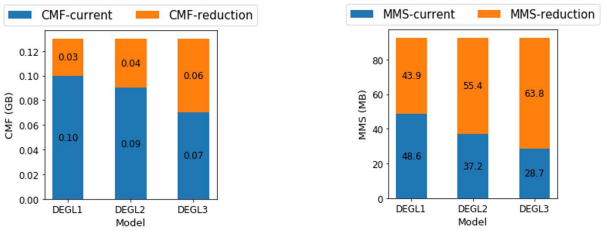

Figure 10: ResNet50 CMF and MMS results on ImageNet

and 0.28%, respectively. It is observed that further compression using group LASSO or EGL leads to larger performance loss. The current CMF and MMS along with the reductions after compression for the DEGL models (given in Table 5) are shown in Figure 8. For instance, DEGL4 reduces FLOPS from 727M to 347M, CMF from 2.57GB to 2.23GB and MMS 238.4MB to 47.5MB, with a test error increase of only 0.07%. In addition, the current number of filters or units in the different layers of DEGL2 are reported in Figure 9.

Compression results for ResNet-50 are given in Table 6. DEGL2 with 54.4% of model parameters pruned clearly outperforms group-LASSO with 39.6% of model parameters pruned, EGL with 54.4% of model parameters pruned and several state-of-the-art methods. Further compression leads to DEGL3 with 67.0% of model parameter pruned; DEGL3 is competitive with group-LASSO with 39.6% of model parameters pruned and EGL with 38.6%. The current CMF and MMS along with the reductions after compression for the DEGL models (given in Table 6) are presented in Figure 10.

6.3.5 Model compression for domain adaptation

We consider the effectiveness of our compression method for domain adaptation using Oxford-102 [21] flower species datasets and Food-5k [30] datasets, which contain 102 different flower species and food/non-food images, respectively. Oxford-102 flower species dataset has 6149, 1020 and 1020 training, validation and testing samples, respectively; for this dataset, we follow the training and testing protocols in [36]. Food-5K dataset contains 2500 food images and 2500 non-food images; the training and testing protocols in [30] are employed for this dataset. Namely, we evaluate the original pre-trained and compressed models for performance loss. Our compression results using pre-trained ResNet-50 are given in Table 7, where DEGL1 and DEGL2 are obtained by setting t_{th} : 5×10^{-3} and 5×10^{-4}, respectively. All models are trained using the same hyperparameters to support fair comparison. It is seen that the proposed DEGL models significantly outpeform Group-LASSO and EGL models. As such, DEGL is well-suited for domain adaptation tasks.

Dataset	Models	Error ↑	FLOPS ↓	Param. ↓
Oxford-102	Group-LASSO	0.93%	36.9%	34.7%
	Group-LASSO	1.26%	59.5%	55.2%
	EGL	0.89%	44.4%	43.9%
	EGL	1.52%	68.0%	62.9%
	Ours: DEGL1	**0.01%**	**44.4%**	**43.9%**
	Ours: DEGL2	**0.22%**	**68.0%**	**62.9%**
Food-5K	Group-LASSO	1.18%	39.9%	39.4%
	Group-LASSO	2.29%	69.5%	64.6%
	EGL	1.35%	42.1%	40.2%
	EGL	2.62%	69.7%	65.8%
	Ours: DEGL1	**0.53%**	**42.1%**	**40.2%**
	Ours: DEGL2	**1.31%**	**69.7%**	**65.8%**

Oxford-102 ResNet-50: Error = 3.43%, Param. = 25.06M, FLOPS = 4.1B
Food-5K ResNet-50: Error = 0.50%, Param. = 25.06M, FLOPS = 4.1B

Table 7: ImageNet pre-trained ResNet-50 compression results

7. Conclusion

In this paper, we address structured compression of deep neural networks by taking inspiration from sparse signal processing. Directly motivated by the drawbacks of applying conventional group LASSO for compressing deep neural networks (DNNs), we propose 'debiased elastic group lasso (DEGL)' that is applied to several state-of-the-art DNNs using six benchmarking datasets. Compression results are evaluated based on the percentage of pruned parameters, FLOPS, computational memory footprint and memory for storage. Results of extensive compression experiments show that DEGL performs better than conventional group LASSO and several state-of-the-art methods. Overall, the results reveal that considerable model compression can be achieved using DEGL with little performance loss or even improved generalization.

Acknowledgments

This work was funded by the National Research Fund (FNR), Luxembourg, under the project reference R-AGR-0424-05-D/Björn Ottersten and and CPPP17/IS/11643091/IDform/Aouada. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

References

[1] S. Anwar, K. Hwang, and W. Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):32, 2017.

[2] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, and D. Batra. Reducing overfitting in deep networks by decorrelating representations. In *International Conference on Learning Representations(ICLR)*, 2016.

[3] X. Ding, G. Ding, J. Han, and S. Tang. Auto-balanced filter pruning for efficient convolutional neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[4] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[5] Y. Guo, A. Yao, and Y. Chen. Dynamic network surgery for efficient dnns. In *Advances In Neural Information Processing Systems*, pages 1379–1387, 2016.

[6] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[7] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2234–2240. AAAI Press, 2018.

[8] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[9] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[10] J. Huang, P. Breheny, and S. Ma. A selective review of group selection in high-dimensional models. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 27(4), 2012.

[11] Q. Huang, K. Zhou, S. You, and U. Neumann. Learning to prune filters in convolutional neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 709–718. IEEE, 2018.

[12] V. Jayasundara, S. Jayasekara, H. Jayasekara, J. Rajasegaran, S. Seneviratne, and R. Rodrigo. Textcaps: Handwritten character recognition with very small datasets. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 254–262. IEEE, 2019.

[13] M. M. Kabir, M. M. Islam, and K. Murase. A new wrapper feature selection approach using neural network. *Neurocomputing*, 73(16-18):3273–3283, 2010.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] D. Li, X. Wang, and D. Kong. Deeprebirth: Accelerating deep neural network execution on mobile devices. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[16] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations(ICLR)*, 2017.

[17] T. Li, B. Wu, Y. Yang, Y. Fan, Y. Zhang, and W. Liu. Compressing convolutional neural networks via factorized convolutional filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3977–3986, 2019.

[18] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814, 2015.

[19] J.-H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[20] N. Meinshausen. Relaxed lasso. *Computational Statistics & Data Analysis*, 52(1):374–393, 2007.

[21] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[22] O. K. Oyedotun, G. Demisse, A. El Rahman Shabayek, D. Aouada, and B. Ottersten. Facial expression recognition via joint deep learning of rgb-depth map latent representations. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3161–3168, 2017.

[23] O. K. Oyedotun, A. El Rahman Shabayek, D. Aouada, and B. Ottersten. Highway network block with gates constraints for training very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1658–1667, 2018.

[24] T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247(4945):978–982, 1990.

[25] W. Ren, L. Ma, J. Zhang, J. Pan, X. Cao, W. Liu, and M.-H. Yang. Gated fusion network for single image dehazing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3253–3261, 2018.

[26] S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5(Aug):941–973, 2004.

[27] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations(ICLR)*, 2015.

[29] P. Singh, V. S. R. Kadi, N. Verma, and V. P. Namboodiri. Stability based filter pruning for accelerating deep cnns. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1166–1174. IEEE, 2019.

[30] A. Singla, L. Yuan, and T. Ebrahimi. Food/non-food image classification and food categorization using pre-trained

googlenet model. In *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management*, pages 3–11. ACM, 2016.

[31] S. Srinivas and R. V. Babu. Data-free parameter pruning for deep neural networks. In *British Machine Vision Conference (BMVC)*, 2015.

[32] S. Srinivas, A. Subramanya, and R. Venkatesh Babu. Training sparse neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 138–145, 2017.

[33] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[34] H. Wang, B. J. Lengerich, B. Aragam, and E. P. Xing. Precision lasso: accounting for correlations and linear dependencies in high-dimensional genomic data. *Bioinformatics*, 2018.

[35] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016.

[36] Y. Wu, X. Qin, Y. Pan, and C. Yuan. Convolution neural network based transfer learning for classification of flowers. In *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP)*, pages 562–566. IEEE, 2018.

[37] H. Xu, C. Caramanis, and S. Mannor. Sparse algorithms are not stable: A no-free-lunch theorem. *IEEE transactions on pattern analysis and machine intelligence*, 34(1):187–193, 2011.

[38] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018.

[39] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[40] M. Yunus, A. Saefuddin, and A. M. Soleh. Characteristics of group lasso in handling high correlated data. *Applied Mathematical Sciences*, 11(20):953–961, 2017.

[41] S. Zagoruyko and N. Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, volume 8, pages 35–67, 2012.

[42] P. Zhao and B. Yu. On model selection consistency of lasso. *Journal of Machine learning research*, 7(Nov):2541–2563, 2006.

[43] H. Zhou, J. M. Alvarez, and F. Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pages 662–677. Springer, 2016.

[44] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 875–886, 2018.

[45] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 875–886, 2018.

[46] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.