Structured Pruning in Deep Neural Networks with Trainable Probability Masks

Structured Pruning in Deep Neural Networks with Trainable Probability Masks

F. Martinini *, A. Enttsel *, A. Marchioni *†, M. Mangia *†, R. Rovatti *†, G. Setti ‡†
*DEI, †ARCES, University of Bologna, Italy -
{filippo.martinini, andriy.enttsel, alex.marchioni, mauro.mangia, riccardo.rovatti}@unibo.it
‡DET, Politecnico di Torino, Italy - gianluca.setti@polito.it

***Abstract*—The current trend of over-parameterized Deep Neural Networks makes the deployment on resource constrained systems challenging. To deal with this, optimization techniques, such as network *pruning*, can be adopted. We propose a novel pruning technique based on trainable probability masks that, when binarized, select the elements of the network to prune.**

Our method features *i*) an automatic selections of the elements to prune by jointly training the binary masks with the model, *ii*) the capability of controlling the pruning level through hyper-parameters of a novel regularization term. We assess the effectiveness of our method by employing it in the structured pruning of the fully connected layers of shallow and deep neural networks where it outperforms the magnitude-based pruning approaches.

***Index Terms*—Deep Neural Network, Probability Mask, Trainable Binary Mask, Network Pruning**

I. Introduction

Over the past years machine learning approaches have been adopted extensively in a variety of applications to solve effectively all kinds of domain related problems. This breakthrough in large part is due to a nice property of Deep Neural Networks (DNNs) for which controlled over-parametrization of the model usually helps in reaching the desired performance levels of the executed task [1], [2]. This approach, on the downside, brings to an increased computational effort and a larger memory occupation. Consequently, over-parametrization alone is not much sustainable in several application domains. For example, to deploy NN models on resource constrained edge devices, or to make inference of models having hundreds of billions of parameters e.g., transformer-based networks used for Natural Language Processing (NLP) [3], over-parametrization has to be coupled with other optimization techniques [4]–[6]. One of such methods is over-parametrization followed by the so called *pruning* step, which allows to shrink the designed model in size without compromising its performance. In the last years, pruning has received much attention and in the literature one can find a multitude of works reporting different techniques to accomplish it [5], [7]–[9]. For instance, to prune transformer-based networks the so called one-shot pruning techniques have been developed [6].

Most approaches for pruning consist in the combination of two main operations: model reduction and parameters search. The former procedure builds a smaller architecture and the latter looks for a suitable new set of parameters (e.g., weights, bias) that approximates the behaviour of the unpruned model. The existing pruning approaches either pursue the parameter search concurrently to the architectural search [10] or introduce an additional training step after a proper architecture has been found [11], [12].

In the most general case, pruning can target single weights or entire structural elements such as neurons or filters. Weight pruning aims at canceling the contribution of a certain amount of weights by forcing sparsification during the parameters search. For instance, to achieve weight pruning one can add a regularization term to the loss function, e.g., the ℓ_1-norm of all network parameters [8]. As a result, weight pruning reduces memory footprint but introduces irregularities in the structure, partially precluding an effective reduction of the computational effort. On the other hand, structured pruning succeeds in cutting down both the memory footprint and computational time by eliminating entire structure elements [7], [13].

In this work, we focus on structured pruning and propose a novel method[1] that couples structural elements with an entry of a trainable probability mask. The use of probability masks in the DNN context is not new, for example in [14], [15] masks have been adopted to speed up the MRI acquisitions. With a simultaneous training of the model and the masks, it is possible to force some probability entries close to zero, making negligible the contribution to the final output of the corresponding structural elements. As a result, their subsequent pruning will not significantly alter the network performance. Our pruning approach is somehow similar to what presented in [5], [16], where the employed masks work as scaling factors without a probabilistic foundation.

To demonstrate the feasibility of our technique we prune a LeNet-5 based network [17] and AlexNet [18]. This work is a first step towards the pruning of higher-dimensional networks also comprising more complex layers, such as Transformers.

The paper is organized as follows. The next Section explains how the masking pruning can be applied, how it is possible to generate the binary masks and describes the training strategy along with a novel regularization term. Before Conclusion, Section III reports the pruning results achieved for three networks-based classifiers of the Fashion-MNIST and CIFAR-100 datasets.

II. Pruning by masking

In a standard deep feed-forward fully connected neural network (NN) the input vector $\boldsymbol{x}$ is processed by a sequence of L hidden layers that finally produce an output vector $\boldsymbol{y}$.

[1] Source code publicly available at https://github.com/SSIGPRO/MaskPruning

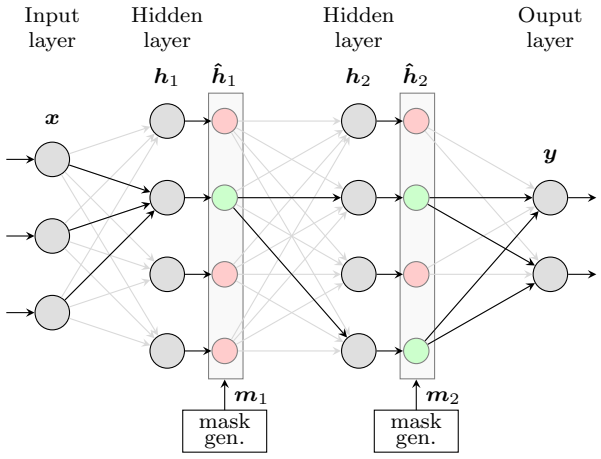

Fig. 1. Block scheme of the proposed pruning method with additional probability mask and sub-nets for their generation.

The i-th hidden layer contains n_i neurons and produces the vector $\boldsymbol{h}_i \in \mathbb{R}^{n_i}$ by processing the n_{i-1} outputs of the previous layer:

$$\boldsymbol{h}_i = f_i(\boldsymbol{W}_i \boldsymbol{h}_{i-1} + \boldsymbol{b}_i) \qquad i = 1, \dots, L \tag{1}$$

where $\boldsymbol{W}_i \in \mathbb{R}^{n_i \times n_{i-1}}$ is the weight matrix, $\boldsymbol{b}_i \in \mathbb{R}^{n_i}$ is the bias vector, $f_i(\cdot)$ is the activation function. With this notation, $\boldsymbol{h}_0$ corresponds to the input $\boldsymbol{x}$, while the output $\boldsymbol{y}$ is $\boldsymbol{h}_L$.

A. Structured Pruning

Structured pruning is a technique that assumes $\boldsymbol{h}_i$ as sparse, i.e., only a fixed subset of its entries is non-negligible and ideally one could completely ignore all the almost-zero entries without any relevant performance loss. This procedure therefore leads to the removal of the neurons in the i-th layer that do not provide a valuable contribution.

Structural sparsity does not naturally arise in DNNs. To promote it, we propose a method that couples binary mask layers $\boldsymbol{m}_i \in \{0,1\}^{n_i}$ with each hidden layer. The purpose of every $\boldsymbol{m}_i$ is to select what neurons to keep and what to prune.

In particular, as shown in Fig. 1, every $\boldsymbol{m}_i$ is multiplied element-wisely with $\boldsymbol{h}_i$, such that $\hat{\boldsymbol{h}}_i = \boldsymbol{m}_i \odot \boldsymbol{h}_i$ is the pruned output, and

$$\boldsymbol{h}_i = f_i(\boldsymbol{W}_i \hat{\boldsymbol{h}}_{i-1} + \boldsymbol{b}_i) \tag{2}$$

now substitutes (1). Here, $\boldsymbol{m}_i$ promotes sparsification by zeroing the elements of $\boldsymbol{h}_i$ corresponding to a 0 in the mask. As a result, for each pruned neuron in the i-th hidden layer it is possible to drop a row in $\boldsymbol{W}_i$, an element in $\boldsymbol{b}_i$ and a column in $\boldsymbol{W}_{i+1}$, i.e., the total amount of network parameters decreases by $n_i + n_{i+1} + 1$. To make this method effective, $\boldsymbol{m}_i$ must be properly designed to avoid performance degradation.

B. Binary Mask Generation

The need of having a binary mask is incompatible with the need of training it. This is because the gradient of a function whose output only consists of $\{0,1\}$ is either 0 or undefined, and as a result backpropagation stops working. Inspired by [14], we propose a three stages approach to build a trainable

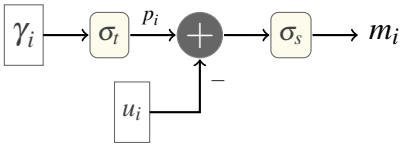

Fig. 2. Three staged sub-network generating binary masks

quasi-binary mask, that is turned totally binary only at the end of the training phase to guarantee structured pruning.

Following the scheme in Fig. 2, the mask $\boldsymbol{m}_i$ is obtained starting from a *probability* mask $\boldsymbol{p}_i \in [0,1]^{n_i}$ whose entries are the probabilities of every entry of $\boldsymbol{h}_i$ to be kept. In particular,

$$\boldsymbol{p}_i = \sigma_t(\boldsymbol{\gamma}_i) \tag{3}$$

where $\boldsymbol{\gamma}_i \in \mathbb{R}^{n_i}$ is a set of trainable mask parameters associated to the i-th layer, $\sigma_t(\cdot) = (1 + \exp(-t\,\cdot))^{-1}$ is the sigmoid function used to element-wisely map $\boldsymbol{\gamma}_i$ in $[0,1]$, and t is a scale parameter called slope.

The second stage consists in creating a vector $\boldsymbol{u}_i \in [0,1]^{n_i}$ containing instances of random independent and uniformly distributed variables, and the third in comparing $\boldsymbol{p}_i$ with $\boldsymbol{u}_i$. This last stage enables $\boldsymbol{m}_i$ to work as a quasi-binary mask whose quasi-ones show up with probabilities dictated by $\boldsymbol{p}_i$. More in details:

$$\boldsymbol{m}_i = \sigma_s(\boldsymbol{p}_i - \boldsymbol{u}_i) = \sigma_s(\sigma_t(\boldsymbol{\gamma}_i) - \boldsymbol{u}_i) \tag{4}$$

where s is the slope controlling the final degree of similarity of $\boldsymbol{m}_i$ with a binary mask, in particular the higher s, the more binary $\boldsymbol{m}_i$. Note that $\boldsymbol{\gamma}_i$ is the only vector containing trainable parameters.

C. Training strategy

To gain control upon the overall amount of sparsity introduced by the masks $\boldsymbol{m}_i$, we introduce a new loss regularization term called r:

$$r(\boldsymbol{\Gamma}, \alpha) = -\frac{1}{\sum_{i=1}^{L-1} n_i} \sum_{i=1}^{L-1} \|\boldsymbol{p}_i - \alpha\|_2^2 \tag{5}$$

where $\boldsymbol{\Gamma} = \{\boldsymbol{\gamma}_1, \dots, \boldsymbol{\gamma}_{L-1}\}$ is the set of all the trainable parameters that learns the probability masks, as in (3); and $\alpha \in]0.5, 1[$ is a hyper-parameter that controls the sparsity. By picking an α close to 1, the network will react by pushing as many entries of $\boldsymbol{p}_i$ as possible to 0, in order to reduce the regularization term, but not too many to avoid a global loss degradation. Note that $\alpha \leq 0.5$ would simply transform $\boldsymbol{m}_i$ into vectors of ones, i.e., sparsity would not be promoted.

The regularization term is added to the original loss function $\mathcal{L}$ (designed to cope with the main network task), such that the new loss becomes:

$$\mathcal{L}^\star(\boldsymbol{\theta}, \boldsymbol{\Gamma}, \alpha) = (1 - \phi)\mathcal{L}(\boldsymbol{\theta}) + \phi r(\boldsymbol{\Gamma}, \alpha) \tag{6}$$

where $\phi \in [0,1]$ is the regularization weight and $\boldsymbol{\theta}$ represents the set of all the trainable network parameters. The amount of pruned neurons is entirely controlled by the two hyper-parameters ϕ and α, which the designer can tune to obtain the desired pruning intensity.

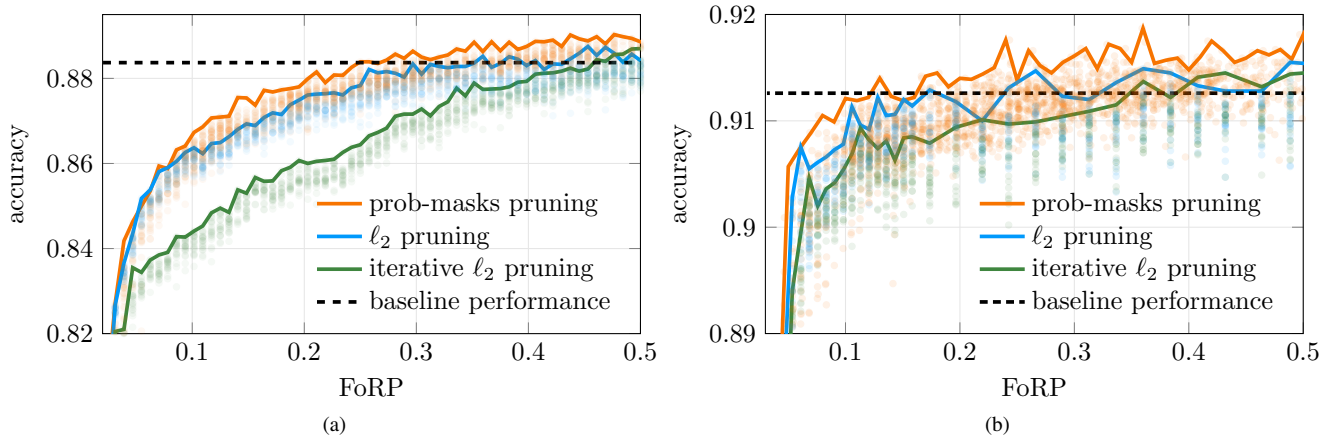

Fig. 3. Structured pruning methods performances in terms of accuracy as a function of the Fraction of Remainig Parameters (FoRP) in case of shallow network (a) and in case of LeNet-5 network (b). Dashed lines report the accuracy of the baseline networks.

We identify two different training strategies. The first consists in training both masks and standard layers from scratch (using (6)), the second divides the training in two steps. The initial architecture is trained with (5), then the dense layers are coupled with a randomly initialized mask and the model is retrained having (6) as loss function. We refer to the first strategy as "one-step-training", while to second as "two-step-training". The trained networks are automatically pruned according to the vectors $\boldsymbol{p}_i$, i.e., only the neurons for which the entries of each associated $\boldsymbol{p}_i$ exceed a threshold T are kept. If not specified, we always consider $T = 0.5$. This pruned network is then used at inference time.

III. Numerical Evidences

We compare our proposal against a family of magnitude based pruning methods from [9], [12], considered to be a standard reference because of their simplicity and effectiveness [19]. When employed for structured pruning, these methods may consider as "negligibly important" the neurons that, with respect to the other neurons within the same layer, have the lowest $\ell_{1,2}$-norm of the weights, and prune them. This has a similar effect to training the network with a group-sparse $\ell_{1,2}$ regularization and then pruning it [5], [7]. We report here performances only for the ℓ_2-based pruning approaches since, in all considered examples, it outperforms ℓ_1-based methods.

The training phase starts from a *baseline* network, specifically designed and trained to solve the given task. Two different training strategies have been implemented by considering both non-iterative and iterative approaches [16]. For the latter, the training is done by recursive steps: the model is first pruned by removing a fixed percentage of neurons according to the ℓ_2-norm of the weights and then the model with the remaining neurons is fine-tuned by retraining. The iteration of this procedure gives us models with an increasing amount of pruned neurons. Conversely, the non-iterative approach directly applies the desired amount of pruning to the baseline model before the network retraining. Note that, to obtain statistically consistent results, twenty independent training procedures are carried out. In case of iterative ℓ_2-based structured pruning, the twenty independent trainings regard each iteration which starts from the model showing the highest accuracy on a validation set.

A. Network setting

To test the pruning capability we look here at a classification task, assessing the final performance in terms of accuracy. In particular, we select the Fashion-MNIST dataset [20], consisting of $51\,000$ grayscale 28×28 images for training, $9\,000$ for validation and $10\,000$ for testing, all equally divided in 10 different classes. We also adopt the CIFAR-100 dataset, whose $60\,000$ native images are expanded to $224 \times 224 \times 3$. CIFAR-100 is equally divided in 100 different classes. We adopt the Fraction of Remaining Parameters (FoRP), e.g., the number of remaining parameters after pruning over the total number of network parameters, to compare different pruned models in terms of pruning strength.

We test the pruning methods on a shallow, on a midly deep and on a truly deep NN. The shallow network comprises one hidden layer with $n_1 = 128$ and an output layer with $n_2 = 10$. The mildly deep one is designed according to the LeNet-5 structure [17] and consists of two convolutional layers coupled with average pooling layers, followed by a flatten layer and a cascade of three fully connected layers with $n_3 = 120, n_4 = 84, n_5 = 10$ respectively. The deep network is designed similarly to AlexNet [18], it consists of 5 convolutional layers and 3 dense layers, having $n_6 = 4\,096$, $n_7 = 4\,096$ and $n_8 = 100$ neurons. The total amount of parameters for this three baseline networks are $101\,770$, $107\,786$ and $58\,696\,548$ respectively. For the second and third, fully-connected layers contain 97.6% and 92.9% of the parameters. AlexNet is trained with CIFAR-100, while the other two use Fashion-MNIST.

The networks adopt ReLu as the activation function of each fully connected (and convolutional if present) hidden layer. Output layers are followed by a softmax activation function, used to express the probability of the input to belong to each of the possible classes. The training employs categorical cross-entropy as main loss function, a batch size equal to 64, and

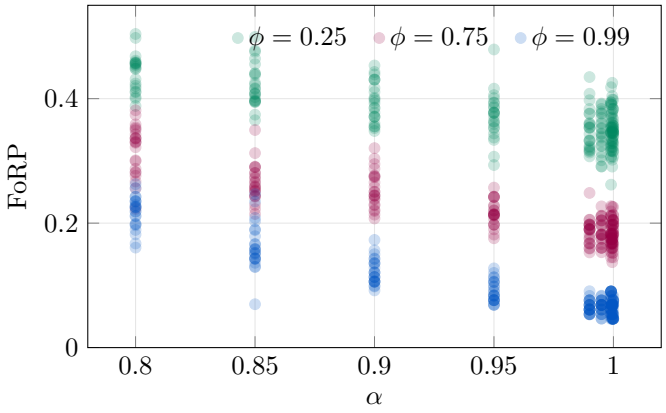

Fig. 4. Sparsity as a function of the threshold α and the regularization weight ϕ for a pruned LeNet-5 network.

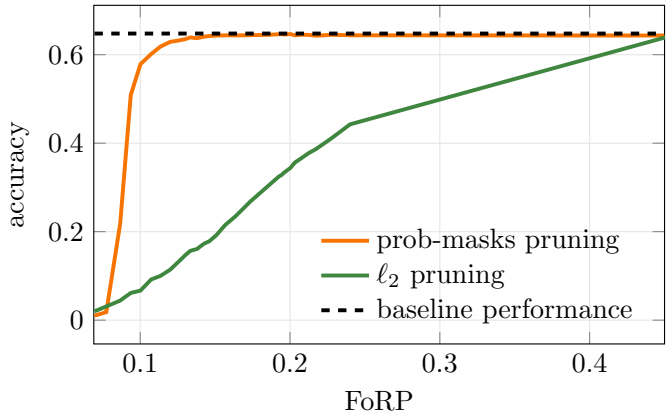

Fig. 5. Accuracy of the AlexNet-based one-step-trained model for different values of $T \in (0, 1)$, compared with the same model, first trained, then pruned with ℓ_2-norm (without retraining). The dashed line represents the accuracy obtained before ℓ_2 pruning was applied.

the Adam optimizer. Learning rate has been fixed to 0.001 and multiplied by 0.2 any 15 epochs passed without the loss decreasing significantly. Only exception is the learning rate used to train the deep NN and masks together, that has been set to 0.0001. In (4), we set $t = 5$ and $s = 200$.

For the LeNet-5 and AlexNet networks, we focus our pruning method on fully-connected layers since they contribute to the majority of parameters as in some well-known architectures [21], [22]. Yet, given its nature, our approach can be readily extended to work with other types of layers, such as convolutional and recurrent ones, and it can be also adopted for unstructured kind of pruning. These and further applications will surely be the subject of future communications.

B. Performance comparison

Fig. 3a and Fig. 3b show the accuracy against the FoRP in case of shallow and mildly deep NN respectively. Both networks with the proposed binary mask layers are trained by means of two-step-training. For each pruned model, accuracy is represented with low opacity dots, while with a solid line we represent the maximum accuracy envelop of a each pruning technique. The graphs show how our proposed model returns better performing models almost all the time. For a fixed accuracy, our pruning technique returns a shallower model than the competitors pruned models. To reach the baseline accuracy $0.884/0.913$ for the shallow/mild deep NN, the minimum FoRP achieved by the proposed method is $0.25/0.13$, while ℓ_2-norm method meets the baseline curve at $0.36/0.17$ FoRP, and its iterative version does at $0.46/0.36$ FoRP.

To show how the FoRP of a model pruned with the proposed model is controlled by the two hyper-parameters α and ϕ, we also provide Fig. 4. The FoRP of each involved pruned LeNet-5 is represented as a function of the threshold α for three different values of the regularization weight ϕ. The graph confirms the role of each parameter. Indeed, increasing either α or ϕ leads to binary masks with a lower number of ones, i.e., a lower FoRP.

Training and pruning AlexNet using the two-step-training strategy did not lead to any interesting comparisons, in fact it turned out that for such task even random pruning performs as well as all the considered methods. Nevertheless, at the end of a one-step-training, our method returns a family of models each with different and competitive FoRP/accuracy ratios. In Fig. 5, we display many pruned realizations of a single one-step-trained AlexNet-based model ($\boldsymbol{\alpha} = 0.9$, $\phi = 0.9$). Every realization modifies T to obtain pruned architectures having different FoRP. Our models always outperform the models normally trained and pruned with ℓ_2-norm (without retraining). These results demonstrate our method can be effectively used to train and prune the model from scratch, without the need of a first round of training, or without the need of a second round of re-training, as opposite to the other approaches, allowing a faster design approach.

IV. Conclusion

We present a structured pruning method where a novel trainable probability mask layer is designed to select the network elements with the highest probability to be pruned, and we apply it to fully connected layers. The probability masks are learned during training by including an ad-hoc designed regularization term in the loss function, that pushes the probabilities of each neuron to be pruned towards a binary value. After training, all the neurons having an associated probability close to 0 are pruned.

The proposed method is assessed on a shallow, a LeNet-5-based and an AlexNet-based neural networks, designed to solve a classification task. We adopt Fashion-MNIST and CIFAR-100 datasets to compare our method with two magnitude-based pruning techniques. For the first two models, our method returns a pruned model that reaches the performance of the baseline model with only the 25% /13% of the initial parameters. Conversely, the magnitude-based approaches require the 36% /17% of parameters to meet the baseline accuracy. With AlexNet, our method does not experience accuracy drops up to 85% of total pruned parameters, while the magnitude-based method, without retraining, starts degrading the accuracy already at 55%.

Future communications will further exploit our method by applying it to a wider range of layers and to more complex network architectures.

REFERENCES

[1] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June. PMLR, 5 2019, pp. 362–372.

[2] Z. Li, E. Wallace, S. Shen, K. Lin, K. Keutzer, D. Klein, and J. E. Gonzalez, "Train Large, then compress: Rethinking model size for efficient training and inference of transformers," in *37th International Conference on Machine Learning, ICML 2020*, vol. PartF16814. PMLR, 11 2020, pp. 5914–5924.

[3] T. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[4] T. Elsken, J. H. Metzen, and F. Hutter, "Neural Architecture Search: A Survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

[5] R. K. Ramakrishnan, E. Sari, and V. P. Nia, "Differentiable Mask for Pruning Convolutional and Recurrent Networks," *Proceedings - 2020 17th Conference on Computer and Robot Vision, CRV 2020*, pp. 222–229, 5 2020.

[6] E. Frantar and D. Alistarh, "Sparsegpt: Massive language models can be accurately pruned in one-shot," 2023.

[7] H. Li, H. Samet, A. Kadav, I. Durdanovic, and H. P. Graf, "Pruning filters for efficient convnets," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 11 2017.

[8] X. Dong, S. Chen, and S. J. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem, 2017, pp. 4858–4868.

[9] M. Hagiwara, "Removal of hidden units and weights for back propagation networks," *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, pp. 351–353, 1993.

[10] P. L. Narasimha, W. H. Delashmit, M. T. Manry, J. Li, and F. Maldonado, "An integrated growing-pruning method for feedforward network training," *Neurocomputing*, vol. 71, no. 13-15, pp. 2831–2847, 8 2008.

[11] S. A. Janowsky, "Pruning versus clipping in neural networks," *Physical review. A, General physics*, vol. 39, no. 12, pp. 6600–6603, 1989.

[12] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 10 2015.

[13] A. Polyak and L. Wolf, "Channel-level acceleration of deep face representations," *IEEE Access*, vol. 3, pp. 2163–2175, 10 2015.

[14] C. D. Bahadir, A. V. Dalca, and M. R. Sabuncu, "Learning-Based Optimization of the Under-Sampling Pattern in MRI," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, pp. 780–792.

[15] F. Martinini, M. Mangia, F. Pareschi, R. Rovatti, and G. Setti, "Compressed Sensing Inspired Neural Decoder for Undersampled MRI with Self-Assessment," *2021 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 01–06, 10 2021.

[16] Z. Zhou, W. Zhou, R. Hong, and H. Li, "Online Filter Weakening and Pruning for Efficient Convnets," *Proceedings - IEEE International Conference on Multimedia and Expo*, vol. 2018-July, 10 2018.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, may 2017.

[19] T. Gale, E. Elsen, and S. Hooker, "The State of Sparsity in Deep Neural Networks," 2 2019.

[20] H. Xiao, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms." *CoRR*, vol. abs/1708.0, 2017.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.