MOX-Report No. 23/2026

Elimination-compensation pruning for fully-connected neural networks

Ballini, E.; Muscarnera, L.; Fumagalli, A.; Scotti, A.; Regazzoni, F.

MOX, Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

mox-dmat@polimi.it https://mox.polimi.it

Elimination-compensation pruning for fully-connected neural networks

Enrico Ballini[1*] Luca Muscarnera[1*] Alessio Fumagalli[1] Anna Scotti[1]

Francesco Regazzoni[1†]

[1] *MOX, Department of Mathematics, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy*

February 25, 2026

Abstract

The unmatched ability of deep neural networks to capture complex patterns in large and noisy datasets is often associated with their large hypothesis space and the vast number of parameters characterizing modern architectures. Pruning techniques have emerged as effective tools to extract sparse representations of neural network parameters while preserving accuracy. However, a fundamental assumption behind pruning is that expendable weights have a small impact on the network error, whereas highly important weights exert a larger influence on inference. We argue that this idea could be generalized; what if a weight is not simply removed but also compensated with a perturbation of the adjacent bias, which does not contribute to the network sparsity? Our work introduces a novel pruning method in which the importance measure of each weight is computed considering the output behavior after an optimal perturbation of its adjacent bias. These perturbations can be then applied directly after the removal of each weight, independently of each other. After deriving analytical expressions for the aforementioned quantities, numerical experiments are conducted to benchmark this technique against some of the most popular pruning strategies, demonstrating an intrinsic efficiency of the proposed approach in very diverse machine learning scenarios.

1 Introduction

Nowadays, neural networks are capable of succeeding in very complex tasks spanning different applications such as, to mention a few, computer vision [32], natural language processing [24], scientific computing and numerical solution of partial differential equations [7, 27, 19]. Neural networks rely on a potentially very large number of parameters that must be optimized for the specific task [27, 14, 4]. Such a high parameter count can give rise to challenges related to training efficiency, storing memory, and inference time, especially when considering the possible embedding of neural networks in portable devices with limited hardware resources. Despite the excellent ability of neural networks to solve complex problems, their application can be limited by the computational burden arising from the sequence of matrix-vector multiplications required to perform inference, thus restricting their use in several domains where computational resources are constrained by structural or energy limitations. Moreover, it is interesting to investigate whether the large number of parameters is truly necessary to achieve good performance.

It is therefore relevant to study methodologies for handling the large amount of parameters issue. There can be different strategies: to mention a few, it is possible to compress the weight matrices with a proper factorization; reduce numerical precision, for instance by using 8-bit representations instead of 64-bit ones; restrict parameters to a finite set of values via quantization; apply knowledge distillation; study better neural network architectures; or merge neurons with highly correlated activations [2]. We refer, among many review papers, to [1, 25, 31, 6] for a deep review of the aforementioned methods, and to [11] for a comparison of well-established techniques.

*These authors contributed equally to this work.

†Corresponding author: `francesco.regazzoni@polimi.it`

In this work, we focus on pruning methods, which are a family of techniques in which some non-important synapses are removed (the weights are set to zero), leading to a lower number of non-zero weights. The authors of [26] explain that it is highly probable that there exists a pruned network close enough to the original one, which motivates the study of pruning method.

The core idea is to first define an *importance* measure of the weight then, starting from a possibly already trained large neural network, set to zero as many weights as possible whose importance measure is low, while preserving the accuracy of the network.

Several techniques were developed in this direction and we refer to [28, 5, 6] for an overview.
Pruning is typically executed in a finite loop of train, then prune, then repeat, which usually reduces to train-prune-train, where the last train phase is referred to as *fine train*. It is important to mention that pruning can be an effective strategy to facilitate training and obtain a good architecture for the specific problem *a priori* to training [3]. This enhancement can be achieved by incorporating information about the specific problem at hand into the architecture of the network. In our work, we focus on a pruning strategy for generic tasks. Pruning can be achieved by a *local* technique or a *global* technique. In the former, the weights are set to zero by considering local quantities, such as their magnitude [13], or more complex importance indicators that are computed by considering the quantities of the isolated layer [9, 35, 21].
In the latter, the importance of the weight is defined by considering quantities related to the output of the neural network. A well-known strategy is Optimal Brain Damage [18], Optimal Brain Surgeon [12], and their variants, which compute the second-order derivatives of the loss and remove weights with low influence on it.

Other strategies focus on sensitivity, considering the first-order derivative of the loss and, possibly, derived quantities such as variance and mean values. One of the first works in this direction is [23], while more recent works include [33, 22, 29]. It is also possible to effectively avoid the use of derivatives, as shown in [34], obtaining a global measurement of the weight importance.
In contrast to global approach, [8] considers statistical quantities of the sensitivity, a line of research that is further developed by [10], who make the computation of importance dependent on local quantities.
We focus on a global approach, which has the advantage of being more suitable for formal analysis.

From the cited literature, it is apparent that the derivative of the loss function w.r.t. the trainable weights and the value of the weights are relevant quantities for obtaining effective pruning. Indeed, [5] proposes the product of these quantities as a possible importance measure.
To the extent of the author's knowledge, little has been done regarding the inclusion of biases in pruning methods, although biases can be relevant in defining importance [20, 21].

Contribution and paper organization. In this paper, we propose a pruning technique for fully connected neural networks applied to different tasks with emphasis on applications in scientific machine learning. Our method is based on the minimization of the expected value of the discrepancy between the output of the original network and the pruned one. Remarkably, we never compute the derivatives of the loss function which can be subject to noise effects nearby a minimum value; instead, using a Taylor expansion of the network, we define a novel importance measure that concurrently includes the effects of both the weights on the network output and the effects of the biases.
The paper is organized as follows. In Section 2, we present a general framework for pruning. In Section 3, we introduce our proposed method. Numerical test cases are presented in Section 4. Conclusions are drawn in Section 5.

2 Basic principles

We present in this section the basic definitions and notation for neural networks, Section 2.1, and pruning in general form, Section 2.2.

2.1 Fully-connected neural networks

The goal of this section is setting some notation regarding fully-connected neural networks. Considering a layer $\ell \in 1, \dots, L$, whose size is denoted by n_ℓ, we have the following relation between the input, $z^{\ell-1} \in \mathbb{R}^{n_\ell - 1}$, and the

output, $z^\ell \in \mathbb{R}^{n_\ell}$, of the layer ℓ:

$$z^\ell = \sigma^\ell(W^\ell z^{(\ell-1)} + b^\ell),$$

where the coefficients $W^\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and $b^\ell \in \mathbb{R}^{n_\ell}$ are the trainable weights of the unknown affine transformation of layer ℓ, and σ^ℓ is the non-linear activation function, except for σ^L which is set to be the identity function. We call the term $W^\ell_{ij} z^{\ell-1}_j$ the *signal* of the layer ℓ. With abuse of notation, calling $z^\ell(z^{(\ell-1)}) \coloneqq \sigma^\ell(W^\ell z^{(\ell-1)} + b^\ell)$, the full neural network can be written as

$$y = z^{(L)} \circ z^{(L-1)} \circ \ldots \circ z^{(1)}(x),$$

with $x \in \mathbb{R}^{n_{\text{in}}}$ the input and $y \in \mathbb{R}^{n_{\text{out}}}$ the output. To simplify the notation, we define $\theta = \cup_\ell (W^\ell, b^\ell)$ to be all the trainable weights of a neural network and we denote by $|\theta|$ the total number of trainable weights. When it is relevant to highlight the dependence of the network on its trainable weights, we use the following notation

$$y = y(x; \theta).$$

The value of the parameters, θ, are optimized to minimize a loss function, $\mathcal{L}$.

2.2 Why is pruning complicated? A formal approach

For clarity of exposition, we introduce the *mask matrix*, denoted by M^ℓ_{ij}, associated with layer ℓ and taking values in $\mathcal{M}^\ell = \{M \in \mathbb{R}^{n_\ell \times n_{\ell-1}} : M_{ij} \in \{0,1\}\}$. Its entries indicate which weights are kept during a training procedure and which are set to zero, pruning the corresponding connections. The pruned layer is given by

$$z^\ell = \sigma^\ell(M^\ell \odot W^\ell z^{(\ell-1)} + b^\ell).$$

We denote by $\overline{\theta}_M$ the set of the weights with the masks applied. Therefore, the pruned neural network will be denoted by $y(x, \overline{\theta}_M)$. We also write $|M^\ell|$ to denote the number of nonzero entries of M^ℓ.

The pruning problem can be formulated within the following general framework[1]:

$$\begin{aligned} &\min_{\substack{M^\ell_{ij} \in \{0,1\} \\ \forall i,j,\ell}} \rho\left(y(x;\theta),\, y(x;\overline{\theta}_{M^\ell_{ij}})\right), \\ &\text{subject to:} \\ &\sum_\ell |M^\ell| = N, \end{aligned} \tag{1}$$

where ρ denotes an appropriate measure of the discrepancy between the original network and its pruned counterpart.

Given the extremely large number of parameters involved, the optimization problem (1) is computationally infeasible. It is therefore commonly replaced by the following surrogate procedure. To determine which weights should be removed, we assign to each weight W^ℓ_{ij} an importance value $\mathcal{I}_{W^\ell_{ij}}$. Weights whose importance falls below a prescribed threshold are then set to zero:

$$\begin{aligned} &\mathcal{I}_{W^\ell_{ij}} = d\left(y(x;\theta), y\left(x;\, \theta|_{W^\ell_{ij} \leftarrow 0}\right)\right), \\ &M^\ell_{ij} = 0, \quad \text{for } i,j,\ell \ \text{ s.t. } \ \mathcal{I}_{W^\ell_{ij}} \leq T_h. \end{aligned} \tag{2}$$

Here T_h is a threshold that may be chosen a priori or determined a posteriori via order statistics to obtain a prescribed number of retained weights. As in (1), the function d serves as a suitable measure of discrepancy between the original and pruned networks. Examples of choices of importance scores are $\mathcal{I}_{W^\ell_{ij}} = \frac{1}{2}\frac{\partial^2 \mathcal{L}}{\partial {W^\ell_{ij}}^2}(W^\ell_{ij})^2$ [18], $\mathcal{I}_{W^\ell_{ij}} = |W^\ell_{ij}|$ [13, 5], or possibly $\mathcal{I}_{W^\ell_{ij}} = \left|\mathcal{L}(y(x;\theta)) - \mathcal{L}(y(x;\, \theta|_{W^\ell_{ij} \leftarrow 0}))\right|$.

Remark 2.1 (Pruning for inefficient training)**.** Let $\mathcal{Y}$ be the set of neural networks associated with a given architecture. The training procedure aims to identify the optimal (in the sense specified by the loss function) neural

[1] We do not use Einstein summation convention.

network in $\mathcal{Y}$. We remark that the pruned neural network is included in $\mathcal{Y}$. Consequently, pruning does not confer any clear theoretical advantage. Its value lies instead in practical considerations: training can be a challenging task, and the large number of parameters to be optimized may prevent the procedure from finding a suitable candidate for minimizing the loss function. Pruning seeks to simplify the optimization problem while preserving the essential characteristics of the network.

3 Elimination-compensation approach

In this section, we present our proposed method. The central idea is to compensate the removal of a weight by modifying the corresponding bias, and to incorporate this adjustment into the definition of the importance measure. In essence, the importance quantifies how strongly a given weight influences the network output and how well it can be replaced by an appropriate bias correction.

Therefore, rather than simply setting a weight to zero, we compensate for its removal by adjusting the bias b_i^ℓ. Denoting the adjustment by Δb^ℓ, the ℓ-th layer of the pruned network becomes

$$z^\ell = \sigma^\ell(M^\ell \odot W^\ell z^{(\ell-1)} + b^\ell + \Delta b^\ell).$$

We highlight that the modification of the bias does not affect the inference cost of the network, as $b^\ell + \Delta b^\ell$ will be replaced by the result of the summation.

For clarity of exposition, we illustrate the main steps for the scalar output case, $n_{\text{out}} = 1$, and provide the general formulation for $n_{\text{out}} \geq 1$ thereafter.

Before giving a formal definition of the importance measure, we introduce the discrepancy Δy between the output of the pruned network and that of the original one:

$$\Delta y(x; \theta, i, j, \ell, \Delta b_{ij}^\ell) = y(x; \theta) - y\left(x; \theta|_{W_{ij}^\ell \leftarrow 0, b_i^\ell \leftarrow b_i^\ell + \sum_j \Delta b_{ij}^\ell}\right),$$

where Δb_{ij}^ℓ[2] is a bias compensation term that is chosen appropriately.

Direct non-linear method. We define the importance of a weight as the expected output variation over the train dataset obtained when the weight is set to zero and compensated by an appropriate value Δb_{ij}^ℓ:

$$\mathcal{I}_{W_{ij}^\ell} = \mathbb{E}_x\left[\, \Delta y(x; \theta, i, j, \ell, \Delta b_{ij}^\ell)^2\right].$$

It remains to determine an optimal choice for Δb_{ij}^ℓ.

A first idea is to choose Δb_{ij}^ℓ so as to preserve the mean value of the signal (Section 2.1):

$$\overline{\Delta b_{ij}^\ell} \coloneqq \mathbb{E}_x\left[W_{ij}^\ell z_j^{\ell-1}\right] = W_{ij}^\ell\, \mathbb{E}_x\left[z_j^{\ell-1}\right].$$

The corresponding importance then becomes

$$\mathcal{I}_{W_{ij}^\ell} = \mathbb{E}_x\Big[\, \Delta y(x; \theta, i, j, \ell, \overline{\Delta b_{ij}^\ell})^2\Big]. \tag{3}$$

This simple and direct approach encounters a practical limitation: the importance must be computed on a weight-by-weight basis, meaning that at each step a single weight is set to zero while all others remain unchanged. This would require a number of forward evaluations equal to the number of weights, rendering the procedure computationally prohibitive. Indeed, denoting by C_f the cost of the forward evaluation, measured as the number of floating-point operations, the computation of the importance of all the weights involves a cost that depends on the total number of weights proportional to $|W|C_f$. Due to the high computational cost, we must adopt an alternative strategy.

[2]Note the presence of double subscript, ij, in the compensation Δb_{ij}^ℓ as it is associated to each weight W_{ij}^ℓ

Proposed method. Assuming that the weights and bias adjustments are small, we proceed by computing a Taylor expansion of the network to approximate the discrepancy:

$$\begin{aligned}\Delta y(x;\theta,i,j,\ell,\Delta b_{ij}^{\ell}) \approx \delta y(x;\theta,i,j,\ell,\Delta b_{ij}^{\ell}) \coloneqq y(x;\theta) - \left(y(x;\theta) + \partial_{W_{ij}^{\ell}} y(x;\theta)(0 - W_{ij}^{\ell}) + \partial_{b_i^{\ell}} y(x;\theta)\Delta b_{ij}^{\ell}\right) = \\ = \partial_{W_{ij}^{\ell}} y\, W_{ij}^{\ell} - \partial_{b_i^{\ell}} y\, \Delta b_{ij}^{\ell}.\end{aligned} \tag{4}$$

We can now define the importance by using the linear approximation (Eq. (4)) and determining the optimal compensation through an associated minimization problem:

$$\mathcal{I}_{W_{ij}^{\ell}} = \min_{\Delta b_{ij}^{\ell} \in \mathbb{R}} \mathbb{E}_x[\delta y(x;\theta,i,j,\ell,\Delta b_{ij}^{\ell})^2]. \tag{5}$$

This optimization problem, although seemingly complicated, can be solved explicitly to obtain the optimal compensation $\widetilde{\Delta b_{ij}^{\ell}}$. To identify the minimizer, we compute the following derivative:

$$\frac{\partial}{\partial \Delta b_{ij}^{\ell}} \left(\mathbb{E}_x[\delta y(x;\theta,i,j,\ell,\Delta b_{ij}^{\ell})^2]\right) = \mathbb{E}_x\left[\left(\partial_{W_{ij}^{\ell}} y\, W_{ij}^{\ell} - \partial_{b_i^{\ell}} y\, \Delta b_{ij}^{\ell}\right)(-\partial_{b_i^{\ell}} y)\right], \tag{6}$$

from which, by setting the derivative to zero, we obtain the optimal compensation $\Delta b_{ij}^{\ell} = \widetilde{\Delta b_{ij}^{\ell}}$:

$$\widetilde{\Delta b_{ij}^{\ell}} = \frac{W_{ij}^{\ell}\mathbb{E}_x\left[\partial_{W_{ij}^{\ell}} y\, \partial_{b_i^{\ell}} y\right]}{\mathbb{E}_x\left[\partial_{b_i^{\ell}} y\right]^2}. \tag{7}$$

Replacing the expression of $\widetilde{\Delta b_{ij}^{\ell}}$ into (5) we have:

$$\mathcal{I}_{W_{ij}^{\ell}} = \mathbb{E}_x\left[\left(\partial_{W_{ij}^{\ell}} y\, W_{ij}^{\ell} - \frac{W_{ij}^{\ell}\mathbb{E}_x\left(\partial_{W_{ij}^{\ell}} y\, \partial_{b_i^{\ell}} y\right)}{\mathbb{E}_x\left(\partial_{b_i^{\ell}} y\right)^2}\partial_{b_i^{\ell}} y\right)^2\right].$$

A note must be done on the computational cost: considering αC_f as the cost of backpropagation, the computation of the importance for all weights is proportional to αC_f instead of $|W|C_f$ as previously mentioned, making this strategy computationally affordable.

In the case of vectorial output, using the ℓ_2-norm of the discrepancy to measure the error, we obtain the following expression:

$$\mathcal{I}_{W_{ij}^{\ell}} = \mathbb{E}_x\left[\sum_k \left(\partial_{W_{ij}^{\ell}} y_k\, W_{ij}^{\ell} - \frac{\sum_k W_{ij}^{\ell}\mathbb{E}_x(\partial_{W_{ij}^{\ell}} y_k\, \partial_{b_i^{\ell}} y_k)}{\sum_k \mathbb{E}_x[(\partial_{b_i^{\ell}} y_k)^2]}\partial_{b_i^{\ell}} y_k\right)^2\right].$$

4 Numerical Experiments

In this section, we present numerical experiments to demonstrate the effectiveness of the proposed method. In the first experiment, we use the well-known MNIST dataset [16], while in the second we employ a dataset constructed from solutions of partial differential equations [30].

The datasets are split into training and test sets. The training of the networks and the evaluation of importance are carried out using the training set. Subsequently, performance is assessed on the test set. Performance is evaluated by computing the test loss for different pruning ratios. We define the pruning ratio, r, as the fraction of removed weights, namely

$$r = 1 - \frac{|W|}{|W_{\text{original}}|},$$

where $|W|$ denotes the total number of weights of the considered network[3], summed over all layers, and W_{original} is the total number of weights before pruning. For clarity, we will report only the test losses.
The proposed method is compared against 5 different strategies, listed below:

- "Non-linear": This is the brute-force approach, in which importance is defined in (3). This strategy is computationally expensive as its cost depends linearly on $|W|$, see Section 3, and therefore not practical. Nonetheless, for the sake of completeness, it is included, with the understanding that it is often infeasible for real applications.
- "Magnitude": A simple yet effective method [5]. In this strategy, importance is defined as $\mathcal{I}_{W_{ij}^{\ell}} = |W_{ij}^{\ell}|$.
- "Gradient-Magnitude": Here, importance is defined as $\mathcal{I}_{W_{ij}^{\ell}} = |W_{ij}^{\ell}| \frac{d\mathcal{L}}{dW_{ij}^{\ell}}$ [5].
- "Random": In this strategy, the weights set to zero are randomly selected. Comparing with this strategy is relevant to demonstrate that the proposed method captures meaningful information during pruning. Nevertheless, random pruning can occasionally yield reasonable results [11].
- "Fully-connected": No pruning is applied. Instead, the number of weights is reduced by uniformly decreasing the width of the layers. The pruning strategy is considered effective if the pruned networks outperform an equivalent fully connected network, i.e. one with the same number of non-zero weights.

All experiments are repeated for 5 different weight initializations to ensure the reliability of the results.
Training is performed using the well-known Adam optimizer [15] with default hyperparameters.

4.1 Case 1: Classification

In this case, we use the MNIST dataset [16, 17]. The pruning strategies are applied to two different architectures, summarized in Tab. 1, to ensure the reliability of the results. The PReLU activation function is adopted in each layer.
Since a classification task is considered, the cross-entropy is used as loss function. The networks are initially trained for 15 epochs, at which point pruning is applied. This is followed by an additional 15 epochs of fine-tuning. The procedure can thus be described as train-prune-train.

The main results are shown in Fig. 1, where the loss values are plotted against the pruning ratio for both architectures in Tab. 1. All the pruning methods described in the itemized list above are represented, highlighting the average loss across different initializations. Additionally, a violin plot is used for each pruning ratio to illustrate the variability in the results.

For completeness, we report also the baseline loss, i.e., the value computed immediately before pruning, shown as a horizontal line.

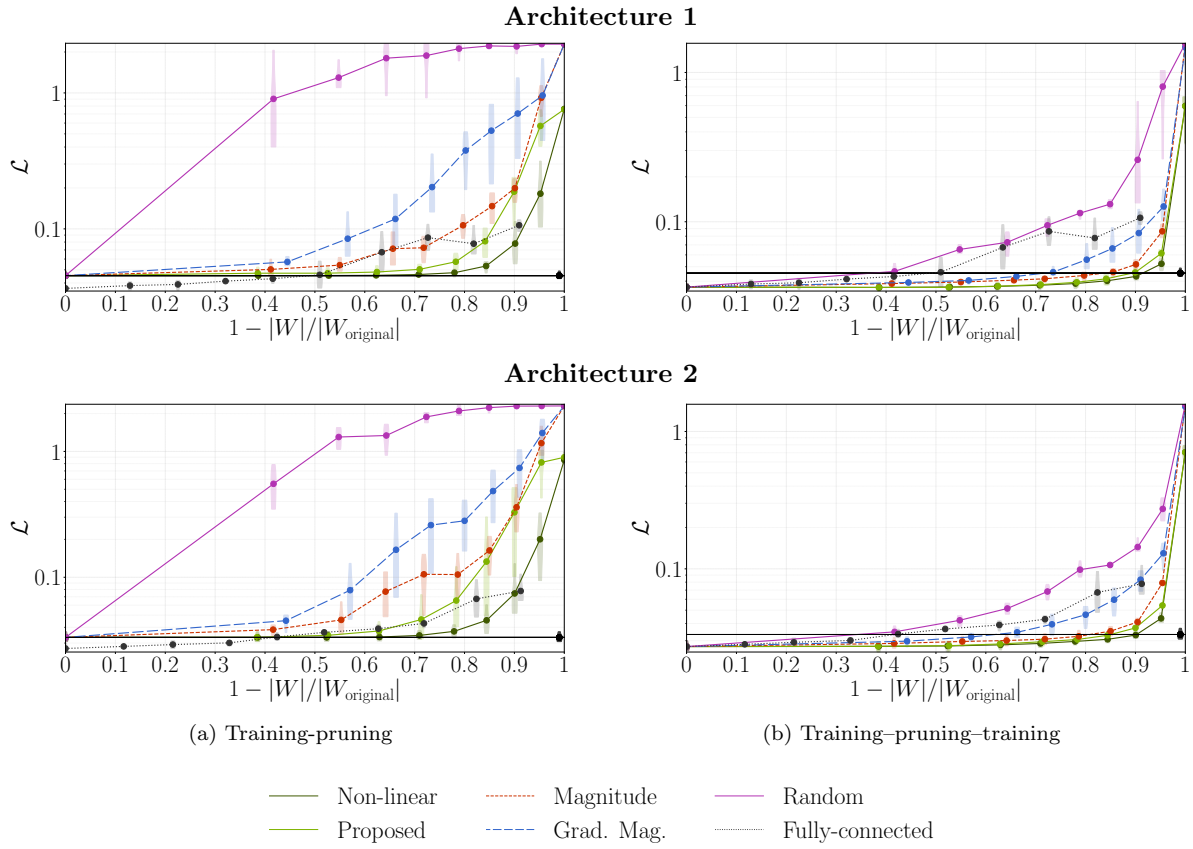
In the left column, the loss values computed immediately after pruning (without any weight adjustment) are shown. Interestingly, using the proposed method, it is possible to remove up to approximately 50% of the weights without a significant change in the loss and without re-training the pruned network.

In the right column, the loss values computed after fine-tuning are shown. We observe that the proposed method consistently achieves lower loss than all other methods, with the loss remaining nearly unchanged even with 80–90% of the weights removed.

	layer's size	$\lvert W\rvert$
Architecture 1	784, 32, 32, 10	26 432
Architecture 2	784, 64, 64, 10	54 912

Table 1: Baseline neural network architectures for MNIST. $|W|$ is the total number of trainable weights.

[3]We do not explicitly refer to pruned weights, as the above definition will also apply to fully connected networks.



(a) Training-pruning

(b) Training–pruning–training

Figure 1: Test losses on MNIST. In the left column, the losses are computed immediately after applying the pruning methods. In the right column, the losses are computed after fine-tuning, so that the overall procedure can be summarized as training–pruning–training.

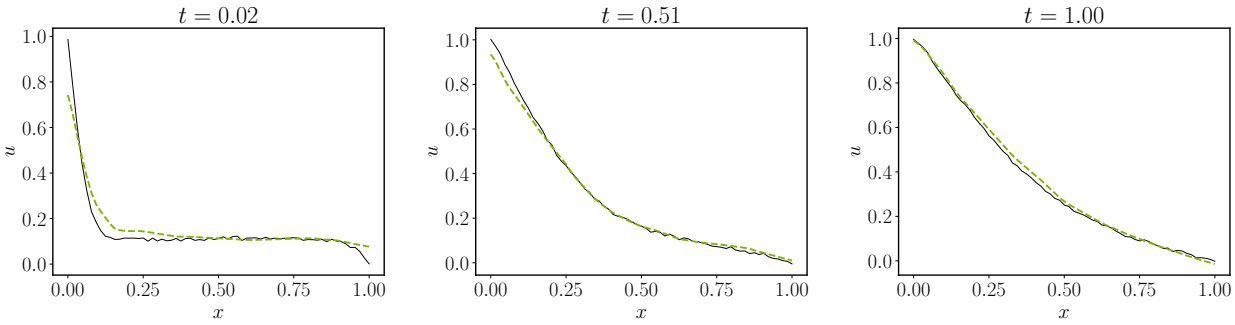


Figure 2: Time snapshots of $u(x,t)$. The noisy data with $d \sim \mathcal{U}(-0.005, 0.005)$ are shown in black, while the green dashed line represents $u(x,t)$ computed by the pruned neural network with architecture 2 (see Tab. 2). In all panels, the network is pruned using the proposed method with a pruning ratio of 0.7.

4.2 Case 2: Partial differential equation with noisy data

In this case, the dataset is derived from the benchmark described in [30]. The data represent the solution u, computed on a uniform grid, of a 1D partial differential equation. The equations describe a time-dependent diffusion-sorption nonlinear system. The equations are parametric, with the parameters denoted by μ. See [30] for further details.

The input to the neural networks consists of the vector (μ, t, x, u_0), where t is time, x is the spatial coordinate, and u_0 denotes the initial conditions on the grid. The output of the networks is $u(t,x)$ for the given μ and u_0.

The dataset used in this paper is smaller than the original one presented in [30] due to the large amount of tests required to compare the methods. Specifically, we restrict the discretization to 10 of the 100 time steps, 64 of the 1024 grid points, and the first 100 samples in the parameter space of size 10 000.

To the original dataset, a uniformly distributed noise, $d(t,x)$, is added to make the test more challenging and to assess the robustness of the pruning method. Considering that the solution u spans a range of approximately $[0,1]$, we test the methods with three levels of noise: $d = 0$, $d \sim \mathcal{U}(-0.005, 0.005)$, and $d \sim \mathcal{U}(-0.01, 0.01)$. See also Fig. 2 for a graphical visualization. A practical interpretation of the noise can, in principle, be related to a measurement noise or numerical inaccuracies.

The pruning method is applied to two neural network architectures: one with 3 hidden layers, and the other with 6 hidden layers, as summarized in Tab. 2.

	layer's size	$\lvert W\rvert$
Architecture 1	68, 32, 32, 32, 1	4 256
Architecture 2	68, 64, 32, 32, 16, 16, 1	8 208

Table 2: Baseline neural network architectures for Diffusion-Sorption. $|W|$ is the total number of trainable weights.

Figs. 3, 4 show the test loss values versus the pruning ratio for both architectures listed in Tab. 2 and for the three aforementioned levels of noise. We observe that, overall, all pruning strategies are robust with respect to noise. The proposed method consistently outperforms all other methods, except for the non-linear strategy, which is occasionally slightly better. However, it should be noted that the non-linear method is roughly $|W|$ times more computationally expensive than the proposed method. The latter has a computational cost comparable to that of a single backpropagation step and therefore requires a negligible amount of time when compared to the overall training time. Here, we do not provide timing comparisons, as they are heavily influenced by the hardware and code optimizations, which are beyond the scope of this work.

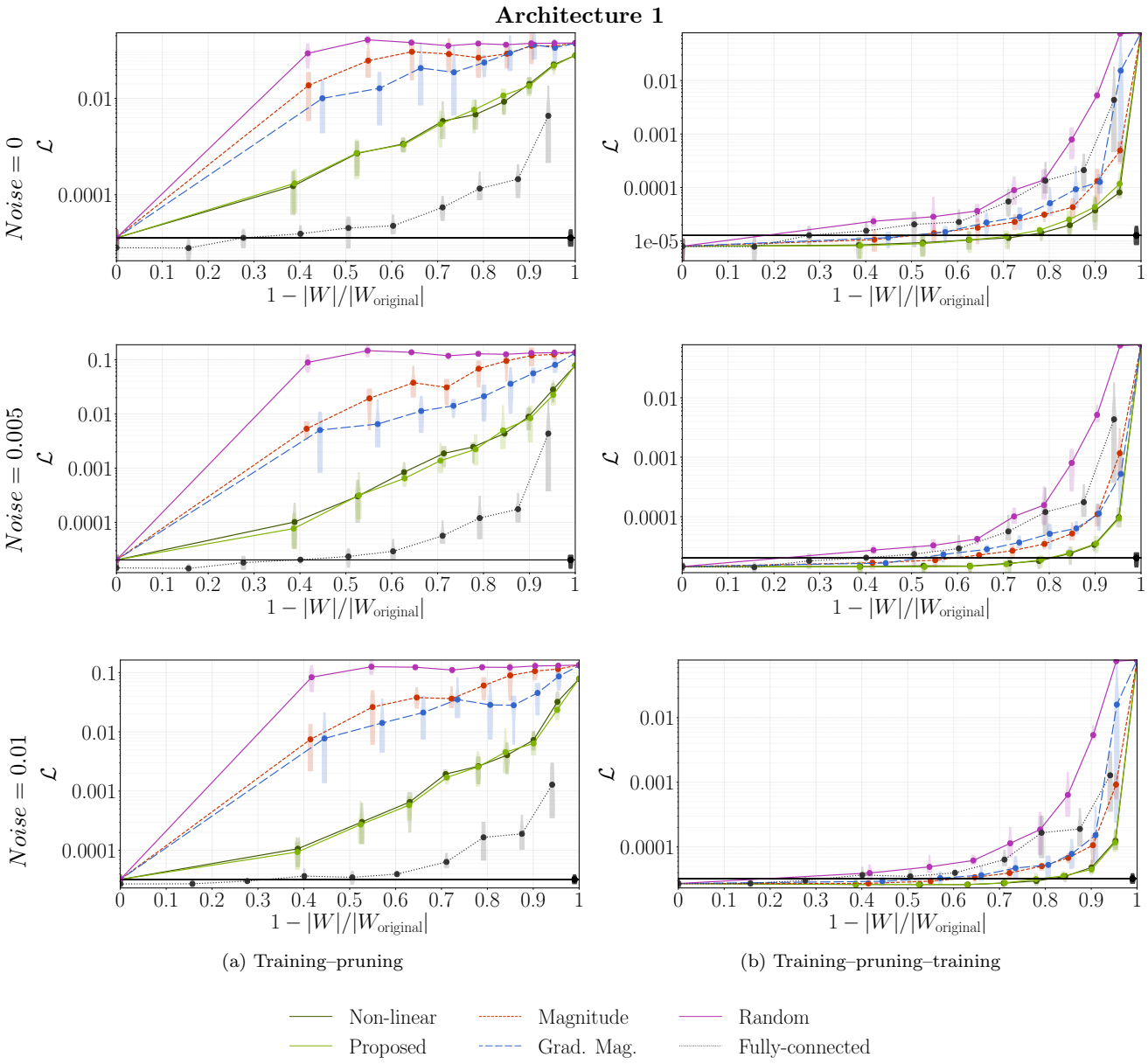


Figure 3: Test losses on Diffusion-Sorption PDE. In the left column, the losses are computed immediately after applying the pruning methods. In the right column, the losses are computed after fine-tuning, so that the overall procedure can be summarized as training–pruning–training.

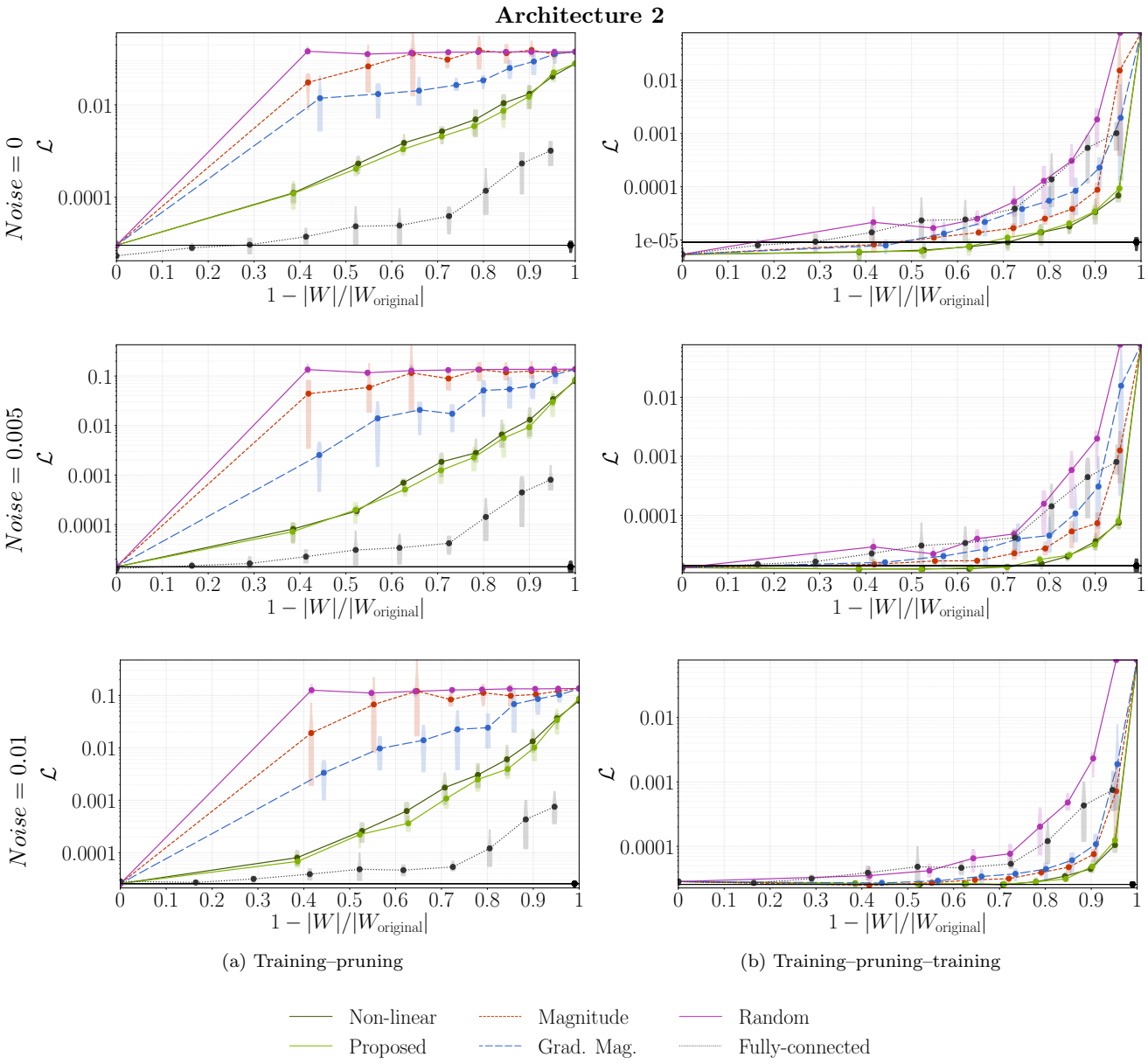


Figure 4: Test losses on Diffusion-Sorption PDE. The bottom row corresponds to a high noisy data. In this case, it is practically difficult to reach low loss values during training, so that the first training phase is sufficient to reach the practical minimum, and the second training phase does not significantly improve performance. Therefore, it is possible for the loss of the baseline to be comparable to that of the fully trained networks, as observed in the bottom-right panel.

5 Conclusions

This work focused on the presentation of a novel pruning strategy, that we implemented considering an ad-hoc weight importance quantification function, based not only on the estimated effect of weight removal but also on the collateral contribution that an optimally tuned adjacent bias could give. Experimental results show a significant improvement against classical pruning strategies, suggesting that the interlaced action of parameters may be important in constructing effective compression techniques. We claim that this should not be too surprising: parameters of any neural network exist as a realization of a highly correlated random vector, that is the image of the initialization through the optimization procedure. This simple observation suggests that correlation between parameters (in our case, weights and biases) should not be neglected, and their consideration (for instance, with our compensation mechanism) could lead to high quality parameter compression techniques. In particular, while "atomic" weight importance metrics such as weight magnitude constitute a very efficient and sometimes effective strategy, we argue that the complex representation of information in modern neural networks requires non myopic approaches, possibly considering the effect of long range correlation between weights in different layers. This second idea is at the basis of the construction of the Δ function: a Taylor based expansion recovers the mechanics of the prediction for asymptotically small displacements of the weights, considering the effect across all the successive subnetwork. As a final point, we also would like to position our idea of using the output function rather than the loss function for constructing our importance metrics, despite the latter being certainly more frequent in literature. In particular, we observe that, at perfect convergence, the derivative of the loss function should be identically 0. Consequently, the ranking induced by loss-gradient-based metrics is dominated by the effects of numerical noise. We conjecture that this partially explains the behavior of the pruning technique based on the loss gradient and weight magnitude ($\mathcal{I}_{W^\ell_{ij}} = \partial_{W^\ell_{ij}} \mathcal{L}$) presented in the experiments, which has poorer performance than our proposed method. This is likely due to the influence of residuals, which vanish at convergence, thereby eliminating the advantage of such an approach.

In this work, we focus exclusively on real-valued fully connected neural networks, which constitute a fundamental general-purpose architecture. Although not reported here, based on encouraging results obtained, the proposed method may be extended to other architectures after appropriate studies.

Acknowledgements

The present research has received support from the project FIS, MUR, Italy 2025-2028, Project code: FIS-2023-02228, CUP: D53C24005440001, "SYNERGIZE: Synergizing Numerical Methods and Machine Learning for a new generation of computational models". L.M. and F.R. have received support from the project PRIN2022, MUR, Italy, 2023-2025, P2022N5ZNP "SIDDMs: shape-informed data-driven models for parametrized PDEs, with application to computational cardiology", funded by the European Union (Next Generation EU, Mission 4 Component 2). The authors of this work acknowledge the grant Dipartimento di Eccellenza 2023-2027, MUR, Italy. A.F., A.S., and F.R. are members of GNCS, "Gruppo Nazionale per il Calcolo Scientifico" (National Group for Scientific Computing) of INdAM (Istituto Nazionale di Alta Matematica).

References

[1] A. Alqahtani, X. Xie, and M. W. Jones. Literature review of deep network compression. Informatics, 8(4):77, Nov. 2021.

[2] M. Babaeizadeh, P. Smaragdis, and R. H. Campbell. Noiseout: A simple way to prune neural networks, 2016.

[3] E. Ballini. Flow and mechanics in fractured porous media: from high fidelity models to efficient reduced order solutions. Ph.d. thesis, Politecnico di Milano, Milan, Italy, 2025. Ph.D. Thesis, MOX, Department of Mathematics, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy.

[4] E. Ballini, A. Cominelli, L. Dovera, A. Forello, L. Formaggia, A. Fumagalli, S. Nardean, A. Scotti, and P. Zunino. Enhancing computational efficiency of numerical simulation for subsurface fluid-induced deformation using deep learning reduced order models. In SPE Reservoir Simulation Conference, 25RSC. SPE, Mar. 2025.

[5] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag. What is the state of neural network pruning?, 2020.

[6] H. Cheng, M. Zhang, and J. Q. Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(12):10558–10578, Dec. 2024.

[7] A. Ed Dyyany, A. Jamea, and A. Ammar. Neural networks for solving partial differential equations, a comprehensive review of recent methods and applications. SHS Web of Conferences, 214:01005, 2025.

[8] A. Engelbrecht. A new pruning heuristic based on variance analysis of sensitivity information. IEEE Transactions on Neural Networks, 12(6):1386–1399, 2001.

[9] A. Engelbrecht and I. Cloete. A sensitivity analysis algorithm for pruning feedforward neural networks. In Proceedings of International Conference on Neural Networks (ICNN'96), volume 2 of ICNN-96, pages 1274–1278. IEEE, 1996.

[10] N. Fnaiech, S. Abid, F. Fnaiech, and M. Cheriet. A modified version of a formal pruning algorithm based on local relative variance analysis. In First International Symposium on Control, Communications and Signal Processing, 2004., pages 849–852. IEEE, 2004.

[11] T. Gale, E. Elsen, and S. Hooker. The state of sparsity in deep neural networks, 2019.

[12] B. Hassibi and D. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In S. Hanson, J. Cowan, and C. Giles, editors, Advances in Neural Information Processing Systems, volume 5. Morgan-Kaufmann, 1992.

[13] S. A. Janowsky. Pruning versus clipping in neural networks. Physical Review A, 39(12):6600–6603, June 1989.

[14] M. Karlbauer, T. Praditia, S. Otte, S. Oladyshkin, W. Nowak, and M. V. Butz. Composing partial differential equations with physics-aware neural networks, 2021.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Technical Report CRP-TR-98-11, AT&T Labs, 1998.

[17] Y. LeCun, C. Cortes, and C. J. C. Burges. Mnist handwritten digit database. `https://yann.lecun.org/exdb/mnist/`, 2010. Accessed: 2025-12-18.

[18] Y. LeCun, J. Denker, and S. Solla. Optimal brain damage. In Advances in Neural Information Processing Systems, volume 2, 1989.

[19] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Machine Intelligence, 3(3):218–229, Mar. 2021.

[20] L. Mauch and B. Yang. A novel layerwise pruning method for model reduction of fully connected deep neural networks. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2382–2386. IEEE, Mar. 2017.

[21] L. Mauch and B. Yang. Least-squares based layerwise pruning of convolutional neural networks. In 2018 IEEE Statistical Signal Processing Workshop (SSP), pages 60–64. IEEE, June 2018.

[22] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance estimation for neural network pruning. In 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, June 2019.

[23] M. C. Mozer and P. Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In Proceedings of the 2nd International Conference on Neural Information Processing Systems (NIPS), pages 107–115, 1988.

[24] D. W. Otter, J. R. Medina, and J. K. Kalita. A survey of the usages of deep learning for natural language processing. IEEE Transactions on Neural Networks and Learning Systems, 32(2):604–624, Feb. 2021.

[25] P. Pant, R. Doshi, P. Bahl, and A. Barati Farimani. Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations. Physics of Fluids, 33(10), 2021.

[26] X. Qian and D. Klabjan. A probabilistic approach to neural network pruning, 2021.

[27] M. Raissi, P. Perdikaris, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378:686–707, 2019.

[28] R. Reed. Pruning algorithms-a survey. IEEE Transactions on Neural Networks, 4(5):740–747, 1993.

[29] V. Sanh, W. Thomas, and A. M. Rush. Movement pruning: Adaptive sparsity by fine-tuning. In Neural Information Processing Systems (NeurIPS), 2020.

[30] M. Takamoto, T. Praditia, R. Leiteritz, D. MacKinlay, F. Alesiani, D. Pflüger, and M. Niepert. Pdebench: An extensive benchmark for scientific machine learning, 2022.

[31] S. Vadera and S. Ameen. Methods for pruning deep neural networks. IEEE Access, 10:63280–63300, 2022.

[32] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis. Deep learning for computer vision: A brief review. Computational Intelligence and Neuroscience, 2018:1–13, 2018.

[33] X. Xiao, Z. Wang, and S. Rajasekaran. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.

[34] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis. Nisp: Pruning networks using neuron importance score propagation. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, June 2018.

[35] J. M. Zurada, A. Malinowski, and S. Usui. Perturbation method for deleting redundant inputs of perceptron networks. Neurocomputing, 14(2):177–193, Feb. 1997.

MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

22/2026 Ballini, E.; Muscarnera, L.; Fumagalli, A.; Scotti, A.; Regazzoni, F.
Elimination-compensation pruning for fully-connected neural networks

21/2026 Bottacini, G.; Torzoni, M.; Manzoni, A.
Neural Markov chain Monte Carlo: Bayesian inversion via normalizing flows and variational autoencoders

17/2026 Caldera, L.; Bottacini, G.; Cavinato, L.
MAGIC-Flow: multiscale adaptive conditional flows for generation and interpretable classification

20/2026 Caldera, L.; Cavinato, L.; Cirone, A.; Cama, I.; Garbarino, S.; Lodi, R.; Tagliavini, F.; Nigri, A.; De Francesco, S.; Cappozzo, A.; Piana, M.; Ieva, F.;
DISARM++: Beyond scanner-free harmonization

19/2026 Caldera, L.; Cavinato, L.; Ieva, F.
Scanner-agnostic MRI harmonization via SSIM-guided disentaglement

15/2026 Zecchi, A. A.; Sanavio, C.; Cappelli, L.; Perotto, S.; Roggero, A.; Succi, S.
Block encoding of sparse matrices with a periodic diagonal structure

14/2026 Agasisti, T.; Cannistrà, M.; Paganoni, A.M.
Nudging communication for students at risk: experimental evidence from an Italian university

13/2026 Dimola, N.; Coclite, A.; Zunino, P.
Neural Preconditioning via Krylov Subspace Geometry

12/2026 Corbetta A; Logan K.M.; Ferro M; Zuccolo L; Perola M.; Ganna A.; Di Angelantionio E.;Ieva F.
Longitudinal patterns of statin adherence and factors associated with decline in over one million individuals in Finland and Italy

11/2026 Cicalese, G.; Ciaramella, G.; Mazzieri, I.; Gander, M. J.
Optimized Schwarz Waveform Relaxation for the Damped Wave Equation