ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Lightening CNN architectures by regularization driven weights' pruning

Giovanni Bonetta[1] and Rossella Cancelliere[1]

1- University of Turin - Department of Computer Science
via Pessinetto 12 - 10149 Turin, Italy

Abstract. Deep learning models are getting increasingly big, leading towards overparametrized architectures with high computational and storage requirements. This hinders the possibility to train/deploy them on IoT or mobile devices, while also creating concerns about their environmental fingerprint. We propose a regularization technique which allows to selectively shrink the norm of non significant weights in order to subsequently prune them, generating highly compressed models. We tested the proposed technique on three well known image classification tasks, obtaining results on par or better than competitors in terms of sparsity and metrics.

1 Introduction

Nowadays big deep learning models are constantly pushing the limits of what can be achieved in the Artificial Intelligence domain, but that comes at the price of increasing the number of parameters, which now can even get to hundreds of billions. This request for huge resources leads to increasing training and deploying costs and worsens the ecological impact on the planet. A sensible way to tackle the problem of model's complexity and dimensions is via weight sparsity, which is the property a model has when a substantial number of its weights have zero value. Sparsity allows to find the core, salient parameters of the network and is well tolerated by deep architectures, as shown for example in [1] and [2]. The advantages are: smaller computational and storage requirements, and improved performance through overfitting control.

Historically techniques based on the $L2$ regularization are among the most popular ways to achieve highly sparsifying models. A central drawback of such algorithms is that they do not directly account for weight relevance in the neural architecture (see [3, 4]), but the entire set of parameters is forced towards very small values.

Aiming at finding a solution to this issue, in this paper we outline and implement a new loss functional which, thanks to a suitable regularization term, allows to account for the actual contribution a weight has on the model loss. As a result, the norm of non relevant weights is selectively decreased, while a standard weight decay update is done for relevant ones. So doing the definition of ad hoc weights' updates (see [1, 3, 4, 5]) is no more required because weights' shrinking directly follows from loss optimization. Shrunk weights are then pruned in order to sparsify the neural architecture.

One important aspect of our proposed regularization term is that it can be

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

used in any loss functional regardless of its form, and constitutes a unified framework potentially exploitable for many different architectures and applications.

We show the effectiveness of our method sparsifying different convolutional neural models, establishing, at the best of our knowledge, new state-of-the-art performance in two out of three image classification tasks.

The paper is organized as follows: in Section 2 the theoretical foundations and a detailed description of our pruning algorithm are presented; Sections 3 describes the datasets and analyzes implementation and results.

2 The pruning method: theory and algorithm

$L2$ norm penalty, also known as Tikhonov regularization [6] or weight decay is often used to turn an original unstable, ill-posed problem into a well-posed one. In a neural context, the weight decay loss $\tilde{L}$ depends on each weight $w^n_{i,j}$ belonging to layer n and connecting neurons i and j:

$$\tilde{L}(\bar{w}) = L(\bar{w}) + \lambda\|\bar{w}\|^2 = L(\bar{w}) + \lambda \sum_{n,i,j} |w^n_{i,j}|^2.$$

$\bar{w}$ is the vector whose elements are $w^n_{i,j}$ and λ is the regularization parameter.

In this context we propose a new loss functional $\hat{L}(\bar{w})$, characterized by a coefficient that measures how much the final loss value is influenced by modification of a weight. Consequently, the magnitudes of only those weights which are not important for the final loss are selectively shrunk.

In order to reach this objective we focus on the quantity $|\frac{\partial L}{\partial w^n_{i,j}}|$: small derivative values indicate that changes in the weights do not influence the final loss value while large derivative values characterize relevant weights that do not need to be shrunk. We therefore propose the following modified loss functional:

$$\hat{L}(\bar{w}) = L(\bar{w}) + \lambda \sum_{n,i,j} C \cdot |w^n_{i,j}|^2, \qquad C \equiv \frac{1}{1 + |\frac{\partial L}{\partial w^n_{i,j}}|} \in (0, 1]$$

where the coefficient C assumes values near $1/0$ for non relevant/relevant weights. If for example stochastic gradient descent is used to optimize $\hat{L}$, the new weights' update rule is:

$$\Delta w^n_{i,j} \equiv -\eta \frac{\partial \hat{\mathrm{L}}}{\partial w^n_{i,j}} = -\eta \frac{\partial \mathrm{L}}{\partial w^n_{i,j}} - 2\eta\lambda \frac{w^n_{i,j}}{1 + |\frac{\partial L}{\partial w^n_{i,j}}|} + \eta\lambda |w^n_{i,j}|^2 \frac{-\mathrm{sgn}(|\frac{\partial^2 L}{\partial {w^n_{i,j}}^2}|)}{(1 + |\frac{\partial L}{\partial w^n_{i,j}}|)^2} \quad (1)$$

(η: learning rate, sgn: sign function). If the second-order derivative term is neglected, as usually done in second order derivative methods, eq. (1) becomes:

$$\Delta w^n_{i,j} = -\eta \frac{\partial \mathrm{L}}{\partial w^n_{i,j}} - 2\eta\lambda \frac{w^n_{i,j}}{1 + |\frac{\partial L}{\partial w^n_{i,j}}|} \quad (2)$$

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Different weights' updates are made in the two cases determined by the extreme values of $|\frac{\partial L}{\partial w^n_{i,j}}|$:
1) $|\frac{\partial L}{\partial w^n_{i,j}}| \sim 0$: in this case the weight is not relevant. The first term in eq. (2) is roughly zero, and the update rule becomes $\Delta w^n_{i,j}(t) \simeq -2\eta\lambda w^n_{i,j}$. The weight is actually driven to zero, because if $w^n_{i,j} > 0$ then $\Delta w^n_{i,j} < 0$, so that the norm of a positive weight is decreased at each iteration. Similarly, the norm of a negative weight is also reduced.
2) $|\frac{\partial L}{\partial w^n_{i,j}}| \gg 0$: in this case the weight is relevant. The second term in eq. (2) is roughly zero, so a classic update of $w^n_{i,j}$ is performed.

Our pruning algorithm follows these steps:

1. We get a checkpoint to finetune from, founding it in literature or training it on our own. A randomly initialized checkpoint can be used too.

2. The checkpoint is finetuned using our regularized loss. We remark that in this step any optimizer can be used and that gradients have to be computed twice for weight updates; a computational overhead is however common to many approaches that aim to select the weights to be pruned (see [2, 3, 4, 5]). Model validation performance is evaluated each *evaluation interval* steps, and we:

- prune. If the validation accuracy is higher than a user-defined *lower-bound*, chosen as slightly lower than the state-of-the-art performance, a fixed percentage (*pruning-percentage*) of the weights with lowest magnitude is pruned.

- not prune. If the validation performance is lower than the user-defined lower-bound the finetuning proceeds normally.

3. Step 2 is iteratively repeated until the model reaches a validation performance plateau.

4. We perform a final finetuning phase without regularization, aiming to get the best performing checkpoint.

In order to avoid accuracy plateau we decay λ exponentially between two validation assessments. Grid search is used to set the hyperparameters for training and finetuning.

3 Implementation details and results

We experiment on three computer vision datasets often used in sparsity research: MNIST [7], Fashion-MNIST [8] and ImageNet [9].

We used one Nvidia TITAN RTX 24Gb GPU, spending ~ 5' for training MNIST and Fashion-MNIST, and ~ 7 days for training ImageNet. The used implementation framework is Pytorch (version 1.10.2).

Code is available at https://github.com/giobin/esann22_sparsity.

As a first result, in table 1 we show the disk space occupation after processing the models with BZIP2[1]; the sparsification induced by our algorithm is exploited for high compression. Note that the compression is at least 9x for VGG-16 and 15x for LeNet-5.

[1] https://www.sourceware.org/bzip2/

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

	Lenet5 (MNIST)	Lenet5 (F-MNIST)	VGG-16 (ImageNet)
Unpruned	1.6 MB	1.6 MB	525.5 MB
Pruned	0.07 MB	0.10 MB	58.9 MB

Table 1: Disk space dimensions of pruned/unpruned models.

LeNet-5 on MNIST

This network, originally introduced by [11], has been lately modified and its more used version consists of a Convolution (Conv) layer with 6 5x5 filters, a 2x2 Pooling layer, a Conv layer with 10 5x5 filters, another 2x2 Pooling layer and two fully connected (FC) layers of sizes (800, 500) and (500, 10) respectively, for a total of 431080 model weights. We stick with this choice in order to compare with state-of-the-art results.

Model	epochs	batch	η	optim.	λ	lower bound	pruning percentage	eval. interval
LeNet	120	100	0.001	Adam	0.001	98.7	4%	250
VGG16	30	100	0.0001	SGD	1e-4	68.5	1%	2500

Table 2: Hyperparameters' values. Lower bound, pruning percentage and eval. interval parameters have been defined in the previous Section.

For this dataset the initial training phase can be skipped directly regularizing and pruning a randomly initialized checkpoint, then we finetune without regularization for 5 epochs. Hyperparameter values are resumed in table 2.

Table 3 compares our results with state-of-the-art, not sparsified checkpoint (Baseline), and recent results obtained for this task in terms of "Accuracy" of the classification and of "Sparsity", i.e. the percentage of pruned weights w.r.t. the initial number of weights. From this definition follows that (100 - Sparsity) is the percentage of remaining weights. We also report the percentage of "Residual Weights" for each layer of the pruned architecture, that is always the lowest w.r.t all competitors, with the only exception of Conv1 layer for Sparse VD.

Methods	Residual Weights (%)				Accuracy (%)	Sparsity (%)
	Conv1	Conv2	FC1	FC2		
Baseline	100	100	100	100	99.32	–
Han et al., 2015 [1]	66	12	8	19	99.23	91.6
Tart. et al., 2018 [3]	67.6	11.8	0.9	31.0	99.22	98.0
Ullr. et al., 2017 [2]	-	-	-	-	99.03	99.4
Tart. et al., 2021 [4]	22	2.38	0.22	5.98	99.21	99.6
Sparse VD [12]	33	2	0.2	5	**99.25**	99.6
Our method	30	2	0.1	3.8	99.22	**99.7**

Table 3: Test results for LeNet-5 architecture on MNIST.

Nowadays accuracy and sparsification performance on MNIST are close to the best possible for all techniques. Our method's non zero residual weights are

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Methods	Residual Weights (%)				Accuracy (%)	Sparsity (%)
	Conv1	Conv2	FC1	FC2		
Baseline	100	100	100	100	91.90	–
Tart. et al., 2018 [3]	76.2	32.56	6.5	44.02	91.50	91.5
Han et al., 2015 [1]	-	-	-	-	91.56	93.0
Tart. et al., 2021 [4]	78.6	26.13	2.88	32.66	91.53	95.7
Our method	78.9	18.85	1.52	6.7	**91.67**	**96.9**

Table 4: Test results for LeNet-5 architecture on Fashion-MNIST.

less than 1,5k, mainly because of the fully connected layers sparsification. We can note that we obtain almost the same accuracy of Han et al., but with 8% less weights.

LeNet-5 on Fashion-MNIST

We train for 24 epochs to have the initial checkpoint for Fashion-MNIST[2]. For finetuning with regularization we use the same MNIST hyperparameters (listed in table 2), except for lower-bound = 89.7 and epochs = 81; finally, we finetune without regularization for 50 additional epochs. Results are shown in table 4, and we can see that our method reaches the best performance both in terms of accuracy and sparsity.

VGG-16 on ImageNet

The model is composed by 13 Convolutional layers with 3x3 filters followed by Relu activations. These layers are interleaved by 5 MaxPooling layers. Finally, an AdaptiveAveragePool layer with 7x7 dimension is used before a sequence of 3 Linear layers with Relu activations which lead to the 1000-dimensional output, for a total of 138,357,544 weights. In our experiment on ImageNet [3] we start a regularized finetuning from a pretrained checkpoint[4] using the hyperparameters shown in table 2. We further finetune for 30 epochs without regularization to get the best accuracy.

Method	Residual Weights (%)		Accuracy (%)	Sparsity (%)
	Convolutional	FC		
Baseline	100	100	71.30	–
Han et al., 2015 [1]	32.77	4.61	68.66	92.5
Tart. et al., 2018 [3]	56.49	2.56	69.08	**92.9**
Our model	38.12	4.01	**69.48**	92.6

Table 5: Test results for VGG-16 architecture on ImageNet.

[2] 70,000 28x28 gray-scale images of fashion products from Zalando website. train/test 60k/10k.

[3] 14 million hand-annotated RGB images arranged according to the WordNet [10] noun hierarchy with over 20k categories. One million of those images come with bounding boxes. The average image size is 469x387 pixels, but it is usually down-sampled to 256x256.

[4] https://pytorch.org/vision/stable/models/generated/torchvision.models.vgg16.html#torchvision.models.VGG16_Weights

ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 5-7 October 2022, i6doc.com publ., ISBN 978287587084-1. Available from http://www.i6doc.com/en/.

Results show that VGG-16 is highly overparameterized for the task, since all methods are able to prune almost 93% of the network weights, shrinking them from ∼138M to less than 10M. Our method confirms its ability to scale to a bigger model (∼100M weights) obtaining the best accuracy among the shown methods with a sparsity percentage very close to the best one.

4 Conclusions

In this paper we proposed a new sparsification technique that focus on pruning non relevant weights, with results on par or better than competitors in terms of sparsity and accuracy. Besides, promising performances are obtained also measuring the architectural compression of the final models. As future work, we plan to apply these ideas to other tasks and architectures, for instance in the framework of natural language processing.

References

[1] S. Han, J. Pool, J. Tran, W. J. Dally, Learning both weights and connections for efficient neural network, in: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems (NIPS) 28, Curran Associates, Inc., Montreal, Canada, 2015, pp. 1135–1143.

[2] K. Ullrich, E. Meeds, M. Welling, Soft weight-sharing for neural network compression, in: 5th International Conference on Learning Representations, (ICLR), Toulon, France, OpenReview.net, 2017.

[3] E. Tartaglione, S. Lepsø y, A. Fiandrotti, G. Francini, Learning sparse neural networks via sensitivity-driven regularization, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems (NIPS), 31, Curran Associates, Inc., Montreal, Canada, 2018.

[4] E. Tartaglione, A. Bragagnolo, A. Fiandrotti, M. Grangetto, LOss-Based SensiTivity rEgulaRization: towards deep sparse neural networks, Neural Networks 146, 2022, pp. 230–237.

[5] A. N. Gomez, I. Zhang, K. Swersky, Y. Gal, G. E. Hinton, Learning sparse networks using targeted dropout, CoRR abs/1905.13678, 2019.

[6] A. N. Tikhonov, Solution of incorrectly formulated problems and the regularization method, Soviet Math. Dokl. 4, 1963, pp. 1035–1038.

[7] Y. LeCun, C. Cortes, MNIST handwritten digit database, 1990.

[8] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, CoRR abs/1708.07747, 2017.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: IEEE conference on computer vision and pattern recognition, 2009, pp. 248–255.

[10] G. A. Miller, Wordnet: A lexical database for english, Commun. ACM 38 (11), 1995, pp. 39–41.

[11] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Computation 1 (4), 1989, pp. 541–551.

[12] D. Molchanov, A. Ashukha, D. P. Vetrov, Variational dropout sparsifies deep neural networks, in: Proceedings of the 34th International Conference on Machine Learning, (ICML), D. Precup, Y. W. Teh (Eds.), Sydney, Australia, Vol. 70, 2017, pp. 2498–2507.