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

Frederic Guyard, Michel Barlaud

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

Frédéric 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 high performances for DNNs were however often obtained with networks containing millions of parameters and for 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 a constrained approach instead. We provide the general framework for our new projection gradient method. Our algorithm iterates a gradient step and a projection on convex constraints. We study 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 use the recent "Lottery optimizer" replacing the threshold 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.

I. Motivation

In recent years, deep neural networks have been applied to different domains and achieved dramatic accuracy improvements in image recognition [31], speech recognition [41] or natural language processing [43]. These works rely on deep networks with millions or even billions of parameters. For instance, the original training of ResNet-50 [25] (image classification) contains 25.6M parameters and required 29 hours of processing 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 [5]. The recent development of DNNs, hardware accelerators like GPUs and the availability of deep learning frameworks for smartphones [29] suggest seamless transfer of DNN models trained on servers onto mobile devices. However, it turns out that memory [45] and energy consumption [20] are still the main bottlenecks on running DNNs on such devices.
Thus computational cost has an impact on carbon footprint. The authors of [50] argued that this trend is environmentally unfriendly. The authors of [47] 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. 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 give experimental comparisons between 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 [52], [23], [16], [24], [1]. The ℓ_1 penalty added to the classification cost can be interpreted as a convexification of the ℓ_0 penalty [12]. In [21], weights with the smallest amplitude in pre-trained networks are removed. Model sensitivity to weights can also be used [51], [18]. where weights with weak influence on network output are pruned. Constraint optimization is used in [6] in order to learn sparse networks with ℓ_0, ℓ_1 or ℓ_2 constraints on the weights.

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 [33]. [26] performs channel pruning using LASSO regression and least squared reconstruction. Neurons are pruned based on the average percentage of zeros (APoZ) after the ReLU activation [27].

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* $\ell_{2,1}$ [57] or on *sparse group LASSO* $\ell_{2,1} + \ell_1$ [16] 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 [46]. Group LASSO regularization is used, in [46], [54]. Group LASSO and filter decorrelation regularization are used in order to discard CNN filters in [17]. Group LASSO and group variance regularization are used in [53].

In order to deal with non-smooth ℓ_1 regularization, subgradient descent is used in [37]. Here, structured sparsification is performed without group regularization.

The idea is to scale 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) [8]. Nuclear norm penalty was successfully used in matrix low rank approximation [8], matrix completion [59], matrix factorization [7] and DNN dropout modeling [49].

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 $f(x)$ [39]:

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

Let W be the weight matrix of a neural network, $L(W)$ be a gradient Lipschitz loss and $R(w)$ be a convex penalty. Lets 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 [35], [9], [40], [48]. Using SGD with penalization is limited and time consuming due to the tuning of the corresponding penalization hyper-parameters [23], [38].

Proximal gradient descent with group LASSO constraint is used in [62] and in [2]. In [28], the output neurons is scaled using a factor λ_i and accelerated proximal gradient is used with ℓ_1 constraint to push as many coefficients λ_i towards 0 without significantly decreasing the performance. In [56], 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 [61] proximal gradient descent with group OWL constraint (grOWL [42]) is used in order to simultaneously sparsify neurons and enforce parameter sharing. Proximal gradient descent is also used in [60] where Ordered Weighted ℓ_1 regularization (OWL [14]) allowing to simulatenously sparsify weights and optimize weight sharing. In [34], 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 an available efficient projection on the ℓ_1-ball [10], [13], [44], projection on the $\ell_{2,1}$ ball [36], [4] 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 ℓ_1-ball [10], [44].

Let $L(W)$ be a gradient Lipschitz loss, $R(w)$ be a convex constraint, and C its convex set. Lets 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 [3]:

$$V_n := W_n - \gamma \nabla L(W_n), \tag{3}$$

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

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 $R(w) = \|w_i\|_1$, efficient algorithms have been proposed [10], [44]. Unfortunately this ℓ_1 constraint does not induce a sparse structure.

Classical Optimizer with $\ell_{2,1}$ norm constraint (Group LASSO): The Group LASSO was first introduced in [58]. 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 $d \times k$ 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 [4] 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 η: compute t_i which is the projection of the vector $(\|v_i\|_i)_{i=1}^d$ on the ℓ_1 ball of $\mathbf{R}^n$ of radius η; then, each row of the projection is obtained according to

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

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

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

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 [36]. The main drawback of their method is to compute the roots of an equation using bisection, which is quite slow.

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 $\mathrm{x} \in \mathbf{R}_*^n$, the value of $\| (A-B)\mathrm{x} \|$ 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 $\sum_{i=1}^n \| B^i \|_1 < \sum_{i=1} \| A^i \|_1$ where B_i and A_i are the i th column of matrices B and A respectively. Let's recall the induced operator norm of $A - B$ in ℓ_1 domain with ℓ_1 co-domain

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

which is computed as the maximum ℓ_1 norm of a columns of $A - B$. We therefore consider the following problem

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

It is possible to show the following result:

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

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

Applying Theorem 1 to (6) we obtain the following algorithm: we first compute the radius t_i and then project the rows using ℓ_1 adaptive constraint t_i:

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

Lottery optimizer

Following the work of Frankle and Carbin [15], [63] proposed a simple algorithm for finding 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—$\text{proj}_{\ell_1}(V, \eta)$ 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
$t := \text{proj}_{\ell_1}((\|v_i\|_1)_{i=1}^d, \eta)$
for $i = 1, \ldots, d$ **do**
  $w_i := \text{proj}_{\ell_1}(v_i, t_i)$
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 [30], a standard optimizer in PyTorch as baseline comparison to our optimizer with ℓ_1 and $\ell_{2,1}$ constraints.
We denote 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 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[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 divide the number of bytes or MACCs by the number of bytes or MACCs obtained with Adam (i.e. without constraint). For all experiments we use Algorithm 4. Computation is performed on a Cocolink Klimax 210 HPC with 10 GPUs (Nvidia Quadro P6000, P100 and GeForce GTX 1080).

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

Results on MNIST with a convolutionnal Network

We selected the popular MNIST dataset [32] 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 unfold 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 do not apply sparsity constraint.

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

We study the layerwise 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).

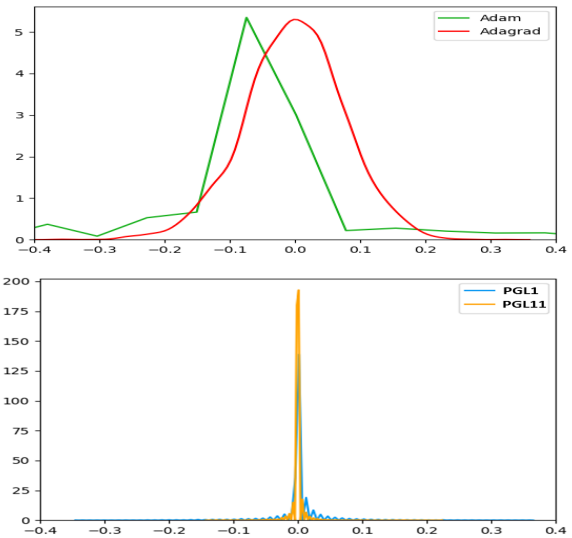

Fig. 1: MNIST, Net4, Top: Distribution of convolutional layer Conv2 with Adam and Adagrad optimizers, Bottom : Distribution of convolutional layer Conv2 with PGL1 and PGL11 optimizers

(Fig. 1) represents the weight distributions using a Parzen kernel method. The top part shows that weights distributions follow a Gaussian shape for Adagrad and Adam optimizer. The bottom part shows that weights distributions follow a

Laplacian shape (thus lower entropy) for PGL1 and PGL11.

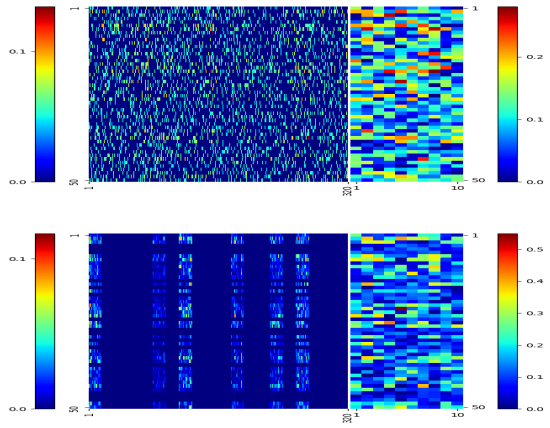

Fig. 2: Visualization for weight matrices: Top of the two linear layers of Net4 with Adam and weights thresholding shows unstructured sparsity. Bottom with structured optimizer: Layer Linear1 exhibits a high structured sparsity.

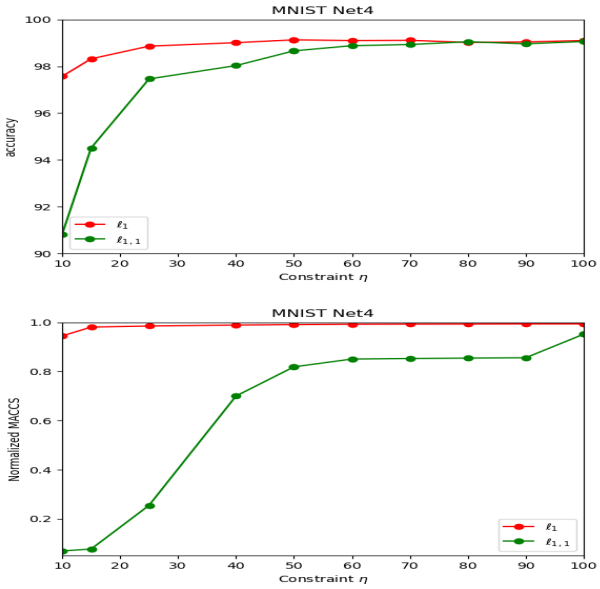

Fig. 3: MNIST with Net4: Top: Accuracy, Bottom: MACCS

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.

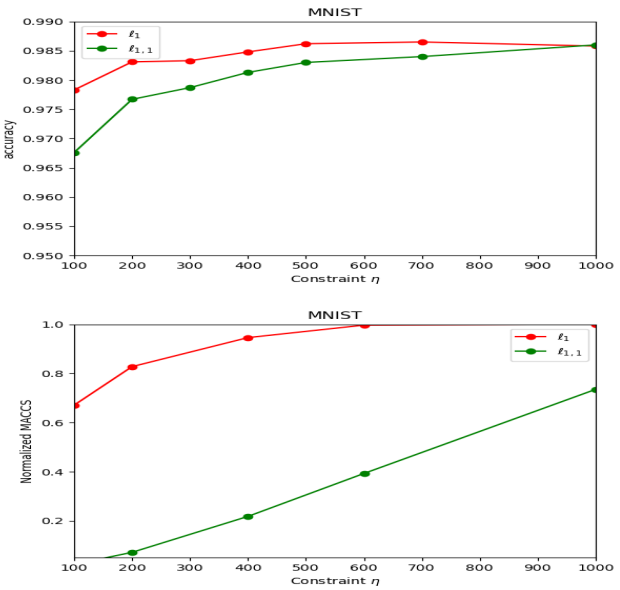

Fig. 4: MNIST with LFC4: Top: Accuracy, Bottom: MACCS

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 show that the main 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 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

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 [51]	33.7	-	96.6

Le Net 300/100 is a popular Linear Fully connected Network. The table III shows that our method outperforms the state of the art [51] 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.

Results on Fashion MNIST using a Linear Fully connected Network.

Fashion-MNIST [55] 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 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.

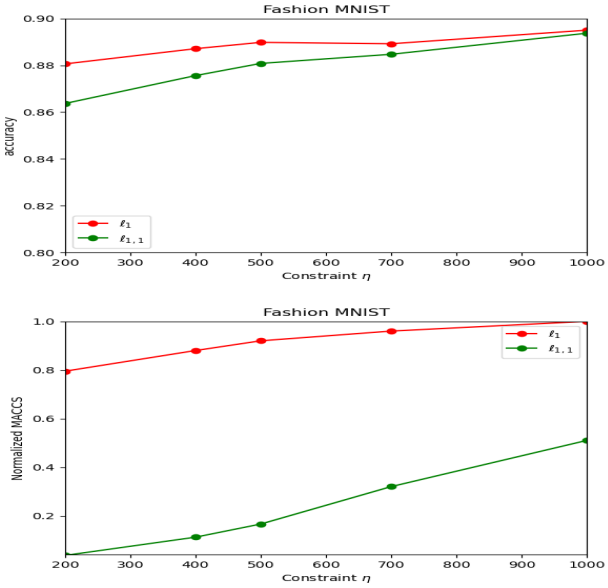

Fig. 5: Fashion MNIST with LFC4:Top Accuracy , Bottom MACCs

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 behaviour similar to the previous experience on MNIST dataset. Table (IV) shows a large global memory reduction by a factor 9. On the other hand, we observe reduction of the calculation cost by a factor 9 for $\ell_{1,1}$ constraint and very small for ℓ_1 constraint.

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*[2], the highly optimized architecture [22]. 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 7 and in the Table V.

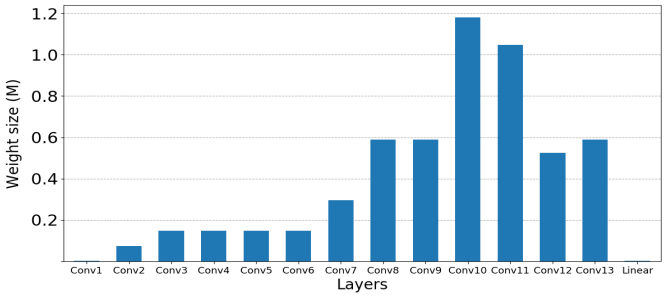

Fig. 6: SimpleNet, weight size for each layer

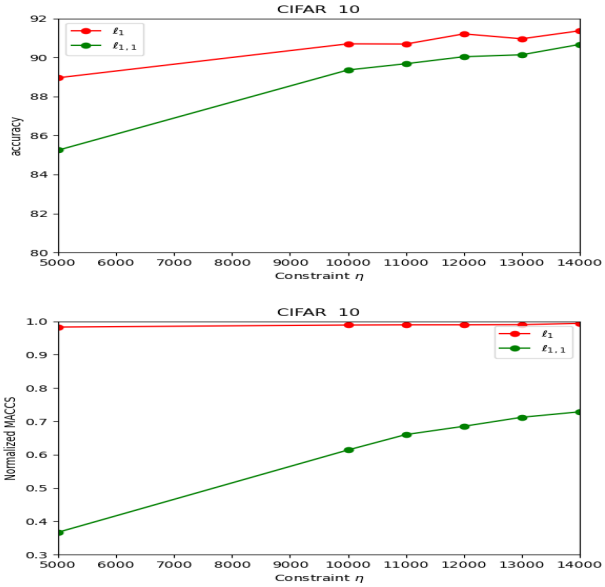

Fig. 7: CIFAR 10 SimpleNet, MACCs as a function of η, PGL1 (red) and PGL11 (green)

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

Table (V) show a large global memory reduction by a factor 10. On the other hand of the calculation cost the reduction is

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

about 30% for $\ell_{1,1}$ constraint and almost null for ℓ_1 constraint (Figure 7).

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 for storage in a forthcoming paper.

Energy consumption is directly related to the number of instructions (Flops or MACCs) [19]. Thus we report MACCs rather than Flops. Experimental results show, as expected, that our new $\ell_{1,1}$ norm constraint is more efficient for structured sparsity and thus MACCs reduction than ℓ_1 constraint.

Experimental results on MNIST and Fashion MNIST using Lenet 300/100 show a memory improvement by a factor 12.9 and 8 and a computational improvement by a factor 3 and 2 of computational power respectively. We obtain on CIFAR10 a good trade-off between 0.98% accuracy drop and large storage memory improvement by a factor 8.5 in comparison with the Adam optimizer.

Algorithm 4 for projecting a matrix on the constraint $\ell_{1,1}$ can be extended to the projection of a tensor on generalized constraints of the form $\ell_{1,1,1,\ldots}$. A straightforward improvement could be obtained using layer-wise adaptive constraints (η).

Remark that, in this paper, the various projections are not layer-wise optimized. The same constraint is applied to each layer of the network. There are *a priori* no reasons to believe that the same constraint value is adapted to the sparsification of all the layers of a network. In particular, layers with large ℓ_1 norm may need small constraint value η in order to significantly decrease the MACCs whereas such small η may be detrimental to other layers with small ℓ_1 norm. This leaves place for further optimization where the size of the constraint is layer-wise dependant. This layer-wise adaptation will be investigated in a forthcoming study.

VI. Conclusion

In order to cope with the computational issue associated to DNNs, a huge literature deals with proximal regularization methods which are time consuming. In this paper, we propose an alternative constrained approach. We provide a general framework of our new projection gradient method. We design algorithms for the classical ℓ_1 constraint and the new $\ell_{1,1}$ constraint. Our experiments show the benefits of Lottery optimizer which uses only one projection for deep neural networks sparsification. Experiments on 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 high reduction of memory cost and computational cost.

Furthermore, we are currently 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] Michel Barlaud, Wafa Belhajali, Patrick L. Combettes, and Lionel Fillatre. Classification and regression using an outer approximation projection-gradient method. volume 65, pages 4635–4643, 2017.

[4] Michel Barlaud, Antonin Chambolle, and Jean-Baptiste Caillau. Robust supervised classification and feature selection using a primal-dual method. *arXiv cs.LG/1902.01600*, 2019.

[5] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv 1605.07678*, 2016.

[6] 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.

[7] Jacopo Cavazza, Pietro Morerio, Benjamin Haeffele, 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.

[8] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv:1710.09282*, 2017.

[9] 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.

[10] L. Condat. Fast projection onto the simplex and the l1 ball. *Mathematical Programming Series A*, 158(1):575–585, 2016.

[11] Misha Denil, Babak Shakibi, Laurent Dinh, Nando De Freitas, et al. Predicting parameters in deep learning. In *Advances in neural information processing systems*, pages 2148–2156, 2013.

[12] 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.

[13] 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*, pages 272–279. ACM, 2008.

[14] Mario Figueiredo and Robert Nowak. Ordered weighted l1 regularized regression with strongly correlated covariates: Theoretical aspects. In *Artificial Intelligence and Statistics*, pages 930–938, 2016.

[15] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

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

[17] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

[18] Aidan N Gomez, Ivan Zhang, Kevin Swersky, Yarin Gal, and Geoffrey E Hinton. Learning sparse networks using targeted dropout. *arXiv :1905.13678*, 2019.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

[31] 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.

[32] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

[33] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[34] 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.

[35] 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.

[36] Jun Liu, Shuiwang Ji, and Jieping Ye. Multi-task feature learning via efficient l2, 1-norm minimization. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 339–348. AUAI Press, 2009.

[37] 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.

[38] 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.

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] Guillaume Perez, Michel Barlaud, Lionel Fillatre, and Jean-Charles Régin. A filtered bucket-clustering method for projection onto the simplex and the ℓ_1-ball. *Mathematical Programming*, May 2019.

[45] 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.

[46] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

[47] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai, 2019.

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

[49] 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.

[50] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *ACL*, 2019.

[51] Enzo Tartaglione, Skjalg Lepsøy, Attilio Fiandrotti, and Gianluca Francini. Learning sparse neural networks via sensitivity-driven regularization. In *Advances in Neural Information Processing Systems*, pages 3878–3888, 2018.

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

[53] 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.

[54] 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.

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

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] Dejiao Zhang, Julian Katz-Samuels, Mário AT Figueiredo, and Laura Balzano. Simultaneous sparsity and parameter tying for deep learning using ordered weighted ℓ_1 regularization. In *2018 IEEE Statistical Signal Processing Workshop (SSP)*, pages 65–69. IEEE, 2018.

[61] Dejiao Zhang, Haozhu Wang, Mario Figueiredo, and Laura Balzano. Learning to share: Simultaneous parameter tying and sparsification in deep learning. 2018.

[62] 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.

[63] 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.