



**HAL**  
open science

# Learning Sparse deep neural networks using efficient structured projections on convex constraints for green AI

Michel Barlaud, Frederic Guyard

► **To cite this version:**

Michel Barlaud, Frederic Guyard. Learning Sparse deep neural networks using efficient structured projections on convex constraints for green AI. 2020. hal-02556382v1

**HAL Id: hal-02556382**

**<https://hal.science/hal-02556382v1>**

Preprint submitted on 28 Apr 2020 (v1), last revised 28 Oct 2020 (v3)

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Learning Sparse deep neural networks using efficient structured projections on convex constraints for green AI

Michel Barlaud  
Laboratoire I3S  
Cote d'Azur University  
Sophia Antipolis, France  
Email: michel.barlaud@i3s.unice.fr

Frederic Guyard  
Orange Labs  
Sophia Antipolis, France  
Email: frederic.guyard@orange.com

**Abstract**—In recent years, deep neural networks (DNN) have been applied to different domains and achieved dramatic performance improvements over state-of-the-art classical methods. These performances of DNNs were however often obtained with networks containing millions of parameters and which training required heavy computational power. In order to cope with this computational issue a huge literature deals with proximal regularization methods which are time consuming. In this paper, we propose instead a constrained approach. We provide the general framework for our new splitting projection gradient method. Our splitting algorithm iterates a gradient step and a projection on convex constraints. We study algorithms for different constraints: the classical  $\ell_1$  unstructured constraint and structured constraints such as the nuclear norm, the  $\ell_{2,1}$  constraint (Group LASSO). We propose a new  $\ell_{1,1}$  structured constraint for which we provide a new projection algorithm. Finally we use the recent "Lottery optimizer" replacing the thresholding by our  $\ell_{1,1}$  projection. We demonstrate the effectiveness of our method on three popular datasets (MNIST, Fashion MNIST and CIFAR). Experiments on these datasets show that our projection method with our new  $\ell_{1,1}$  structured constraint provides the best reduction of memory and computational power. Experiments show that fully connected linear DNN are more efficient for memory and MACCs reduction thus for green AI.

## I. MOTIVATION

In recent years, deep neural networks have been applied to different domains and achieved dramatic accuracy improvements in image recognition [33], speech recognition [44] or natural language processing [46]. These works rely on deep networks with millions or even billions of parameters. For instance, the original training of ResNet-50 [26] (image classification) contains 25.6M parameters and required 29 hours using 8 GPUs. Storing the model requires 98MB. The memory cost of the inference on a single 224x224 image is about 103MB and 4 GFLOPs are needed [6]. The recent development of DNNs hardware accelerators like GPUs and the availability of deep learning frameworks for smartphones [30] suggest seamless transfer of DNNs models trained on servers onto mobile devices. However, it turns out that memory [48] and energy consumption [21] are still the main bottlenecks on running DNNs on such devices. Thus computational cost has an impact on carbon footprint.

The authors of [53] argued that this trend is environmentally unfriendly. The authors of [50] advocate a practical solution by making an efficient evaluation criterion.

In this paper, we propose a new splitting projection-gradient method with an efficient structured constraint to cope with these computational and memory issues. In the formulation of our method, a constraint defines a convex set and the regularization is replaced by a projection onto this convex set. The benefits of this formulation are twofold. The constraint has a direct geometric interpretation whereas the impact of parameter values in traditional regularization methods are more difficult to understand. Furthermore, the convergence of this new method is formally proved.

The paper is organized as follows. We first present related works in Section II, then we develop in Section III the theoretical background of our constrained projection method. In Section IV, we give experimental comparisons between methods. The tests involve several datasets with different neural network architectures.

## II. RELATED WORKS

### *Weights sparsification*

It is well known [12] that DNN models are largely over-parametrized and that in practice, relatively few network weights are actually necessary to learn accurately data features. Based on this result, numerous methods have been proposed in order to remove network weights (*weight sparsification*) either on pre-trained models or during the training phase. A basic idea to sparsify the weights of the neural network is to use the *Least Absolute Shrinkage and Selection Operator* (LASSO) formulation [55], [24], [17], [25], [1]. The  $\ell_1$  penalty added to the classification cost can be interpreted as a convexification of the  $\ell_0$  penalty [13]. In [22], weights with the smallest amplitude in pre-trained networks are removed. Model sensitivity to weights can also be used [54], [19], where weights with weak influence on network output are pruned. Constraint optimization is used in [7] in order to learning sparse networks with  $\ell_0$ ,  $\ell_1$  or  $\ell_2$  constraints on the weights.

These methods generally produce networks with random sparse connectivity, i.e. high-dimensional but sparse weight matrices. They reduce only partially the computational demand since they result in networks with sparse weight matrices, requiring the availability of sparse matrix multiplication to effectively take advantage of the sparsity. Decreasing both memory and computation requirements can however be achieved by suppressing neurons instead of weights. This approach is frequently referred to as *structured sparsification* or *neuron level sparsification*. The two main approaches for structured sparsity are based on group regularization and low-rank factorization.

Many regularizing techniques have been proposed to allow structured sparsification. Pruning methods are sparsifying pre-trained networks. For instance, in [35] filters in CNN are pruned based on the  $\ell_1$  norm of their kernel weights, [27] performs channel pruning using LASSO regression and least squared reconstruction or [28] where neurons are pruned based on the average percentage of zeros (APoZ) after the ReLU activation.

#### *Learning structured sparse DNNs using regularization methods*

In contrast to the case of weight sparsification, neuron level sparsification introduces a new challenge forcing to adopt other types of regularization.

The most common approaches are based on *group LASSO*  $\ell_{2,1}$  [60] or on *sparse group LASSO*  $\ell_{2,1} + \ell_1$  [17] regularization.

Numerous other methods include regularization during the training of the DNN. It is customary in DNN learning to train networks with Stochastic Gradient Descent (SGD) with momentum, even in the case where non-smooth penalization are used [49]. This is for instance the case in [57] where group LASSO regularization is used, in [49] using sparse group LASSO [18], in [67] where group LASSO and filter decorrelation regularization are used in order to discard CNN filters or in [56] where group LASSO and group variance regularization are used.

In order to deal with non-smooth  $\ell_1$  regularization, subgradient descent is used in [39]. Here, structured sparsification is performed without group regularization. The idea is to scale neurons output with a given factor  $\lambda_i$  and apply  $\ell_1$  regularization to push the various factors  $\lambda_i$  towards 0.

Fully connected layers can be represented by their weights matrix (i.e. 2d tensor) whereas convolutional layers correspond to 4d tensors. One of the popular compression methods for DNN is nuclear regularization (Nuclear norm penalty) [9]. Nuclear norm penalty was successfully used in matrix low rank approximation [9], matrix completion [62], matrix factorization [8] and DNN dropout modeling [52].

#### *Learning structured sparse DNNs using proximal regularization methods*

A different approach is however based on optimization under convex constraint where proximal methods are the most natural

tools. Let's recall the proximal operator of a function  $f(x)$  [42]:

$$\text{prox}_{\tau f}(\bar{x}) := \arg \min_x f(x) + \frac{\|x - \bar{x}\|^2}{2\tau}, \quad (1)$$

Let  $W$  be the weight matrix of a neural network,  $L(W)$  be a gradient Lipschitz loss and  $R(w)$  be a convex penalty. Let define the penalty criterion by  $\min_W L(W) + \lambda R(W)$ . This criterion can be minimized using a classical forward-backward method belonging to the class of splitting methods [37], [10], [43], [51]. Using SGD with penalization is limited and time consuming due to the tuning of the corresponding penalization hyper-parameters [24], [40].

Proximal gradient descent with group LASSO constraint is used in [65] and in [2]. In [29], the output neurons is scaled using a factor  $\lambda_i$  and accelerated proximal gradient is used with  $\ell_1$  constraint to push as many coefficients  $\lambda_i$  towards 0 without decreasing significantly the performance. In [59], neuron sparsification is realized with proximal gradient descent with group LASSO constraint and an additional  $\ell_{1,2}$  constraint (*Exclusive Sparsity*) enforces neurons to fit disjoint sets of features. Similarly, in [64] proximal gradient descent with group OWL constraint (grOWL [45]) is used in order to simultaneously sparsify neurons and enforce parameter sharing. Proximal gradient descent is also used in [63] where Ordered Weighted  $\ell_1$  regularization (OWL [15]) allowing simulatenously sparsifying weights and optimizing weight sharing. In [36], filters in CNN layers are pruned by solving an optimization problem using a dedicated optimizer with either group LASSO or  $\ell_{2,0}$  regularization.

#### *Goal of the work*

Classical Learning structured sparse DNNs are based on proximal regularization methods; In this paper, we propose instead a constrained approach that takes advantage of an available efficient projection on the  $\ell_1$ -ball [11], [14], [47], projection on the  $\ell_{2,1}$  ball [38], [5] and a new  $\ell_{1,1}$  projection proposed in this paper.

### III. LEARNING SPARSE DNN

#### *A Projection gradient algorithm for constrained learning*

In this work, we propose a constrained approach where the constraint is directly related to the number of zero-weights. Moreover it takes advantage of an available efficient projection on the  $\ell_1$ -ball [11], [47].

Let  $L(W)$  be a gradient Lipschitz loss,  $R(w)$  be a convex constraint, and  $C$  its convex set. Let define the following criterion

$$\min_W L(W) \quad \text{s.t.} \quad R(W) \leq \eta \quad (2)$$

where the scalar  $\eta \geq 0$  is the constraint parameter. We use a splitting gradient-projection method to minimize this criterion

based on the following forward-backward scheme to generate a sequence of iterates [3]:

$$V_n := W_n - \gamma \nabla L(W_n), \quad (3)$$

$$W_{n+1} := \text{proj}(V_n) + \varepsilon_n, \quad (4)$$

where  $\text{proj}$  denotes the projection on the convex constraint. We can therefore apply the algorithm to any constraint which exact or approximate projection can be computed. We derive the following algorithm

---

**Algorithm 1** Splitting gradient-projection algorithm where  $\nabla L(W)$  is provided by the net and  $\text{proj}(\eta V)$  is the projection on the constraint

---

**Input:**  $X, Y, W_0, N, \gamma, \eta$

**for**  $n = 1, \dots, N$  **do**

$$V \leftarrow W - \gamma \cdot \nabla L(W)$$

$$W \leftarrow \text{proj}(\eta V)$$

**end for**

**Output:**  $W$

---

### Optimizer with structured constraints

In the case where the constraint  $R(w) = \|w_i\|_1$  efficient algorithms have been proposed [11], [47]. Unfortunately this  $\ell_1$  constraint does not induce sparsity structure.

*Classical Optimizer with  $\ell_{2,1}$  norm constraint (Group LASSO):* The Group LASSO was first introduced in [61]. The main idea of Group LASSO is to enforce models parameters for different classes to share features. Group sparsity reduce complexity by eliminating entire features. Group LASSO consists in using the  $\ell_{2,1}$  norm for the constraint on  $W$ . The row-wise  $\ell_{2,1}$  norm of a  $d \times k$  matrix  $W$  (whose rows are denoted  $w_i, i = 1, d$ ) is defined as follows:

$$\|W\|_{2,1} := \sum_{i=1}^d \|w_i\|.$$

We use the following approach proposed in [5] to compute the projection  $W$  of a  $d \times k$  matrix  $V$  (whose rows are denoted  $v_i, i = 1, d$ ) on the  $\ell_{2,1}$ -ball of radius  $\eta$ : compute  $t_i$  which is the projection of the vector  $(\|v_i\|_i)_{i=1}^d$  on the  $\ell_1$  ball of  $\mathbf{R}^n$  of radius  $\eta$ ; then, each row of the projection is obtained according to

$$w_i = \frac{t_i v_i}{\max\{t_i, \|v_i\|\}}, \quad i = 1, \dots, d.$$

This last operation is denoted as  $W_i := \text{proj}_{\ell_2}(V_i, t_i)$  in Algorithm 2.

This algorithm requires the projection projection of the vector  $(\|v_i\|_i)_{i=1}^d$  on the  $\ell_1$  ball of  $\mathbf{R}^n$  of radius  $\eta$  whose complexity is only  $O(d \times \log(d))$ . Note than another approach was proposed in [38]. The main drawback of their method is to compute the roots of an equation using bisection, which is quite slow.

---

**Algorithm 2** Projection on the  $\ell_{2,1}$  norm— $\text{proj}_{\ell_1}(V, \eta)$  is the projection on the  $\ell_1$ -ball of radius  $\eta$

---

**Input:**  $V, \eta$

$$t := \text{proj}_{\ell_1}((\|v_i\|_i)_{i=1}^d, \eta)$$

**for**  $i = 1, \dots, d$  **do**

$$w_i := \text{proj}_{\ell_2}(v_i, t_i)$$

**end for**

**Output:**  $W$

---

*A new optimizer with an adaptive weighted  $\ell_{1,1}$  norm constraint:* The "sparsification" of a matrix  $A \in \mathbf{R}^{m \times n}$  consists in finding a matrix  $B \in \mathbf{R}^{m \times n}$

- (i) with as many 0 components as possible (memory footprint decrease),
- (ii) with as many vanishing columns as possible (both MACCs and memory footprint decrease),
- (iii) without degrading too much the performance of the network.

The matrix  $A$  defines a mapping  $A : \mathbf{R}^n \mapsto \mathbf{R}^m$ . In order to avoid degrading too much the performance of the network, it can be required that, for each layer, the matrix  $A$  is replaced with a sparsified version  $B$  such that, for any  $x \in \mathbf{R}_*^n$ , the value of  $\|(A - B)x\|$  is as small as possible for a given norm. The goal is to find a matrix  $B : \mathbf{R}^n \mapsto \mathbf{R}^m$  with much more vanishing components than in the matrix  $A$ , thus it is convenient to impose the constraint  $\sum_{i=1}^n \|B^i\|_1 < \sum_{i=1}^n \|A^i\|_1$ . where  $B_i$  respectively  $A_i$  is the  $i$  th column of the matrix. Let recall the induced operator norm of  $A - B$  in  $\ell_1$  domain with  $\ell_1$  co-domain

$$\|A - B\|_{1,1} = \left( \sup_{\|x\|_1=1} \|(A - B) \cdot x\|_1 \right) \quad (5)$$

which is computed as the maximum  $\ell_1$  norm of a columns of  $A - B$ . We consider therefore the following problem

$$B^* = \underset{\sum_{i=1}^n \|B^i\|_1 = \beta}{\text{argmin}} \left( \sup_{\|x\|_1=1} \|(A - B) \cdot x\|_1 \right) \quad (6)$$

It is possible to show ([4]) the following result:

**Theorem 1** Let  $A = (A^1 \dots A^n)$  be a  $\mathbf{R}^{m \times n}$  matrix. Assume w.l.o.g. that  $A^1 \geq A^2 \geq \dots \geq A^{n-1} \geq A^n$ . For an optimal solution  $B_*$  for (6), there exists  $p \in \{0, \dots, n\}$  such that

- (i) For  $i = 0, \dots, p, A^i - B_*^i = A^p - B_*^p$
- (ii) For  $i > p, B_*^i = 0$

Applying Theorem 1 to (6) we obtained the following algorithm: we compute first the radius  $t_i$  and then project the rows using  $\ell_1$  adaptive constraint  $t_i$ :

### Lottery optimizer

Following the work by Frankle and Carbin [16] further developed by [66] which proposed a simple algorithm for finding sparse sub-networks within larger networks that are trainable from scratch. Their approach to finding these sparse

---

**Algorithm 3** Projection on the  $\ell_{1,1}$  norm— $\text{proj}_{\ell_1}(V, \eta)$  is the projection on the  $\ell_1$ -ball of radius  $\eta$

---

**Input:**  $V, \eta$   
 $t := \text{proj}_{\ell_1}(\|v_i\|_1)_{i=1}^d, \eta$   
**for**  $i = 1, \dots, d$  **do**  
 $w_i := \text{proj}_{\ell_1}(v_i, t_i)$   
**end for**  
**Output:**  $W$

---

networks is as follows: after training a network, set all weights smaller than some threshold to zero, rewind the rest of the weights to their initial configuration, and then retrain the network from this starting configuration but with the zero weights frozen (not trained).

We replace the thresholding by our  $\ell_{1,1}$  projection and devise the following algorithm:

---

**Algorithm 4** Projection on the  $\ell_{1,1}$  norm— $\text{proj}_{\ell_1}(V, \eta)$  is the projection on the  $\ell_1$ -ball of radius  $\eta$ ,  $\nabla L(W, M_0)$  is the masked gradient with binary mask  $M_0$ , and  $f$  is the ADAM optimizer,  $\gamma$  is the learning rate

---

**Input:**  $W^*, \gamma, \eta$   
**for**  $n = 1, \dots, N(\text{epochs})$  **do**  
 $V \leftarrow f(W, \gamma, \nabla L(W))$   
**end for**  
 $t := \text{proj}_{\ell_1}(\|v_i\|_1)_{i=1}^d, \eta$   
**for**  $i = 1, \dots, d$  **do**  
 $w_i := \text{proj}_{\ell_1}(v_i, t_i)$   
**end for**  
**Output:**  $W, M_0$   
**Input:**  $W^*$   
**for**  $n = 1, \dots, N(\text{epoch})$  **do**  
 $W \leftarrow f(W, \gamma, \nabla L(W, M_0))$   
**end for**  
**Output:**  $W$

---

#### IV. EXPERIMENTAL RESULTS

We modify the pytorch framework to implement our sparse learning method using a constraint approach. We choose Adam optimizer [31], a standard optimizer in PyTorch as baseline comparison to our optimizer with  $\ell_1$  and nuclear norm constraints.

We denote as PGL1, the algorithm with  $\ell_1$  constraint, PGL21, the algorithm with  $\ell_{2,1}$  constraint and PGL11 the algorithm with  $\ell_{1,1}$  constraint.

The first layer, which interacts directly with the input image and the last one interacting directly with the output are most sensitive to sparsity and thus we do not apply sparsity constraint.

We use the entropy (bit/weight) of the weights distributions to compute estimations of the models storage memory cost. To this end, the weights can for instance be coded using JPEG2000<sup>1</sup> an image coding system that uses state-of-the-

art compression techniques based on wavelet technology. The classical computational cost evaluates FLOPS (floating point operations per second) as a measure. When using FLOPS, additions (accumulates) and multiplications are counted separately. However, a lot of hardware can compute multiply-add operation in a single instruction. We use therefore **MACCs** (multiply-accumulate operations) as computational cost for which one multiplication and one addition are counted as a single instruction. All experiments are performed on a Cocolink Klimax 210 HPC with 10 GPUs (Nvidia Quadro P6000, P100 and GeForce GTX 1080).

#### Results on MNIST

We select the popular MNIST dataset [34] containing  $28 \times 28$  grey-scale images of handwriting digits of 10 classes (from 0 to 9). This dataset consists on a training set of 60,000 instances and in a test set of 10,000 instances.

We consider a neural network with two convolutional layers and two linear layers denoted as Net4. The size of its weight matrices are  $(1 \times 10 \times 5 \times 5)$ ,  $(10 \times 20 \times 5 \times 5)$ ,  $(320 \times 50)$  and  $(50 \times 10)$  respectively. Thus the total number of elements of these weight matrices are 250, 5000, 16000 and 500 respectively.

The sizes of matrix weights are very unbalanced. Thus our strategy is to sparsify the 2 most numerous layers, i.e. the 2<sup>nd</sup> convolutional and the 1<sup>st</sup> linear layer with the same optimizer.

We study the layerwise influence of the constraint parameter  $\eta$  on the accuracy and the weight sparsity defined as the percentage of weights set to zero. For comparison purpose, the weights using Adam for the two linear layers are depicted in figure 3 top shows unstructured sparsity while figure 3 bottom using PGN shows structured sparsity.

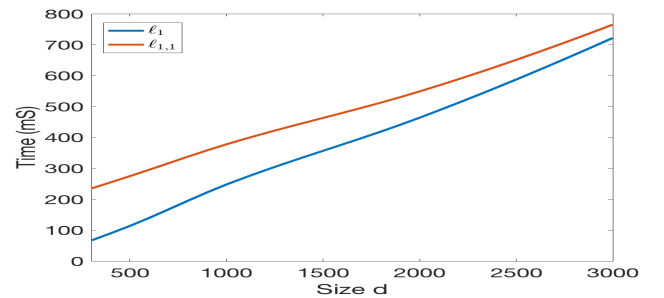


Fig. 1: Comparison of computational time for  $\ell_1$  and  $\ell_{1,1}$  constraints versus size  $d$  (Input  $m=1000$ )

The weight distributions using a Parzen kernel method (Fig. 2) top shows that weights distributions follow a Gaussian shape for Adagrad and Adam optimizer. Figure 2) bottom shows that weights distributions follow a Laplacian shape (thus lower entropy) for PGL1 and PGL11.

<sup>1</sup><https://jpeg.org/jpeg2000/index.html>

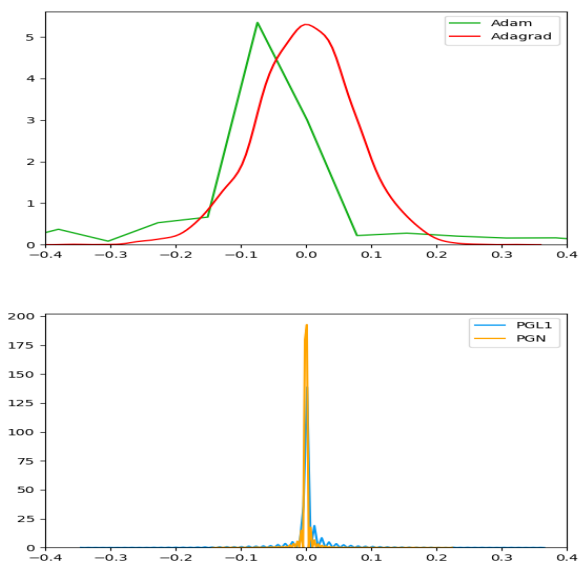


Fig. 2: MNIST, Net4,Up: Distribution of convolutional layer Conv2 with Adam and Adagrad optimizers, Bottom : Distribution of convolutional layer Conv2 with PGL1 and PGL11 optimizers

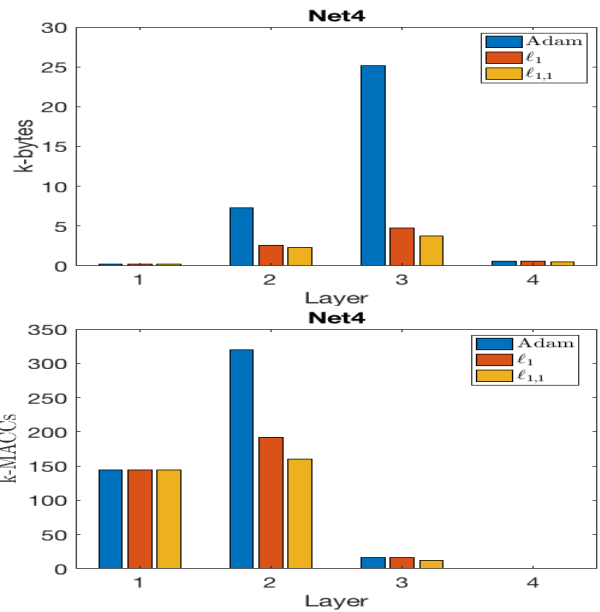


Fig. 4: MNIST with Net4: Top Memory per layer for optimizers Adam, PGL1 and PGL11. Bottom: MACCs for optimizers Adam, PGL1 and PGL11

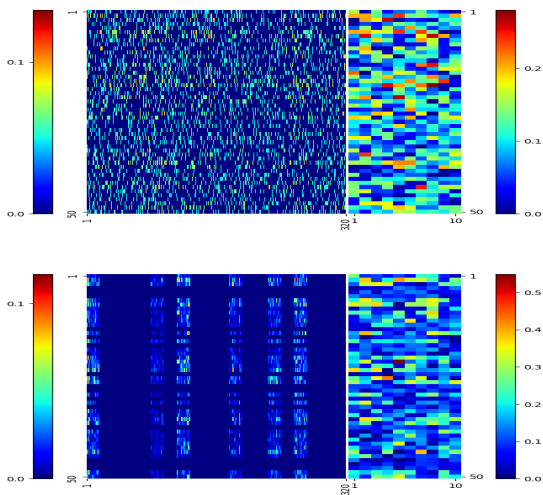


Fig. 3: Visualization for weight matrices: Top of the two linear layers Linear1 and Linear2 ) of Net4 with Adam and weights thresholding shows unstructured sparsity. Bottom with structured optimizer: Layer Linear1 exhibits a high structured sparsity.

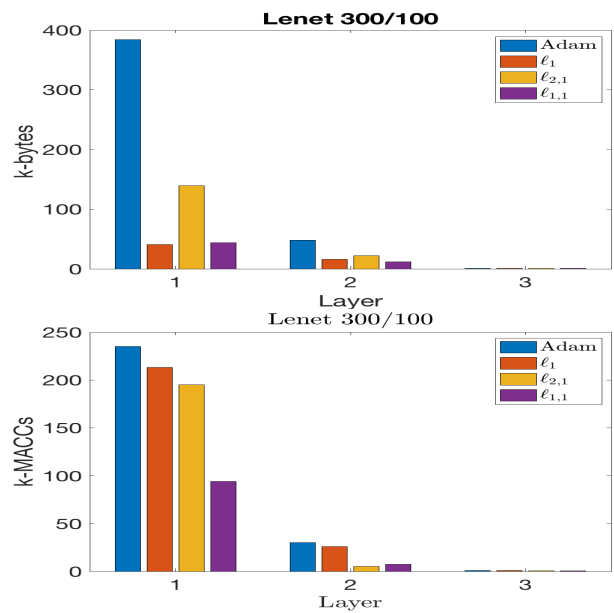


Fig. 5: MNIST with Lenet: Top Memory per layer for optimizers Adam, PGL1, PGL21 and PGL11. Bottom: MACCs for optimizers Adam, PGL1,PGL21 and PGL11

TABLE I: MNIST Net 4 total memory, MACCs, and accuracy

Methods	Memory (kBytes)	MACCs (k-MACCs)	Accuracy (%)
Adam	33.64	563.7	99.
PGL1 eta=100	6.82	460	97.5
PGL11 eta= 80	9.03	380	97.7

TABLE II: MNIST Le Net 300/100 total memory, MACCs, and accuracy

Methods	Memory (kBytes)	MACCs (k-MACCs)	Accuracy (%)
ADAM	440	266.2	98.21
PGL1(250)	34.03	217.7	96.17
PGL11( $\eta = 250$ )	34.42	89.9	97.2
Tartaglione [54]	33.72	-	96.6
PGL21(30)	197	217	97.6

Our splitting algorithm with  $\ell_{1,1}(L11)$  structured constraint outperforms state of the art sparsifying methods for memory footprint. The main advantage of our method using the  $\ell_{1,1}$  constraint over  $\ell_1$  and [54] is the reduction of the calculation cost (MACCs) by a factor 3 which is crucial for low capacity devices. Note that convolutional network Net 4 requires at least 380 k-MACCs while fully connected Lenet 300/100 network requires only 90 k-MACCs. Note the poor performances of  $\ell_{2,1}$  constraint.

#### Results on Fashion MNIST

**Results on Fashion MNIST with convolutional Net5**<sup>2</sup> Fashion-MNIST [58] is a dataset of Zalando’s article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. Fashion-MNIST and MNIST share the same image size and structure of training and testing splits.

TABLE III: Fashion MNIST Net 5 total memory, MACCs, and accuracy

Methods	Memory (kBytes)	MACCs (k-MACCs)	Accuracy (%)
ADAM	45	24,700	89.7
PGL1(200)	11.06	21,270	84.18
PGL11(250)	16.82	20,700	84.46

Note that computational cost is huge (24,700 k-MACCs) with Net 5 and none of the method is efficient for reducing computational cost.

#### Results on Fashion MNIST with Lenet 300/100

Note that computational cost is  $\times 200$  smaller with Lenet 300/100 than with Net5 (266 k-MACCs). Moreover PGL11 projection is efficient reducing computational cost by a factor 2.

<sup>2</sup><https://engmrk.com/lenet-5-a-classic-cnn-architecture/>

TABLE IV: Fashion MNIST Lenet 300/100 total memory, MACCs, and accuracy

Methods	Memory (kBytes)	MACCs (k-MACCs)	Accuracy (%)
ADAM	418	266.2	89.3
PGL1(400)	40.06	193	84.37
PGL11(700)	52.5	136.08	84.8

#### Results on CIFAR10

The CIFAR-10 data set is composed of 60,000 32x32 colour images, 6,000 images per class, for a classification in 10 classes. The training set is made up of 50,000 images, while the remaining 10,000 are used for the testing set.

We use *Simplenet*<sup>3</sup>, the highly optimized architecture [23] This network is composed of 13 blocks  $B_i = (\text{convolutional2D}/\text{Batch Normalization}/\text{ReLU})$  for  $i = 1, \dots, 13$  with sequences *MaxPool2d/ Dropout* after the blocks  $B_4, B_7, B_9, B_{10}, B_{12}$  and  $B_{13}$  followed by a classifier layer. For this dataset, we use Lottery optimizer (Algorithm 4).

Results are reported in Figures 9 and 8 and in the Table VI.

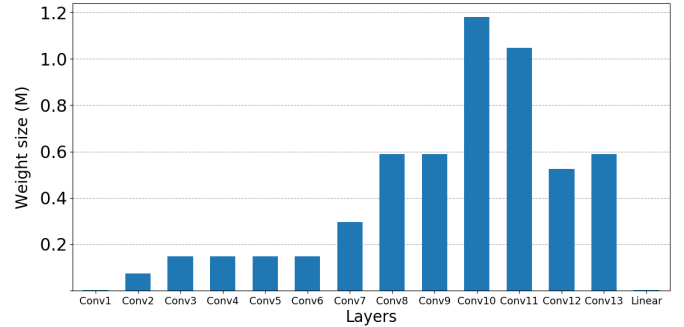


Fig. 6: SimpleNet, weight size for each layer

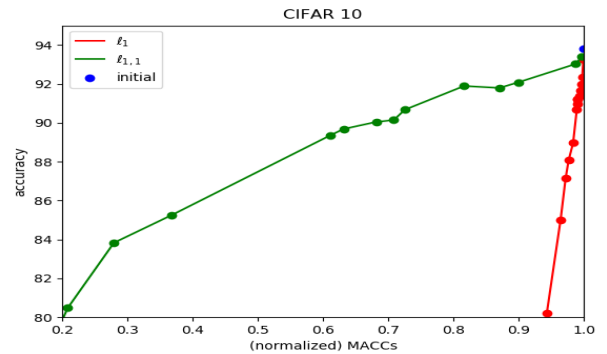


Fig. 7: CIFAR 10 SimpleNet, Accuracy as a function of MACCs, PGL1 (red) and PGL11 (green)

Figure 8 and Table (VI) shows a large global memory reduction by a factor 25. On the other hand, the reduction

<sup>3</sup>[https://github.com/Coderx7/SimpleNet\\_Pytorch](https://github.com/Coderx7/SimpleNet_Pytorch)



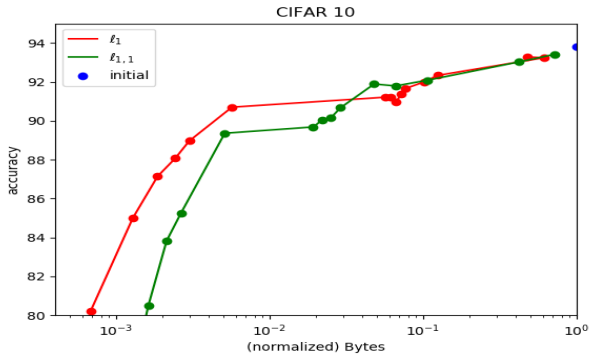


Fig. 8: CIFAR 10 SimpleNet, Accuracy as a function of Bytes, PGL1 (red) and PGL11 (green)

TABLE V: CIFAR10: MACCs, memory, and accuracy using Simplenet

Methods	MACCs (M-MACCs)	Memory (M-Bytes)	Accuracy %
Adam	631.51	9.44	93.8
PGL1	626.08	0.37	91.12
PGL11	441	0.37	91

of the calculation cost is low for  $\ell_{1,1}$  constraint and null for  $\ell_1$  constraint.

### Results on biological dataset

Lung dataset [41] is a metabolomic dataset with 1005 samples and 2944 features. These are urine samples obtained from two groups of patients, one group has a lung cancer, the other is a control group. We use a linear fully connected network (LFC) with an input layer, 3 hidden layer layers followed by a RELU activation function and followed by a classifier layer inspired by the network in [32].

TABLE VI: Lung dataset: MACCs , memory, and accuracy using LFC network

Methods	MACCs (M-MACCs)	Memory (M-Bytes)	Accuracy %
Adam	4.03	53.4	77
PGL1	4.03	5.3	82.2
PGL11	2.4	5.3	81.2

Contrary to the previous results, for a compression ratio of 10 on the memory bytes, the accuracy increases by 4.2% and 5.2% with PGL1 and PGL11 respectively. Note that accuracy is slightly better for memory bytes compression when using  $\ell_1$  constraint. We obtain compression ratio of 1.7 for MACCS with accuracy improvement by 4.2% when using  $\ell_{1,1}$  constraint.

## V. DISCUSSION

To the best of our knowledge, entropy of the weights (bit/weight) as a measure of memory was never reported in the DNN literature. Thus memory comparison with previous results is not straightforward. A more in depth study will be performed as well as the JPEG2000 compression of the model

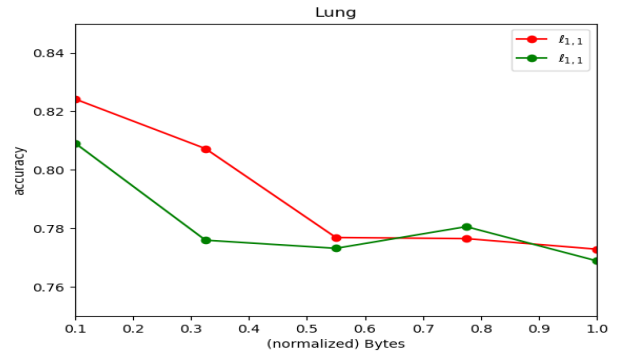
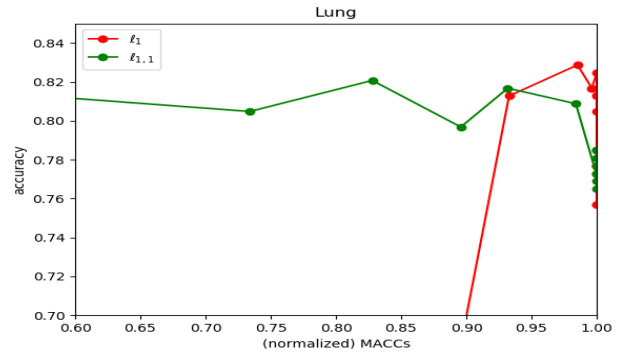


Fig. 9: Lung LFC Network ,Left Accuracy as a function of MACCs, PGL1 (red) and PGL11 (green),Right: Accuracy as a function of Bytes, PGL1 (red) and PGL11 (green)

for storage in a forthcoming paper.

Energy consumption is directly related to the number of instructions (Flops or MACCs) [20].

Experimental results show that convolutional networks require a huge amount of Flops and thus Energy while improving very slightly accuracy. On the MNIST and Fashion MNIST datasets, we note an improvement by a factor 200 for MACCs by using linear fully connected (LFC) Lenet 300/100 instead of convolutional neural networks (CNN).

Experimental results show, as expected, that our new  $\ell_{1,1}$  norm constraint is more efficient for structured sparsity and thus MACCs reduction than  $\ell_1$  constraint.

## VI. CONCLUSION

In order to cope with computational issue of DNN, a huge literature deals with proximal regularization methods which are time consuming. In this paper, we propose instead a constrained approach. We provide a general framework of our new splitting projection gradient method. Our splitting algorithm iterates a gradient step and a projection on the convex constraint. We design a new algorithm for the  $\ell_{1,1}$  constraint. Our experiments show the benefit of Lottery optimizer which uses only one projection for deep neural networks sparsification.

Experiments on three popular dataset (MNIST, FASHION MNIST and CIFAR) and a metabolomics dataset show that our projection method with our new  $\ell_{1,1}$  constraint provides better



structured sparsity resulting in high reduction of memory and computational power reduction.

We highlight the two main results. First fully connected neural networks outperform in terms of MACCs convolutional networks by a factor of 200. Second our algorithm with L11 projection improve MACCS by a factor of 2. So we believe that fully connected neural networks with our new  $\ell_{1,1}$  projections is the best solution to reduce energy consumption and thus for green AI.

We are currently developing layer-wise adaptive constraints ( $\eta$ ) and applying our method to other large Neural Networks.

## REFERENCES

- [1] A Ali and R Tibshirani. The generalized lasso problem and uniqueness. *Electronic Journal of Statistics*, 13(2):2307–2347, 2019.
- [2] Jose M Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, pages 2270–2278, 2016.
- [3] M. Barlaud, W. Belhajali, P. L. Combettes, and L. Fillatre. Classification and regression using an outer approximation projection-gradient method. volume 65, pages 4635–4643, 2017.
- [4] M. Barlaud and F. Guyard. The  $\ell_{1,1}$  projection and matrices sparsification. in progress, 2020.
- [5] Michel Barlaud, Antonin Chambolle, and Jean-Baptiste Caillaud. Robust supervised classification and feature selection using a primal-dual method. *arXiv cs.LG/1902.01600*, 2019.
- [6] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv 1605.07678*, 2016.
- [7] Miguel Á. Carreira-Perpiñán and Yerlan Idelbayev. Learning-compression algorithms for neural net pruning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [8] Jacopo Cavazza, Pietro Morerio, Benjamin Haefele, Connor Lane, Vittorio Murino, and Rene Vidal. Dropout as a low-rank regularizer for matrix factorization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 435–444, 2018.
- [9] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv:1710.09282*, 2017.
- [10] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.
- [11] L. Condat. Fast projection onto the simplex and the  $\ell_1$  ball. *Mathematical Programming Series A*, 158(1):575–585, 2016.
- [12] Misha Denil, Babak Shikibi, Laurent Dinh, Nando De Freitas, et al. Predicting parameters in deep learning. In *Advances in neural information processing systems*, pages 2148–2156, 2013.
- [13] D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via  $\ell_1$  minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.
- [14] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the  $\ell_1$ -ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.
- [15] Mario Figueiredo and Robert Nowak. Ordered weighted  $\ell_1$  regularized regression with strongly correlated covariates: Theoretical aspects. In *Artificial Intelligence and Statistics*, pages 930–938, 2016.
- [16] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.
- [17] J. Friedman, T. Hastie, and R. Tibshirani. Regularization path for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33:1–122, 2010.
- [18] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.
- [19] Aidan N Gomez, Ivan Zhang, Kevin Swersky, Yarín Gal, and Geoffrey E Hinton. Learning sparse networks using targeted dropout. *arXiv :1905.13678*, 2019.
- [20] E Grochowski and M Annavaram. Energy per instruction trends in intel  $\text{\textcircled{R}}$  microprocessors. 2006.
- [21] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, et al. ESE: Efficient speech recognition engine with sparse lstm on fpga. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 75–84. ACM, 2017.
- [22] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.
- [23] Seyyed Hossein Hasanpour, Mohammad Rouhani, Mohsen Fayyaz, and Mohammad Sabokrou. Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv preprint arXiv:1608.06037*, 2016.
- [24] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.
- [25] T. Hastie, R. Tibshirani, and M. Wainwright. Statistical learning with sparsity: The lasso and generalizations. *CRC Press*, 2015.
- [26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [27] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.
- [28] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.
- [29] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 304–320, 2018.
- [30] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool. AI benchmark: Running deep neural networks on android smartphones. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.
- [31] D Kingma and J Ba. a method for stochastic optimization. *International Conference on Learning Representations*, pages=1–13, year=2015,.
- [32] Anna Klimovskaia, David Lopez-Paz, Léon Bottou, and Maximilian Nickel. Poincaré maps for analyzing complex hierarchies in single-cell data. *bioRxiv*, 2019.
- [33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [34] Yann LeCun. The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>.
- [35] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.
- [36] Shaohui Lin, Rongrong Ji, Yuchao Li, Cheng Deng, and Xuelong Li. Toward compact convnets via structure-sparsity regularized filter pruning. *IEEE transactions on neural networks and learning systems*, 2019.
- [37] P.-L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.
- [38] Jun Liu, Shuiwang Ji, and Jieping Ye. Multi-task feature learning via efficient  $\ell_2$ ,  $\ell_1$ -norm minimization. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 339–348. AUAI Press, 2009.
- [39] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017.
- [40] J. Mairal and B. Yu. Complexity analysis of the lasso regularization path. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 353–360, 2012.
- [41] E. Mathé et al. Noninvasive urinary metabolomic profiling identifies diagnostic and prognostic markers in lung cancer. *Cancer research*, 74(12):3259–3270, June 2014.
- [42] J.J Moreau. Proximité et dualité dans un espace hilbertien. *Bull. Soc.Math. France.*, 93, pages 273–299, 1965.
- [43] S. Mosci, L. Rosasco, M. Santoro, A. Verri, and S. Villa. Solving structured sparsity regularization with proximal methods. In *Machine Learning and Knowledge Discovery in Databases*, pages 418–433. Springer, 2010.

- [44] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE Access*, 7:19143–19165, 2019.
- [45] Urvashi Oswal, Christopher Cox, Matthew Lambon-Ralph, Timothy Rogers, and Robert Nowak. Representational similarity learning with application to brain networks. In *International Conference on Machine Learning*, pages 1041–1049, 2016.
- [46] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning in natural language processing. *arXiv:1807.10854*, 2018.
- [47] Guillaume Perez, Michel Barlaud, Lionel Fillatre, and Jean-Charles Régin. A filtered bucket-clustering method for projection onto the simplex and the  $\ell_1$ -ball. *Mathematical Programming*, May 2019.
- [48] S Rallapalli, H Qiu, A Bency, S Karthikeyan, R Govindan, B Manjunath, and R Uргаonkar. Are very deep neural networks feasible on mobile devices. *IEEE Trans. Circ. Syst. Video Technol.*, 2016.
- [49] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.
- [50] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai, 2019.
- [51] S. Sra, S. Nowozin, and S. J. Wright. *Optimization for Machine Learning*. MIT Press, 2012.
- [52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- [53] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *ACL*, 2019.
- [54] Enzo Tartaglione, Skjalg Lepsoy, Attilio Fiandrotti, and Gianluca Francini. Learning sparse neural networks via sensitivity-driven regularization. In *Advances in Neural Information Processing Systems*, pages 3878–3888, 2018.
- [55] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
- [56] Amirsina Torfi, Rouzbeh A Shirvani, Sobhan Soleymani, and Nasser M Nasrabadi. Attention-based guided structured sparsity of deep neural networks. *arXiv preprint arXiv:1802.09902*, 2018.
- [57] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016.
- [58] Han Xiao, K Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv cs.LG/1708.07747*, 2017.
- [59] Jaehong Yoon and Sung Ju Hwang. Combined group and exclusive sparsity for deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3958–3966. JMLR. org, 2017.
- [60] Ming Yuan and Yi 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.
- [61] Ming Yuan and Yi 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.
- [62] D. Zhang, Y. Hu, J. Ye, X Li, and X He. Matrix completion by truncated nuclear norm regularization. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012.
- [63] Dejiao Zhang, Julian Katz-Samuels, Mário AT Figueiredo, and Laura Balzano. Simultaneous sparsity and parameter tying for deep learning using ordered weighted  $\ell_1$  regularization. In *2018 IEEE Statistical Signal Processing Workshop (SSP)*, pages 65–69. IEEE, 2018.
- [64] Dejiao Zhang, Haozhu Wang, Mario Figueiredo, and Laura Balzano. Learning to share: Simultaneous parameter tying and sparsification in deep learning. 2018.
- [65] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pages 662–677. Springer, 2016.
- [66] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3597–3607. Curran Associates, Inc., 2019.
- [67] Xiaotian Zhu, Wengang Zhou, and Houqiang Li. Improving deep neural network sparsity through decorrelation regularization. In *IJCAI*, pages 3264–3270, 2018.