Accepted for publication ICPR 2020 Milan

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

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

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

***Abstract*—Deep neural networks (DNN) have been applied recently to different domains and perform better than classical state-of-the-art methods. However the high level of performances of DNNs is most often obtained with networks containing millions of parameters and for which training requires substantial computational power. To deal with this computational issue proximal regularization methods have been proposed in the literature but they are time consuming.**

In this paper, we propose instead a constrained approach. We provide the general framework for this new projection gradient method. Our algorithm iterates a gradient step and a projection on convex constraints. We studied algorithms for different constraints: the classical ℓ_1 unstructured constraint and structured constraints such as 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 used the recent "Lottery optimizer" replacing the threshold by our $\ell_{1,1}$ projection. We demonstrate the effectiveness of this method with three popular datasets (MNIST, Fashion MNIST and CIFAR). Experiments with these datasets show that our projection method using this new $\ell_{1,1}$ structured constraint provides the best decrease in memory and computational power.

I. Motivation

Deep neural networks have been applied recently to different domains and have shown a dramatic improvement in accuracy of image recognition [1], speech recognition [2] or natural language processing [3]. These studies relied on deep networks with millions or even billions of parameters. For instance, the original training of ResNet-50 [4] (image classification) contains 25.6M parameters and required 29 hours of processing using 8 GPUs. Storing the model requires 98MB. The cost in memory of the inference on a single 224x224 image is about 103MB and 4 GFLOPs are needed [5]. The recent development of DNNs, hardware accelerators like GPUs and the availability of deep learning frameworks for smartphones [6] suggest seamless transfer of DNN models trained on servers onto mobile devices. However, it turns out that the memory [7] and energy consumption [8] are still the main bottlenecks for running DNNs on such devices.

Thus the computational cost has an impact on the carbon footprint. It has been argued that this trend is environmentally unfriendly [9]. Some authors of [10] advocate a practical solution by providing 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 projection onto this convex set. The benefits of this formulation are twofold. Firstly, the constraint has a direct geometric interpretation whereas the impact of parameter values in traditional regularization methods are more difficult to understand. Secondly, the convergence of this new method is formally proved.

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

II. Related Works

Weights sparsification

It is well known [11] that DNN models are largely over-parametrized and that in practice, relatively few network weights are actually necessary to accurately learn 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 [12], [13]. The ℓ_1 penalty added to the classification cost can be interpreted as a convexification of the ℓ_0 penalty. In [14], weights with the smallest amplitude in pre-trained networks are removed. Model sensitivity to weights can also be used [15], [16]. where weights with a weak influence on the network output are pruned. Constraint optimization is used in order to learn sparse networks with ℓ_0, ℓ_1 or ℓ_2 constraints on the weights [17].

These methods generally produce networks with random sparse connectivity, i.e. high-dimensional but sparse weight matrices. They only partially reduce 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 computational 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. Filters in CNN are pruned based on the ℓ_1 norm of their kernel weights [18]. Some authors perform channel pruning using LASSO regression and least squared reconstruction [19]. Neurons are pruned based on the average percentage of zeros (APoZ) after the ReLU activation [20].

Learning structured sparse DNNs using regularization methods

In contrast to the case of weight sparsification, neuron level sparsification introduces a new challenge forcing adoption of other types of regularization.
The most common approaches are based on *group LASSO* $\ell_{2,1}$ [21] or on *sparse group LASSO* $\ell_{2,1} + \ell_1$ [12] 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 is used [22]. Group LASSO regularization in a number of studies in [22], [23]. Group LASSO and filter decorrelation regularization are used in order to discard CNN filters [24]. Group LASSO and group variance regularization have been used in [25].

To deal with non-smooth ℓ_1 regularization, subgradient descent is used in [26]. Here, structured sparsification is performed without group regularization.
The idea is to scale the neurons output with a given factor λ_i and apply ℓ_1 regularization to push the various factors λ_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) [27]. Nuclear norm penalty was successfully used in matrix low rank approximation [27], matrix completion [28], matrix factorization [29] and DNN dropout modeling [30].

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. We recall the proximal operator of a function $f(x)$ [31]:

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

Let W be the weight matrix of a neural network, $L(W)$ be a gradient Lipschitz loss and $R(w)$ be a convex penalty. We 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 [32], [33], [34], [35]. Using SGD with penalization is limited and time consuming due to the tuning of the corresponding penalization hyper-parameters [36], [37].

Proximal gradient descent with group LASSO constraints was used in [38] and in [39]. In [40], the output neurons are scaled using a factor λ_i and an accelerated proximal gradient is used with a ℓ_1 constraint to push as many coefficients λ_i towards 0 without significantly decreasing the performance. In [41], neuron sparsification is performed 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 [42] proximal gradient descent with group OWL constraint (grOWL [43]) is used to simultaneously sparsify neurons and enforce parameter sharing. Proximal gradient descent is also used in [44] where Ordered Weighted ℓ_1 regularization (OWL [45]) allowing simultaneous sparsify weights and optimized weight sharing. In [46], 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 an alternative constrained approach that takes advantage of available efficient projections on the ℓ_1-ball [47], [48], [49] and on the $\ell_{2,1}$ ball [50], [51]. We further propose a new $\ell_{1,1}$ projection.

III. Learning sparse DNN

A Projection gradient algorithm for constrained learning

In this work, we propose a constrained approach in which the constraint is directly related to the number of zero-weights. Moreover it takes advantage of an available efficient projection on the ℓ_1-ball [47], [49].
Let $L(W)$ be a gradient Lipschitz loss, $R(w)$ be a convex constraint, and C its convex set. Lets us define the following criterion

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

where the scalar $\eta \geqslant 0$ 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 [52]:

$$V_n := W_n - \gamma \nabla L(W_n), \quad (3)$$
$$W_{n+1} := \mathrm{proj}(V_n) + \varepsilon_n, \quad (4)$$

where proj denotes the projection on the convex constraint. We can therefore apply the algorithm to any constraint for which an 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 $\mathrm{proj}(\eta V)$ is the projection on the constraint

Input: $X, Y, W_0, N, \gamma, \eta$
for $n = 1, \ldots, N$ **do**
  $V \leftarrow W - \gamma \cdot \nabla L(W)$
  $W \leftarrow \mathrm{proj}(\eta V)$
end for
Output: W

Optimizer with structured constraints

In the case of the constraint $R(w) = \|W\|_1$, efficient algorithms have been proposed [47], [49]. Unfortunately this ℓ_1 constraint does not induce a sparse structure.

$$\|W\|_1 := \sum_{i,j} |w_{i,j}|.$$

Classical optimizer with $\ell_{2,1}$ norm constraint (Group LASSO): The Group LASSO was first introduced in [53]. The main idea of Group LASSO is to enforce model parameters for different classes to share features. Group sparsity reduces 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 $n \times d$ matrix W (whose columns are denoted w_i, $i = 1, \ldots, d$) is defined as follows:

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

We use the approach proposed in [51] to compute the projection W on the $\ell_{2,1}$-ball of radius η of a $n \times d$ matrix V (whose columns are denoted v_i, $i = 1, \cdots, d$) : *compute t_i which is the projection of the vector $(\|v_i\|_i)_{i=1}^d$ on the ℓ_1 ball of radius η in $\mathbf{R}^n$. Each column of the projection is then obtained according to*

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

This last operation is denoted as $W_i := \mathrm{proj}_{\ell_2}(V_i, t_i)$ in Algorithm 2.
This algorithm requires the projection of the vector $(\|v_i\|_i)_{i=1}^d$ on the ℓ_1 ball of $\mathbf{R}^n$ of radius η whose complexity is only $O(d \times \log(d))$. We can note that another approach was proposed in [50]. 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—$\mathrm{proj}_{\ell_1}(V, \eta)$ is the projection on the ℓ_1-ball of radius η

Input: V, η
$t := \mathrm{proj}_{\ell_1}((\|v_i\|_i)_{i=1}^d, \eta)$
for $i = 1, \ldots, d$ **do**
  $w_i := \mathrm{proj}_{\ell_2}(v_i, t_i)$
end for
Output: W

A new optimizer with an adaptive weighted $\ell_{1,1}$ norm constraint: Unfortunately, algorithm 2 does not provide efficient sparsity. Thus we propose the algorithm 3.

Algorithm 3 Projection on the $\ell_{1,1}$ norm—$\mathrm{proj}_{\ell_1}(V, \eta)$ is the projection on the ℓ_1-ball of radius η

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

This "projection" can be seen in the following way: Given a matrix $V = (v_{i,j})$, $1 \le i \le d$, $1 \le j \le n$, the algorithm computes $\bar{t}$ as the solution of:

$$\min_{\sum_{i=1}^d |t_i| \le \eta} \sum_{i=1}^{d} \left(\sum_{j=1}^{n} |v_{i,j}| - t_i \right)^2 \quad (5)$$

and then for all $i = 1, \ldots, d$, the solution $\bar{w}$ of:

$$\min_{\sum_{j=1}^n |w_{i,j}| \le \bar{t}_i} \sum_{j=1}^{n} (v_{i,j} - w_{i,j})^2. \quad (6)$$

Clearly both $\bar{t}$ and $\bar{w}$ are bounded by η in their respective ℓ_1 norm.

Let us consider, for $\varepsilon > 0$, the problem (P_ε)

$$\min_{\substack{\sum_i t_i \le \eta \\ \sum_j |w_{i,j}| - t_i \le 0}} \sum_{i=1}^{d} \left(\sum_{j=1}^{n} |v_{i,j}| - t_i \right)^2 + \varepsilon \sum_{i=1}^{d} \sum_{j=1}^{n} (v_{i,j} - w_{i,j})^2$$

It turns out that this is a (strongly) convex problem, with solution $(t^\varepsilon, w^\varepsilon)$. It is easy to show that $(t^\varepsilon, w^\varepsilon) \to (\bar{t}, \bar{w})$ as $\varepsilon \to 0$. Indeed , since $(\bar{t}, \bar{w})$ is admissible $\sum_i \bar{t}_i \le \eta$ and

$\sum_j |\bar{w}_{i,j}| \leq \bar{t}_i$, and then, using the strong convexity of the objective with respect to t_i,

$$\sum_{i=1}^{d} (t_i^\varepsilon - \bar{t}_i)^2 + \sum_{i=1}^{d} \left(\sum_{j=1}^{n} |v_{i,j}| - t_i^\varepsilon \right)^2 + \varepsilon \sum_{i=1}^{d} \sum_{j=1}^{n} (v_{i,j} - w_{i,j}^\varepsilon)^2 \leq \sum_{i=1}^{d} \left(\sum_{j=1}^{n} |v_{i,j}| - \bar{t}_i \right)^2 + \varepsilon \sum_{i=1}^{d} \sum_{j=1}^{n} (v_{i,j} - \bar{w}_{i,j})^2.$$

Using first that $\bar{t}$ is a minimizer of the first term in the right-hand side, we find:

$$\sum_{i=1}^{d} (t_i^\varepsilon - \bar{t}_i)^2 + \varepsilon \sum_{i=1}^{d} \sum_{j=1}^{n} (v_{i,j} - w_{i,j}^\varepsilon)^2 \leq \varepsilon \sum_{i=1}^{d} \sum_{j=1}^{n} (v_{i,j} - \bar{w}_{i,j})^2$$

Sending $\varepsilon \to 0$, we obtain that $t^\varepsilon \to \bar{t}$. Then, dividing the expression by ε, and considering $\varepsilon \to 0$ again along a subsequence for which w^ε converges to some limit w, we obtain:

$$\sum_{i=1}^{d} \sum_{j=1}^{n} (v_{i,j} - w_{i,j})^2 \leq \sum_{i=1}^{d} \sum_{j=1}^{n} (v_{i,j} - \bar{w}_{i,j})^2.$$

Now, since in the limit, $t^\varepsilon \to \bar{t}$, we have $\sum_{j=1}^{n} |w_{i,j}| \leq \bar{t}_i$ for all $i = 1, \ldots, d$, so that w is admissible in the minimization problem (6) whose solution is $\bar{w}$. Hence $w = \bar{w}$, and in fact we get that $\lim_{\varepsilon \to 0} w^\varepsilon = \bar{w}$. Observe also that this shows also $\|t^\varepsilon - \bar{t}\| = o(\sqrt{\varepsilon})$. Similarly, one can show that $\|w^\varepsilon - \bar{w}\| = o(\varepsilon^{1/4})$. Actually a more precise analysis shows that $\|t^\varepsilon - \bar{t}\| = O(\varepsilon^{2/3})$ and $\|w^\varepsilon - \bar{w}\| = O(\varepsilon^{1/3})$.

A further remark is that one can show the problem (P_ε) to be equivalent to (P_ε^+) defined with

$$\min_{\substack{\sum_i t_i \leq \eta \\ \sum_j |w_{i,j}| - t_i \leq 0}} \sum_{i=1}^{d} \left[\left(\sum_{j=1}^{n} |v_{i,j}| - t_i \right)^+ \right]^2 + \varepsilon \sum_{i=1}^{d} \sum_{j=1}^{n} (v_{i,j} - w_{i,j})^2.$$

whose value is now convex with respect to V. Indeed, if $\sum_{i,j} |v_{i,j}| \leq \eta$, it is easy to see that the value of both problems is zero, with $\bar{w} = v$ (while the value of $\bar{t}$ is no longer unique in this case in (P_ε)). On the other hand, when $\sum_{i,j} |v_{i,j}| > \eta$, then at least for one i one must have $\sum_j |v_{i,j}| > t_i$ so that, if for some other i', $\sum_j |v_{i',j}| < t_{i'}$, it is clear that decreasing slightly $t_{i'}$ and increasing t_i of the same amount, one can reduce the objective. Hence, in that case, the unique solution of (P_ε^+) is given by the solution of (P_ε) and the value is the same. This shows that in fact, the values of both problems are equal, and hence, also, convex with respect to V (since the objective in the (P_ε^+) is globally convex in (V, t, w)).

Lottery optimizer

Following the work of Frankle and Carbin [54], [55] proposed a simple algorithm to find sparse sub-networks within larger networks that are trainable from scratch. Their approach to finding these sparse 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 replaced the thresholding by our $\ell_{1,1}$ projection and devised the following algorithm:

Algorithm 4 Projection on the $\ell_{1,1}$ norm—$\mathrm{proj}_{\ell_1}(V, \eta)$ is the projection on the ℓ_1-ball of radius η, $\nabla L(W, M_0)$ is the masked gradient with binary mask M_0, and f is the ADAM optimizer, γ is the learning rate

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

IV. Experimental results

We used the pytorch framework to implement our sparse learning method using a constrained approach. We chose the Adam optimizer [56], a standard optimizer in PyTorch as baseline comparison to our optimizer with ℓ_1 and $\ell_{2,1}$ constraints.
We denoted as "PGL1", the algorithm with ℓ_1 constraint, "PGL21", the algorithm with $\ell_{2,1}$ constraint and "PGL11" the algorithm with $\ell_{1,1}$ constraint.
We used the entropy (bit/weight) of the weights distributions to compute estimations of the model storage memory cost. To this end, the weights can for instance be coded using JPEG2000[1] an image coding system that uses state-of-the-art compression techniques based on wavelet theory. The classical computational cost evaluates FLOPs (floating point operations) as a measure. When using FLOPs, additions (accumulates) and multiplications are counted separately. However, a lot of hardware can compute multiply-add operations in a single instruction. We therefore use **MACCs** (multiply-accumulate operations) as computational cost for which one multiplication and one addition are counted as a single instruction. We provide the results in normalized bytes and MACCs: we divided

[1] https://jpeg.org/jpeg2000/index.html

the number of bytes or MACCs by the number of bytes or MACCs obtained with Adam (i.e. without constraint). For all experiments we used Algorithm 4. Computation was performed on a Cocolink Klimax 210 HPC with 10 GPUs (Nvidia Quadro P6000, P100 and GeForce GTX 1080).

Results on MNIST with a convolutionnal Network

We selected the popular MNIST dataset [57] containing 28×28 grey-scale images of handwritten digits of 10 classes (from 0 to 9). This dataset consists of a training set of 60,000 instances and 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 $(1 \times 10 \times 5 \times 5)$, $(10 \times 20 \times 5 \times 5)$, (320×50) and (50×10) respectively. Thus the total number of elements of these weight matrices are 250, 5000, 16000 and 500 respectively.
To apply the $\ell_{1,1}$ constraint to the tensor, we unfolded the tensor in a matrix form. 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 did not apply sparsity constraint.

The sizes of matrix weights was very unbalanced. Thus our strategy was to sparsify the 2 most numerous layers, i.e. the 2^{nd} convolutional and the 1^{st} linear layer with the same optimizer.

We studied the layer-wise influence of the constraint parameter η on the accuracy and the weight sparsity, defined as the percentage of weights set to zero. For comparison purposes, the weights using Adam for the two linear layers are depicted in figure 2 (top shows unstructured sparsity, bottom shows structured sparsity using PGL11).

Figure 1 represents the weight distributions using a Parzen kernel method. The top part shows that weights distributions follow a Gaussian shape for Adagrad and the Adam optimizer. The bottom part shows that weights distributions follow a Laplacian shape (thus lower entropy) for PGL1 and PGL11.

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

Methods	Memory (kBytes)	MACCs (k-MACCs)	Accuracy (%)
Adam	33.64	480	99.
PGL1 $\eta = 80$	10.9	477	99.01
PGL11 $\eta = 40$	3.7	336	98.03
PGL11 $\eta = 25$	2.1	122	97.4

Results on MNIST with a Linear fully connected Network

We used a linear fully connected network (LFC4) with an input layer of d neurons, 4 hidden layers followed by a RELU activation function and a latent layer of dimension k.

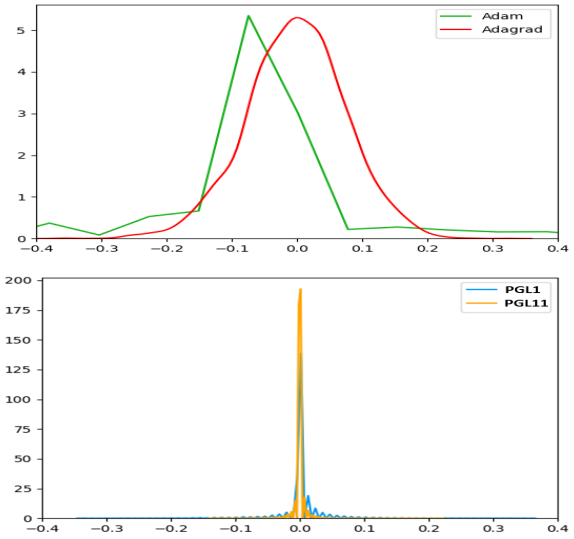

Fig. 1: MNIST, Net4, Top: Distribution of convolutional layer Conv2 with Adam and Adagrad optimizers, Bottom : Distribution of the convolutional layer Conv2 with PGL1 and PGL11 optimizers

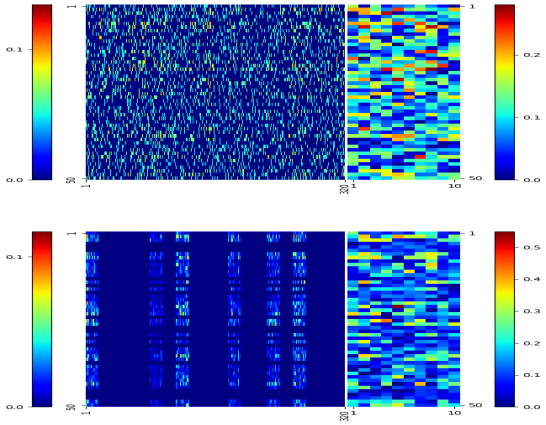

Fig. 2: Visualization for weight matrices: Top of the two linear layers of Net4 with Adam and weights thresholding shows unstructured sparsity. Bottom with structured optimizer: Layer Linear1 exhibits a high structured sparsity.

TABLE II: MNIST wth LFC4, total memory, MACCs, and accuracy

Methods	Memory (kBytes)	MACCs (k-MACCs)	Accuracy (%)
ADAM	3989	2379	98.3
PGL1($\eta = 200$)	438	1960	98.29
PGL11($\eta = 200$)	72	150	97.7
PGL11($\eta = 400$)	215	480	98.07
PGL21($\eta = 50$)	1810	1408	98.05

Figure 3, figure 4 and tables I and II[2] show that the main

[2]The projections PGL1, PGL11 and PGL21 being different, the impact of the projection parameter η is different on each of them. When comparing the projections (tables II and III), we provide parameter η such that the models PGL1, PGL11 and PGL21 have a similar accuracy.

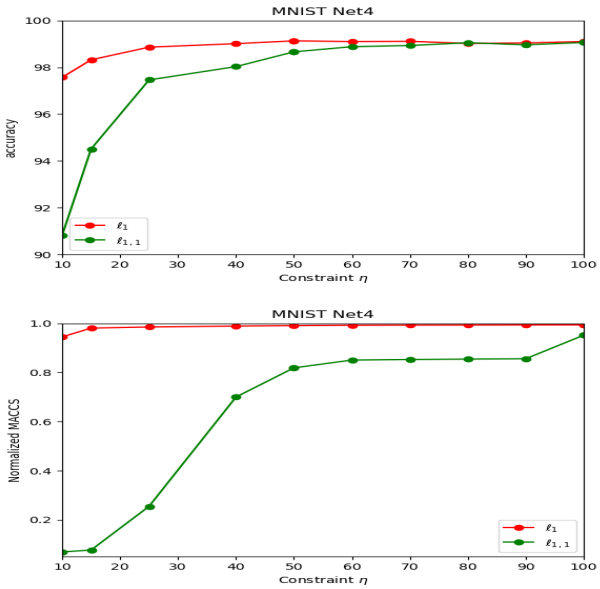

Fig. 3: MNIST with Net4: Top: Accuracy, Bottom: MACCS

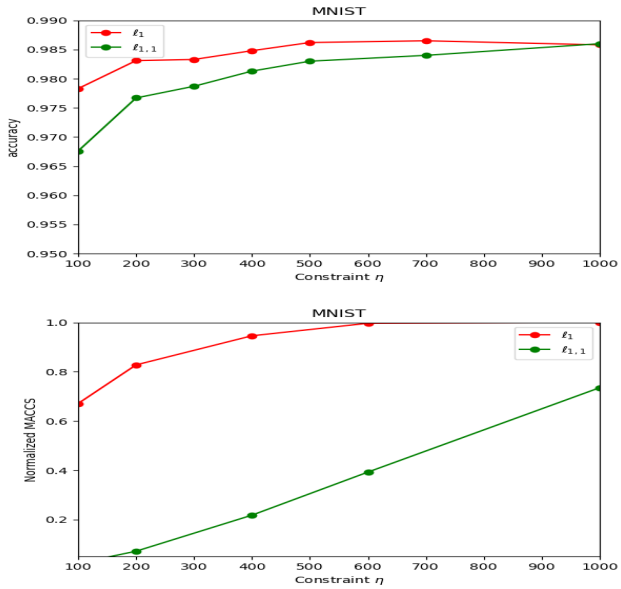

Fig. 4: MNIST with LFC4: Top: Accuracy, Bottom: MACCS

advantage of our method using the $\ell_{1,1}$ constraint over ℓ_1 is the reduction of the calculation cost (MACCs) by a factor 14 when using a LFC4 network which is crucial for low capacity devices such as smartphones. Note that performance in MACCs using the $\ell_{2,1}$ constraint is intermediate between the use of ℓ_1 and $\ell_{1,1}$.

Comparison with public results on MNIST using Le Net 300/100

Le Net 300/100 is a popular Linear Fully connected Network. Table III shows that our method outperforms the state-of-the-art [15] in terms of bytes accuracy compromise. Note that, to the best of our knowledge no results have been published in terms of FLOP reduction on this basis.

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

Methods	Memory (kBytes)	MACCs (k-MACCs)	Accuracy (%)
ADAM	477	266.2	98.21
PGL1($\eta = 200$)	61	189	98.03
PGL11($\eta = 200$)	32	62	96.4
PGL11($\eta = 400$)	83	150	97.8
PGL21 ($\eta = 50$)	164	257	98.1
Tartaglione [15]	33.7	-	96.6

Results on Fashion MNIST using a Linear Fully connected Network.

Fashion-MNIST [58] is a dataset from the publication by Zalando of 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 is 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 IV: Fashion MNIST LFC4 : Memory, MACCs, and accuracy

Methods	Memory (kBytes)	MACCs (k-MACCs)	Accuracy (%)
ADAM	3989	2379	89.9
PGL1($\eta = 400$)	567	2114	89.2
PGL11($\eta = 400$)	131	267	87.5

We observe a similar behavior to that of the previous experiment with the MNIST dataset. Table (IV) shows a large decrease in the global memory by a factor 9. On the other hand, we observed a decrease in the calculation cost by a factor of 9 for the $\ell_{1,1}$ constraint and a very small decrease for the ℓ_1 constraint.

Results on CIFAR10

The CIFAR-10 data set is composed of 60,000 32x32 color 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*[3], the highly optimized architecture [59]. This network is composed of 13 blocks $B_i = (convolutional2D/Batch\,Normalization/ReLU)$ for $i = 1, .., 13$ 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. Results are reported in Figures 5 and in the Table V.

Table (V) shows a large global decrease in memory by a factor of 10. On the other hand the decrease in the calculation cost was about 30% for the $\ell_{1,1}$ constraint and almost null for ℓ_1 constraint (Figure 5).

[3]https://github.com/Coderx7/SimpleNet_Pytorch

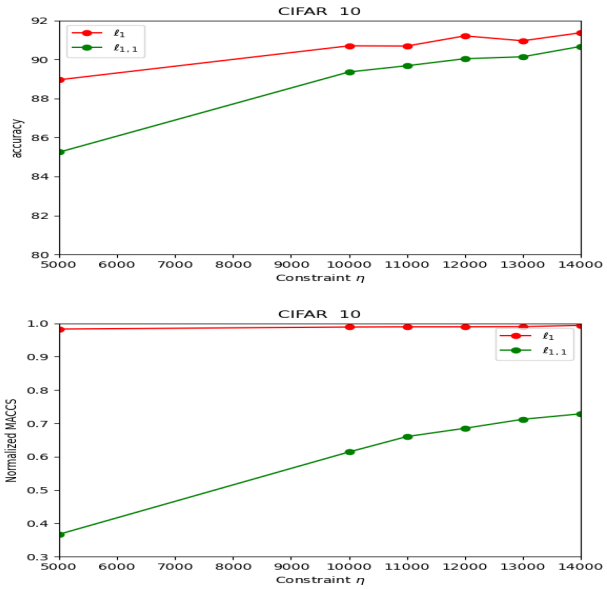

Fig. 5: CIFAR 10 SimpleNet, MACCs as a function of η, PGL1 (red) and PGL11 (green)

TABLE V: CIFAR10 total memory, and accuracy using Simplenet

Methods	MACCs (M-MACCs)	Memory (M-Bytes)	Accuracy %
Adam	631.51	9.44	93.8
PGL1 ($\eta = 13000$)	626.08	1.45	91.12
PGL11($\eta = 14000$)	441	0.86	91

V. Discussion

To the best of our knowledge, entropy of the weights (bit/weight) as a measure of memory has never been reported in the DNN literature. Thus comparison of memory with that of previous reports is not straightforward. A more in-depth study will be performed as well as the JPEG2000 compression of the model for storage and reported in a forthcoming paper. Energy consumption is directly related to the number of instructions (Flops or MACCs) [60]. Thus we report MACCs rather than Flops. The experimental results with MNIST and Fashion MNIST using Lenet 300/100 showed an improvement in memory by a factor 12.9 and 8 and by a factor 3 and 2 for computational power. We obtain on CIFAR10 a good trade-off between a 0.98% drop in accuracy and a substantial improvement in memory by a factor of 8.5 in comparison with the Adam optimizer.

Note that, in this paper, the different projections are not layer-wise optimized. The same constraint was applied to each layer of the network. There is *a priori* no reason to believe that the same constraint value is adapted to the sparsification of all the layers of a network.

VI. Conclusion

To deal with the computational issue associated to DNNs, a lot of studies into proximal regularization methods which are time consuming have been published. In this paper, we propose an alternative constrained approach. We provide a general framework with a new projection gradient method. We designed algorithms for the classical ℓ_1 constraint and the new $\ell_{1,1}$ constraint. Our experiments show the benefit of the Lottery optimizer that uses only one projection for deep neural network sparsification. Experiments using three popular datasets (MNIST, FASHION MNIST and CIFAR) show that our new projection method on the $\ell_{1,1}$ constraint provides better structured sparsity resulting in a substantial decrease in the cost of memory and of computation.

Furthermore, we are currently applying our method to other large Neural Networks.

Acknowledgement The authors would like to thank Antonin Chambolle for his contribution to the theoretical background of the $\ell_{1,1}$ "projection".

References

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

[2] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol. 7, pp. 19 143–19 165, 2019.

[3] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning in natural language processing," *arXiv:1807.10854*, 2018.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv 1605.07678*, 2016.

[6] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. Van Gool, "AI benchmark: Running deep neural networks on android smartphones," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[7] S. Rallapalli, H. Qiu, A. Bency, S. Karthikeyan, R. Govindan, B. Manjunath, and R. Urgaonkar, "Are very deep neural networks feasible on mobile devices," *IEEE Trans. Circ. Syst. Video Technol*, 2016.

[8] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. 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*. ACM, 2017, pp. 75–84.

[9] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," in *ACL*, 2019.

[10] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," 2019.

[11] M. Denil, B. Shakibi, L. Dinh, N. De Freitas *et al.*, "Predicting parameters in deep learning," in *Advances in neural information processing systems*, 2013, pp. 2148–2156.

[12] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization path for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, pp. 1–122, 2010.

[13] T. Hastie, R. Tibshirani, and M. Wainwright, "Statistcal learning with sparsity: The lasso and generalizations," *CRC Press*, 2015.

[14] 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*, 2015, pp. 1135–1143.

[15] E. Tartaglione, S. Lepsøy, A. Fiandrotti, and G. Francini, "Learning sparse neural networks via sensitivity-driven regularization," in *Advances in Neural Information Processing Systems*, 2018, pp. 3878–3888.

[16] A. N. Gomez, I. Zhang, K. Swersky, Y. Gal, and G. E. Hinton, "Learning sparse networks using targeted dropout," *arXiv :1905.13678*, 2019.

[17] M. Carreira-Perpiñán and Y. Idelbayev, "Learning-compression algorithms for neural net pruning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[18] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[19] 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*, 2017, pp. 1389–1397.

[20] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250*, 2016.

[21] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

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

[23] 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*, 2016, pp. 2074–2082.

[24] J. Friedman, T. Hastie, and R. Tibshirani, "A note on the group lasso and a sparse group lasso," *arXiv preprint arXiv:1001.0736*, 2010.

[25] A. Torfi, R. A. Shirvani, S. Soleymani, and N. M. Nasrabadi, "Attention-based guided structured sparsity of deep neural networks," *arXiv preprint arXiv:1802.09902*, 2018.

[26] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2736–2744.

[27] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv:1710.09282*, 2017.

[28] 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.

[29] J. Cavazza, P. Morerio, B. Haeffele, C. Lane, V. Murino, and R. Vidal, "Dropout as a low-rank regularizer for matrix factorization," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018, pp. 435–444.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[31] J. Moreau, "Proximité et dualité dans un espace hilbertien," *Bull. Soc.Math. France., 93*, pp. 273–299, 1965.

[32] P.-L. Lions and B. Mercier, "Splitting algorithms for the sum of two nonlinear operators," *SIAM Journal on Numerical Analysis*, vol. 16, no. 6, pp. 964–979, 1979.

[33] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 2011, pp. 185–212.

[34] 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*. Springer, 2010, pp. 418–433.

[35] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for Machine Learning*. MIT Press, 2012.

[36] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, "The entire regularization path for the support vector machine," *Journal of Machine Learning Research*, vol. 5, pp. 1391–1415, 2004.

[37] J. Mairal and B. Yu, "Complexity analysis of the lasso regularization path," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, 2012, pp. 353–360.

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

[39] J. M. Alvarez and M. Salzmann, "Learning the number of neurons in deep networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 2270–2278.

[40] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 304–320.

[41] J. Yoon and S. J. Hwang, "Combined group and exclusive sparsity for deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3958–3966.

[42] D. Zhang, H. Wang, M. Figueiredo, and L. Balzano, "Learning to share: Simultaneous parameter tying and sparsification in deep learning," 2018.

[43] U. Oswal, C. Cox, M. Lambon-Ralph, T. Rogers, and R. Nowak, "Representational similarity learning with application to brain networks," in *International Conference on Machine Learning*, 2016, pp. 1041–1049.

[44] D. Zhang, J. Katz-Samuels, M. A. Figueiredo, and L. Balzano, "Simultaneous sparsity and parameter tying for deep learning using ordered weighted ℓ_1 regularization," in *2018 IEEE Statistical Signal Processing Workshop (SSP)*. IEEE, 2018, pp. 65–69.

[45] M. Figueiredo and R. Nowak, "Ordered weighted l1 regularized regression with strongly correlated covariates: Theoretical aspects," in *Artificial Intelligence and Statistics*, 2016, pp. 930–938.

[46] S. Lin, R. Ji, Y. Li, C. Deng, and X. Li, "Toward compact convnets via structure-sparsity regularized filter pruning," *IEEE transactions on neural networks and learning systems*, 2019.

[47] L. Condat, "Fast projection onto the simplex and the l1 ball," *Mathematical Programming Series A*, vol. 158, no. 1, pp. 575–585, 2016.

[48] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, "Efficient projections onto the l 1-ball for learning in high dimensions," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 272–279.

[49] G. Perez, M. Barlaud, L. Fillatre, and J.-C. Régin, "A filtered bucket-clustering method for projection onto the simplex and the ℓ_1-ball," *Mathematical Programming*, May 2019.

[50] J. Liu, S. Ji, and J. Ye, "Multi-task feature learning via efficient l2, 1-norm minimization," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, ser. UAI '09. AUAI Press, 2009, pp. 339–348.

[51] M. Barlaud, A. Chambolle, and J.-B. Caillau, "Robust supervised classification and feature selection using a primal-dual method," *arXiv cs.LG/1902.01600*, 2019.

[52] M. Barlaud, W. Belhajali, P. L. Combettes, and L. Fillatre, "Classification and regression using an outer approximation projection-gradient method," vol. 65, no. 17, 2017, pp. 4635–4643.

[53] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[54] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations*, 2019.

[55] H. Zhou, J. Lan, R. Liu, and J. Yosinski, "Deconstructing lottery tickets: Zeros, signs, and the supermask," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 3597–3607.

[56] D. Kingma and J. Ba, "a method for stochastic optimization." *International Conference on Learning Representations , pages=1–13, year=2015,*.

[57] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*.

[58] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv cs.LG/1708.07747*, 2017.

[59] S. H. Hasanpour, M. Rouhani, M. Fayyaz, and M. Sabokrou, "Lets keep it simple, using simple architectures to outperform deeper and more complex architectures," *arXiv preprint arXiv:1608.06037*, 2016.

[60] E. Grochowski and M. Annavaram, "Energy per instruction trends in intel ® microprocessors," 2006.