On the Effects of Pruning on Evolved Neural Controllers for Soft Robots

Giorgia Nadizar
giorgia.nadizar@studenti.units.it
giorgian@oslomet.no
Department of Engineering and Architecture, University of Trieste
Trieste, Italy
Department of Computer Science, Artificial Intelligence Lab, Oslo Metropolitan University
Oslo, Norway

Eric Medvet
Felice Andrea Pellegrino
Marco Zullich
emedvet@units.it
fapellegrino@units.it
marco.zullich@phd.units.it
Department of Engineering and Architecture, University of Trieste
Trieste, Italy

Stefano Nichele
stenic@oslomet.no
Department of Computer Science, Artificial Intelligence Lab, Oslo Metropolitan University
Oslo, Norway
Department of Holistic Systems, Simula Metropolitan Center for Digital Engineering
Oslo, Norway

ABSTRACT

Artificial neural networks (ANNs) are commonly used for controlling robotic agents. For robots with many sensors and actuators, ANNs can be very complex, with many neurons and connections. Removal of neurons or connections, i.e., *pruning*, may be desirable because (a) it reduces the complexity of the ANN, making its operation more energy efficient, and (b) it might improve the generalization ability of the ANN. Whether these goals can actually be achieved in practice is however still not well known. On the other hand, it is widely recognized that pruning in biological neural networks plays a fundamental role in the development of brains and their ability to learn. In this work, we consider the case of Voxel-based Soft Robots, a kind of robots where sensors and actuators are distributed over the body and that can be controlled with ANNs optimized by means of neuroevolution. We experimentally characterize the effect of different forms of pruning on the effectiveness of neuroevolution, also in terms of generalization ability of the evolved ANNs. We find that, with some forms of pruning, a large portion of the connections can be pruned without strongly affecting robot capabilities. We also observe sporadic improvements in generalization ability.

CCS CONCEPTS

• **Computing methodologies** → *Mobile agents*; *Continuous space search*; **Evolutionary robotics**; • **Computer systems organization** → **Evolutionary robotics**; • **Theory of computation** → **Evolutionary algorithms**.

KEYWORDS

Evolutionary robotics, Neuroevolution, Synaptic pruning

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07…$15.00
https://doi.org/10.1145/3449726.3463161

ACM Reference Format:
Giorgia Nadizar, Eric Medvet, Felice Andrea Pellegrino, Marco Zullich, and Stefano Nichele. 2021. On the Effects of Pruning on Evolved Neural Controllers for Soft Robots. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449726.3463161

1 INTRODUCTION

Artificial Neural Networks (ANNs) are continuously being deployed to solve a very diverse variety of tasks with great success. However, engineering ANNs is not an easy task. It requires domain expert knowledge, experience, and trial and error in order to identify a suitable architecture, network size, number of hidden layers, number of neurons and connections (or synapses), and many other relevant hyper-parameters. In some cases, the chosen network may be too simple for a given task and therefore achieving poor performances, while in other cases the chosen network may be too large and therefore hindering the learning process.

The recent trend of scaling to ever-larger neural networks, such as DALL-E, a 12 billion parameters version of GPT-3 [30], or Switch Transformers, a trillion parameters language models [6], has been criticized in terms of carbon footprint, energy consumption, and compute costs [38].

In stark contrast with ANNs, biological brains undergo a developmental process which initially creates a very large number of synapses, too many in fact [29]. While a large number of synapses is beneficial for faster incremental learning and allows for redundancy, it is not beneficial in the long term. Therefore, subsequently the brain is optimized through a rather large process of synaptic pruning.

The choice of a large fully-connected ANN may be considered a safe solution when the ideal topology for the task at hand is not known. Another alternative is to complexify the network using neuroevolution. In both cases, a relevant question is whether unnecessary and redundant connectivity may be removed after the training process has been completed, while preserving the network functionality.

ANNs with unnecessary connections may be problematic in case of physical implementations in robotic systems, where the wiring and resources are limited. In addition, larger networks require more

energy to run, which can be a bottleneck in resource-constrained robots. This is particularly relevant in the context of Voxel-based Soft Robots (VSRs), as the ones investigated through simulations in this work. VSRs are a class of modular robots made of connected soft components (voxels), that resemble biological soft tissues. Since in VSRs each voxel may contain sensing elements, actuators, as well as the neural controller itself [24], unnecessary neural network wiring is not desirable. In addition, if the robots had to be physically built, the energy resources would typically be rather limited.

In this work, we study the unstructured pruning of ANNs optimized with neuroevolution to control VSRs. More precisely, we incorporate pruning in the neuroevolution algorithm and we evaluate its effect in terms of performance of the obtained controller in a locomotion tasks. The study encompasses two VSR morphologies (biped and worm), three ANN topologies, and several pruning rates and pruning criteria (including random pruning). Our results show that the application of a proper pruning criterion during evolution can lead to controllers that are as effective as the ones obtained without pruning, and more effective than the controllers evolved without pruning and tested with pruning. Moreover, the evaluation of the performance achieved in untried terrains, suggests that pruning has only a slightly detrimental effect on the adaptability of the controller to similar tasks.

2 RELATED WORK

Synaptic pruning in the nervous system. It is observed that animals with larger brains are more likely to have higher learning abilities [31]. However, over an optimal network size threshold, adding more neurons and synapses deteriorates learning performance [29]. In addition, maintaining a large brain is expensive in terms of energy [20]. The developmental stage of the brain is characterized by hyperconnectivity, i.e., the formation of an excessive number of synapses. Superfluous synapses are to be removed. Failing in doing so may results in neural diseases, e.g., autism, epilepsy, schizophrenia [27]. Neural disorders are particularly affected by network topology. In particular, it is widely recognized that neural behaviors beneficial for computation and learning, such as power-law scaling of neural avalanches (criticality), are dependent on the network connectivity [13]. The mechanism that performs synaptic pruning is carried out by glial cells. In humans, synapses are eliminated from birth until the mid-twenties [43]. It has been shown that the process of synaptic pruning results in roughly half of the synapses to be eliminated until puberty, while the performance of the brain is retained [5]. In particular, the identified elimination strategy resulted in the pruning of weaker synapses first, i.e., deletion based on synaptic efficacy. Such biological findings raise questions in regards to artificial neural network controllers for artificial agents, and in particular the identification of suitable network sizes and number of parameters needed to learn a given task. The possibility of optimizing the networks in regards to learning, robustness to noise, and other factors such as energetic cost remains still an open area of research. In the following sections, several pruning strategies for artificial neural networks are reviewed.

Pruning in ANNs. Pruning in the context of Artificial Neural Networks (ANNs) is a *sparsification* technique consisting in the removal of connections (*synapses*) between neurons. It can be motivated by either efficiency (i.e., we want our network to train or evaluate faster) or by the belief that pruned ANNs are more robust or generalize better [15]. Early attempts at pruning ANNs in the Machine Learning (ML) environment include Optimal Brain Damage (OBD) [21] and L1-norm loss regularization [10] adapted from LASSO regression [33].

Pruning techniques for ANNs can be categorized as either *structured* or *unstructured* [3]. Structured pruning removes synapses from well-defined substructures of an ANN, such as a whole neuron or a convolutional filter in a Convolutional Neural Network. On the other hand, unstructured pruning removes connections without concern for the geometry of the ANN. Since structured pruning can lead to a *regular* pattern of sparsity on the parameters space, it is usually possible to directly take advantage of this sparsity as far as computation is concerned (e.g., if we remove one neuron from a fully-connected ANN, we can effectively reduce the dimension of the weights and bias matrices, thus leading to immediate computational gains). On the other hand, with unstructured pruning, the resulting *irregular* sparsity in the parameters tensors can be taken advantage of only via dedicated software [22] (e.g., CUSPARSE [26]) or hardware (e.g., NVIDIA Tesla A100[1]).

Hoefler et al. [15] list a large amount of heuristics for pruning an ANN. They can be categorized into (a) data-free heuristics, which in principle require no model evaluation to apply them, and (b) data–driven heuristics, which require the ANN to be evaluated on some given data points.

In this work, we will be adopting unstructured pruning solutions, adapting them to the context of neuroevolution. We will experiment with either data-free heuristics, namely *Least-Magnitude Pruning* (LMP) [4, 11], which prunes parameters exhibiting small magnitude, and data-driven heuristics, namely *Contribution Variance Pruning* (CVP) [40], which prunes parameters exhibiting low variance across multiple data instances. Moreover, we will utilize two pruning schemes similar in concept to CVP which prune connections manifesting low signals across data instances. Eventually, we will be employing random pruning (as done also in [7] and [42]) in order to obtain a "control group" to ensure whether the results obtained by means of the pruning schemes we exploited are indeed due to the heuristic and not due to randomness.

Pruning ANNs in the context of statistical learning. In the context of statistical learning, ANNs are typically trained iteratively (e.g., using any variant of stochastic gradient descent), updating the parameters after each iteration (e.g., using any variant of back-propagation).

In this background, a very useful resource for exploring various pruning paradigms is [15]. We can distinguish between different kinds of pruning techniques depending on whether the application of pruning is performed during or after the training phase. Usually, in the latter case, a performance drop is noticed in pruned networks: hence a re-training phase follows with various heuristics, after which the ANN may even improve the performance of the unpruned model even at high sparsity rates [7, 11, 22, 32]. In the literature, it is still a matter of debate what is the most effective re-training schedule [32, 41, 44], as the pressure is high to find well-performing

[1] See https://blogs.nvidia.com/blog/2020/05/14/sparsity-ai-inference/.

pruned ANNs trained in a time-efficient fashion, and how these pruned ANNs compare with respect to the unpruned counterpart [1, 2]. Nevertheless, the effect of unstructured pruning in helping with generalization is well-known in the literature (e.g., [32]). On the other hand, pruning techniques acting during training struggle to keep up with the performance of an analogous unpruned ANN at high pruning rates (for instance, larger than 90 %), even if recent advances such as [19] show very promising results.

Pruning ANNs in the context of neuroevolution. Unlike statistical learning, neuroevolution does not employ iterative training for ANNs. Rather, ANNs usually go through multiple phases of (a) fitness evaluation, and (b) variation, either via crossover and/or mutation [36]. One of the paramount approaches for neuroevolution is NEAT [37], which incorporates both crossover and mutation.

There exist works applying pruning phases in addition to the ones operated by NEAT or one of its variants. For instance, in [35], pruning is operated on neural controllers in a manner inspired by OBD. The need for pruning is motivated by numerical conditioning and empirical observations that EANT [18], a NEAT variant, was already removing a large number of parameters in the ANNs.

Recently, Gerum et al. [9] experimented with the application of random pruning to small neural controllers designed to navigate agents through a maze, concluding that pruning improved generalization. This work is of interest for our work since it presents approach and setting similar to our experiments, although our conclusions are different.

Pruning biologically-inspired ANNs. In the context of ANNs inspired by Biological Neural Networks, Spiking Neural Networks (SNNs), driven by the early work of Gerstner and Kistler [8], represent what has been called the "third generation of Neural Network models" [23]. Despite inheriting the fully-connected structures typical of Multilayer Perceptrons, they differ greatly from their statistical learning counterpart as (a) the input is encoded in a temporal rather than spatial structure, and (b) the training is operated in an unsupervised manner using Hebbian-based parameter update rules [12], thus detatching from the gradient-based methods of statistical learning.

Motivated by the aforementioned discoveries on human brain connectivity, some works have experimented with the application of pruning techniques to SNNs. For instance, Iglesias et al. [17] experimented with the application of a pruning heuristic similar to CVP to SNNs, although their work was not focused on producing high-performing models, rather observing the patterns of connectivity after various phases of pruning. Moreover, Shi et al. [34] experimented with applying LMP to SNNs during training. They were unable, though, to produce SNNs whose performance was comparable to those of the unpruned models.

3 VOXEL-BASED SOFT ROBOTS

In this study, we consider a kind of modular robots composed of several soft cubes (*voxel*) that are known as Voxel-based Soft Robots (VSRs) [14]. In particular, we experiment with a 2-D version of VSRs that can be easily simulated (in discrete time and continuous space) and hence favor optimization [25].

A VSR has a *morphology*, or body, and a *controller*, or brain. The morphology consists of the voxels composing the VSR, arranged in a 2-D grid. The voxels are equipped with sensors that can provide the controller with the information regarding the environment and the VSR itself. The controller is in charge of determining how the area of each voxel varies over the time, based on the readings of the sensors of the VSR.

3.1 VSR morphology

The morphology of a VSR is an arrangement of voxels, i.e., deformable squares in the 2-D case that we consider in this study, organized in a grid. Figure 2 shows two examples of VSR morphologies, both composed of 10 voxels.

In our simulation, voxels are modeled as compounds of spring-damper systems, masses, and distance constrains [25]: voxels that are at adjacent positions in the grid are rigidly connected at the vertices. Over the time, the area of each voxel changes based on (a) the external forces acting on the voxel (resulting from, e.g., other connected voxels or the ground) and (b) an expansion/contraction force dictated by the controller for that voxel. The latter effect is modeled in the simulation as an instantaneous change in the resting length of the spring-damper systems of the voxel. The change is linearly dependent on the *actuation value* for that voxel: at each time step, the actuation value is assigned by the controller and is defined in $[-1, 1]$, where -1 corresponds to maximum requested expansion and 1 corresponds to maximum requested contraction.

A VSR can be equipped with sensors, that are located in its voxels. In this work, we consider four types of sensor, described below, and put at most one sensor of each type in each voxel of the VSR. At each time step, a sensor S outputs a *sensor reading* $\boldsymbol{r}_S \in [0, 1]^m$, with m being the dimensionality of the sensor type. Sensors of type *area* sense the ratio between the current area of the voxel and its rest area ($m = 1$). Sensors of type *touch* sense if the voxel is in contact with the ground or not and output a value being 1 or 0, respectively ($m = 1$). Sensors of type *velocity* sense the velocity of the center of mass of the voxel along the voxel x- and y-axes ($m = 2$). Finally, sensors of type *vision* sense the distances towards closest objects along a predefined set of directions: for each direction, the corresponding element of the sensor reading $\boldsymbol{r}_S$ is the distance of the closest object, if any, from the voxel center of mass along that direction. If the distance is greater than a threshold d, it is clipped to d. We use the vision sensor with the following directions with respect to the voxel positive x-axis: $-\frac{1}{4}\pi$, $-\frac{1}{8}\pi$, 0, $\frac{1}{8}\pi$, $\frac{1}{4}\pi$; the dimensionality is hence $m = 5$. Velocity and vision sensors employ a soft normalization of the outputs, using the tanh function and rescaling, to ensure that the output is defined in $[0, 1]^m$.

3.2 VSR controller

In this work, we take inspiration from [39] and use ANNs as controllers for the VSR. In particular, we use a fully connected feed-forward NN, also known as multilayer perceptron, with a number of input neurons corresponding to the overall number of sensor readings and a number of outputs corresponding to the number of voxels in the VSR. Other forms of controller may be employed in VSRs, including ANNs that are distributed over the VSR body [24].

At each time step, the controller takes the concatenation $\boldsymbol{r} = \left[\boldsymbol{r}_{S_1}\ \boldsymbol{r}_{S_2}\ \dots\right]$ of the current sensor readings, feeds it to the ANN, and uses its output $\boldsymbol{a} \in [-1,1]^n = f_{\boldsymbol{\theta}}(\boldsymbol{r})$ as actuation values for the n voxels composing the VSR. We use tanh as activation function in the neurons of the ANN.

The controller is defined by its parameters $\boldsymbol{\theta} \in \mathbb{R}^p$, with p depending on the ANN *topology*, i.e., the number and size of the ANN layers—we recall that the size of the input and output layers are determined by the sensors the VSR is equipped with and the number of voxels, respectively.

Given a morphology, a VSR can be optimized for a given task by optimizing the controller parameters $\boldsymbol{\theta}$.

4 PRUNING TECHNIQUES

We consider different forms of pruning of a fully connected feed-forward ANN. They share a common working scheme and differ in three parameters that define an instance of the scheme: the *scope*, i.e., the subset of connections that are considered for the pruning, the *criterion*, defining how those connections are sorted in order to decide which ones are to be pruned first, and the *pruning rate*, i.e., the rate of connections in the scope that are actually pruned. In all cases, the pruning of a connection corresponds to setting to 0 the value of the corresponding element θ_i of the network parameters vector $\boldsymbol{\theta}$.

Since we are interested in the effects of pruning of ANNs used as controllers for robotic agents, we assume that the pruning can occur during the life of the agent, at a given time. As a consequence, we may use information related to the working of the network up to the pruning time, as, e.g., the actual values computed by the neurons, when defining a criterion.

Algorithm 1 shows the general scheme for pruning. Given the vector $\boldsymbol{\theta}$ of the parameters of the ANN, we first partition its elements, i.e., the connections between neurons, using the scope parameter (as detailed below): in Algorithm 1, the outcome of the partitioning is a list $(\boldsymbol{i}_1, \dots, \boldsymbol{i}_n)$ of lists of indices of $\boldsymbol{\theta}$. Then, for each partition, we sort its elements according to the criterion, storing the result in a list of indices $\boldsymbol{i}$. Finally, we set to 0 the $\boldsymbol{\theta}$ elements corresponding to an initial portion of $\boldsymbol{i}$: the size of the portion depends on the pruning rate ρ and is $\lfloor |\boldsymbol{i}| \rho \rfloor$.

```
1  function prune(θ):
2      (i_1, ..., i_n) ← partition(θ, scope)
3      foreach j ∈ {1, ..., n} do
4          i ← sort(i_j, criterion)
5          foreach k ∈ {1, ..., ⌊|i|ρ⌋} do
6              θ_i ← 0
7          end
8      end
9      return θ
10 end
```

Algorithm 1: The algorithm for pruning a vector $\boldsymbol{\theta}$ of ANN parameters given the parameters scope, criterion, and ρ.

We explore three options for the scope parameter and five for the criterion parameter; concerning the pruning rate $\rho \in [0,1]$, we experimented with many values (see Section 5).

For the scope, we have:

- *Network*: all the connections are put in the same partition.
- *Layer*: connections are partitioned according to the layer of the destination neuron (also called post-synaptic neuron).
- *Neuron*: connections are partitioned according to the destination neuron.

For the criterion, we have:

- *Weight*: connections are sorted according to the absolute value of the corresponding weight. This corresponds to LMP (see Section 2).
- *Signal mean*: connections are sorted according to the mean value of the signal they carried from the beginning of the life of the robot to the pruning time.
- *Absolute signal mean*: similar to the previous case, but considering the mean of the absolute value.
- *Signal variance*: similar to the previous case, but considering the variance of the signal. This corresponds to CVP (see Section 2).
- *Random*: connections are sorted randomly.

All the criteria work with ascending ordering: lowest values are pruned first. Obviously, the ordering does not matter for the random criterion. When we use the signal variance criterion and prune a connection, we take care to adjust the weight corresponding to the bias of the neuron the pruned connection goes to by adding the signal mean of the pruned connection: this basically corresponds to making that connection carry a constant signal.

We highlight that the three criteria based on signal are data-driven; on the contrary, the weight and the random criteria are data-free. In other words, signal-based criteria operate based on the experience the ANN acquired up to pruning time. As a consequence, they constitute a form of adaptation acting on the time scale of the robot life, that is shorter than the adaptation that occurs at the evolutionary time scale; that is, they are a form of *learning*. As such, we might expect that, on a given robot that acquires different experiences during the initial stage of its life the pruning may result in different outcomes. Conversely, the weight criterion always results in the same outcome, given the same robot. In principle, hence, signal-based criteria might result in a robot being able to adapt and perform well also in conditions that are different than those used for the evolution. We verified experimentally this hypothesis: we discuss the results in Section 5.

5 EXPERIMENTS AND RESULTS

We performed several experiments in order to answer the following research questions:

- RQ1 Is the evolution of effective VSR controllers adversely affected by pruning? Does the impact depend on the type of pruning and on the size of the ANN?
- RQ2 Does pruning have an impact on the adaptability of the evolved VSR controllers to different tasks? Does the impact depend on the type of pruning and on the size of the ANN?

For answering these questions, we evolved the controller for two different robots, each with three different ANN topologies: during the evolution, we enabled different variants of pruning, including, as a baseline, the case of no pruning. We considered the task of

locomotion, in which the goal for the robot is to travel as fast as possible on a terrain. We describe in detail the experimental procedure and discuss the results in Section 5.2.

After the evolution, we took each evolved robot and measured its performance in locomotion on a set of terrains different than the one used during the evolution, in order to assess the adaptability of the robot. We describe the procedure and discuss the results in Section 5.3.

In order to reduce the number of variants of pruning to consider when answering RQ1 and RQ2, we first performed a set of experiments to assess the impact of pruning in a static context, i.e., in ANNs not subjected to evolutionary optimization. We present these experiments and our findings in the next section.

5.1 Static characterization of pruning variants

We aimed at evaluating the effect of different forms of pruning on ANNs in terms of how the output changes with respect to no pruning, given the same input. In order to make this evaluation significant with respect to the use case of this study, i.e., ANNs employed as controllers for VSRs, we considered ANNs with a topology that resembles the one used in the next experiments and fed them with inputs that resemble the readings of the sensors of a VSR doing locomotion.

In particular, for the ANN topology we considered three input sizes $n_{\text{input}} \in \{10, 25, 50\}$ and three depths $n_{\text{layers}} \in \{0, 1, 2\}$, resulting in $3 \times 3 = 9$ topologies, all with a single output neuron. For the topologies with inner layers, we set the inner layer size to the size of the input layer. In terms of the dimensionality p of the vector $\boldsymbol{\theta}$ of the parameters of the ANN, the considered ANN topologies corresponding to values ranging from $p = (10 + 1)1 = 11$, for $n_{\text{input}} = 10$ and $n_{\text{layers}} = 0$, to $p = (50{+}1)(50{+}1)(50{+}1)1132\,651$, for $n_{\text{input}} = 50$ and $n_{\text{layers}} = 2$, where the +1 is the bias. We instantiated 10 ANNs for each topology, setting $\boldsymbol{\theta}$ by sampling the multivariate uniform distribution $U(-1, 1)^p$ of appropriate size, hence obtaining 90 ANNs.

Concerning the input, we fed the network with sinusoidal signals with different frequencies for each input, discretized in time with a time step of $\Delta t = \frac{1}{10}$ s. Precisely, at each time step k, with $t = k\Delta t$, we set the ANN input to $\boldsymbol{x}^{(k)}$, with $x_i^{(k)} = \sin\left(\frac{k\Delta t}{i{+}1}\right)$, and we read the single output $y^{(k)} = f_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(k)}\right)$.

We considered the 3×5 pruning variants (scope and criteria) and 20 values for the pruning rate ρ, evenly distributed in $[0, 0.75]$. We took each one of the 90 ANNs and each one of the 300 pruning variants, we applied the periodic input for 10 s, triggering the actual pruning at $t = 5$ s, and we measured the mean absolute difference e of the output $f_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(k)}\right)$ during the last 5 s, i.e., after pruning, to the output $f_{\hat{\boldsymbol{\theta}}}\left(\boldsymbol{x}^{(k)}\right)$ of the corresponding unpruned ANN:

$$e = \frac{1}{50} \sum_{k=50}^{k=100} \left\| f_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(k)}\right) - f_{\hat{\boldsymbol{\theta}}}\left(\boldsymbol{x}^{(k)}\right) \right\|. \tag{1}$$

Figure 1 summarizes the outcome of this experiment. It displays one plot for each ANN topology (i.e., combination of n_{layer} and n_{input}) and one line showing the mean absolute difference e, averaged across the 10 ANNs with that topology, vs. the pruning

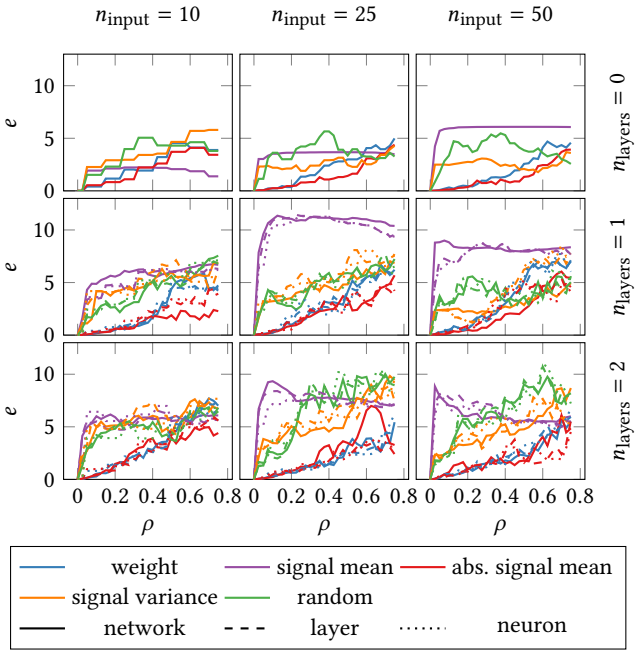


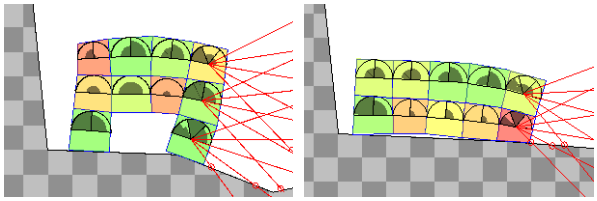
Figure 1: Mean absolute difference e between the output of a pruned ANN and the output of the corresponding unpruned ANN vs. the pruning rate ρ, for different ANN structures and with different pruning criteria (color) and scopes (linetype).

rate ρ for each pruning variant: the color of the line represents the criterion, the line type represents the context. Larger ANNs are shown in the bottom right matrix of plots.

By looking at Figure 1 we can do the following observations. First, the factor that appears to have the largest impact on the output of the pruned ANN is the criterion (the color of the line in Figure 1). Weight and absolute signal mean criteria consistently result in lower values for the difference e, regardless of the scope and of the pruning rate. On the other hand, with the signal mean criterion, e becomes large even with low pruning rates: for $\rho > 0.1$ there seems to be no further increase in e. Interestingly, the random criterion appears to be less detrimental, in terms of e, than signal mean in the vast majority of cases. We explain this finding by the kind of input these ANNs have been fed with, that is, sinusoidal signals: the mean of signals that are periodic with a period shorter enough than the time before the pruning is close to 0 and this results in connections actually carrying some information to be pruned. We recall that we chose to use sinusoidal signals because they are representative of the sensor readings a VSR doing locomotion could collect, in particular when exhibiting an effective gait, that likely consists of movements that are repeated over the time.

Second, apparently, there are no bold differences among the three values for the scope parameter. As expected, for the shallow ANNs (with $n_{\text{layers}} = 0$) the scope parameter does not play any role, since there is one single layer and one single output neuron (being the same destination for all connections).

Third, the pruning rate ρ impacts on e as expected: in general, the larger ρ, the larger e. However, the way e changes by increasing



(a) Biped (b) Worm

Figure 2: Frames of the two VSR morphologies used in the experiments. The color of each voxel encodes the ratio between its current area and its rest area: red indicated contraction, yellow rest state, and green expansion. The circular sector drawn at the center of each voxel indicates the current sensed values: subsectors represent sensors and are, where appropriate, internally divided in slices according to the sensor dimensionality m. The rays of the vision sensors are shown in red.

ρ seems to depend on the pruning criterion: for weight and absolute signal mean, Figure 1 suggests a linear dependency. For the other criteria, e quickly increases with ρ and then remains stable, for signal mean, or slowly increases, for signal variance and random.

Fourth and finally, the ANN topology appears to play a minor role in determining the impact of pruning. The ANN depth (i.e., n_{layers}) seems to impact slightly on the difference between pruning variants: the deeper the ANN, the fuzzier the difference. Concerning the number of inputs n_{input}, by looking at Figure 1 we are not able to extract any strong claim.

Based on the results of this experiment, summarized in Figure 1, we decided to consider only weight, absolute signal mean, and random criteria and only the network scope for the next experiments.

5.2 RQ1: impact on the evolution

In order to provide an answer to this question, we evolved the controller for six VSRs, resulting from the combination of two morphologies and three ANN topologies. For each combination, we optimized the weights of the ANN with and without pruning.

Figure 2 shows the two morphologies. Both consist of 10 voxels and have several sensors. We put area sensors in each voxel, velocity sensors in the voxels in the top row, touch sensors in the voxels in the bottom row (just the two "legs" for the biped), and vision sensors in the voxels of the rightmost column. As a result, both morphologies correspond to the same number of inputs and outputs for the ANN, respectively 35 and 10.

Concerning the ANN topologies, we experimented with $n_{\text{layers}} \in \{0, 1, 2\}$. For the ANN with inner layers, we set the size of those layers to 35. These settings resulted in the size p of the parameter vector $\boldsymbol{\theta}$ to be 360, 1620, and 2880, respectively.

For each of the six combinations of morphology and ANN topology, we used three different pruning criteria: weight, absolute signal mean, and random, all with network scope, as thoroughly described in Section 4. For each criterion, we employed the following pruning rates: $\rho \in \{0.125, 0.25, 0.5, 0.75\}$. Furthermore, we evolved, for each combination, an ANN without pruning to have a baseline for meaningful comparisons.

To perform evolution we used the simple evolutionary algorithm (EA) described in Algorithm 2, a form of evolutionary strategy. At first, n_{pop} individuals, i.e., numerical vectors $\boldsymbol{\theta}$, are put in the initially empty population, all generated by assigning to each element of the vector a randomly sampled value from a uniform distribution over the interval $[-1, 1]$. Subsequently, n_{gen} evolutionary iterations are performed. On every iteration, which corresponds to a generation, the fittest quarter of the population is chosen to generate $n_{\text{pop}} - 1$ children, each obtained by adding values sampled from a normal distribution $N(0, \sigma)$ to the element-wise mean $\boldsymbol{\mu}$ of all parents. The generated offspring, together with the fittest individual of the previous generation, end up forming the population of the next generation, which maintains the fixed size n_{pop}.

We used the following EA parameters: $n_{\text{pop}} = 48$, $n_{\text{gen}} = 416$ (corresponding to 20 000 fitness evaluations), and $\sigma = 0.35$. We verified that, with these values, the evolution was in general capable of converging to a solution, i.e., longer evolutions would have resulted in negligible fitness improvements.

```
1  function evolve():
2      P ← ∅
3      foreach i ∈ {1, ..., n_pop} do
4          P ← P ∪ {0 + U(−1, 1)^p}
5      end
6      foreach g ∈ {1, ..., n_gen} do
7          P_parents ← bestIndividuals(P, ⌊|P|/4⌋)
8          μ ← mean(P_parents)
9          P' ← {bestIndividuals(P, 1)}
10         while |P'| < n_pop do
11             P' ← P' ∪ {μ + N(0, σ)^p}
12         end
13         P ← P'
14     end
15     return bestIndividuals(P, 1)
16 end
```

Algorithm 2: The simple EA used for neuroevolution.

We optimized VSRs for the task of *locomotion*: the goal of the VSR is to travel as fast as possible on a terrain along the positive x axis. We quantified the degree of achievement of the locomotion task of a VSR by performing a simulation of duration t_f and measuring the VSR average velocity $v_x = \frac{x(t_f) - x(t_i)}{t_f - t_i}$, $x(t)$ being the position of the robot center of mass at time t and t_i being the initial time of assessment. In the EA of Algorithm 2 we hence used v_x as fitness for selecting the best individuals. We set $t_f = 60$ s and $t_i = 20$ s to discard the initial transitory phase. For the controllers with pruning, we set the pruning time at $t_p = 20$ s.

We remark that the EA of Algorithm 2 constitutes a form of Darwinian evolution with respect to pruning: the effect of pruning on an individual does not impact on the genetic material that is passed to the offspring by that individual. More precisely, the element-wise mean $\boldsymbol{\mu}$ is computed by considering the parents $\boldsymbol{\theta}$ vectors before the pruning. We leave the investigation on pruning effects on neuroevolution in the case of Lamarckian evolution to future work.

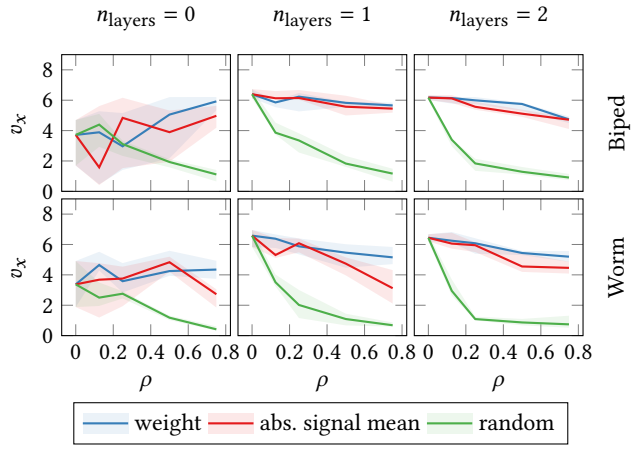


Figure 3: Fitness v_x (median with lower and upper quartiles across the 10 repetitions) vs. pruning rate ρ, for different pruning criteria (color), VSR morphologies (plot row), and ANN topologies (plot column).

For favoring generalization, we evaluated each VSR on a different randomly generated hilly terrain, i.e., a terrain with hills of variable heights and distances between each other. To avoid propagating VSRs that were fortunate in the random generation of the terrain, we re-evaluated, on a new terrain, the fittest individual of each generation before moving it to the population of the next generation.

For each of the 2×3×(3×4+1) combinations of VSR morphology, ANN topology, pruning criterion, and pruning rate (the +1 being associated with no pruning) we performed 10 independent, i.e., with different random seeds, evolutionary optimizations of the controller with the aforementioned EA. We hence performed 780 evolutionary optimizations.

We used 2D-VSR-Sim [25] for the simulation, setting all the parameters to default values. We performed the experiments on a few C2 VM instances of Google Compute Engine cloud infrastructure, each equipped with 16 vCPUs. On average, a single simulation, i.e., a fitness evaluation, took ≈ 6 s (wall time on a single vCPU); an entire evolutionary optimization took ≈ 9400 s (wall time on the instance).

Figure 3 summarizes the findings of this experiment. In particular, the plots show how the pruning rate ρ impacts the fitness of the best individual of the last generation, for the different VSR morphologies and ANN topologies employed in the experiment.

What immediately pops out from the plots is how individuals whose controllers have been pruned with weight or absolute signal mean criteria significantly outperform those who have undergone random pruning. This suggests that randomly pruning controllers at each fitness evaluation is detrimental to their evolution. In fact, individuals with a good genotype could perform poorly after the removal of important connections, while others could surpass them thanks to a luckier pruning, hence the choice of fittest individuals for survival and reproduction could be distorted. Moreover, Figure 3 confirms that the heuristics employed, based on weight and absolute signal mean criteria (Section 4), successfully choose connections that are less important for the controller to be removed, thus limiting the damage of connection removal.

In addition, comparing the subplots of Figure 3 there are no bold differences between the first and the second row, which leads us to conclude that the morphology of the VSR does not play a key role in determining the impact of pruning on the performances of the controller. On the contrary, different ANN topologies seem to be affected differently by pruning. The subplots of the first column ($n_{\text{layers}} = 0$), in particular, suggest pruning could have a beneficial impact on shallow networks. However, the upper and lower quartiles reveal that the distribution of the fitness v_x is spread across a considerably large interval, hence it is difficult to draw any sharp conclusion on the possible benefits of pruning for such controllers. Differently, for ANN topologies with $n_{\text{layers}} \in \{1, 2\}$, we can note that a higher pruning rate ρ leads to weaker performances of the controller. In this case, the result is in line with our expectations, as an increasing ρ means that we are removing more connections from the ANN, thus making it harder for the signal to spread across neurons. Nevertheless, controllers pruned with a proper heuristic have evolved to achieve results comparable to those who have not undergone pruning during their evolution, considered here as baseline. We performed a Mann-Whitney U test with the null hypothesis that, for each combination of VSR morphology, ANN topology, pruning criterion, and pruning rate ρ, the distribution of the best fitness is the same as obtained from the corresponding baseline controller, i.e., with the same VSR morphology and ANN topology, evolved without pruning, and we found that the p-value is greater than 0.05 in 30 out of 72 cases, 26 of which are with $\rho \leq 0.25$.

Based on the results of Figure 3, we speculate that controllers with weight and absolute signal mean pruning criteria look robust to pruning because they result from an evolution in which VSRs are subjected to pruning, rather than because those kinds of pruning are, in general, not particularly detrimental. To test this hypothesis, we carried out an additional experiment. We took all the best individuals of the last generations and we re-assessed them (on a randomly generated hilly terrain similar to the one used in evolution). For the individuals that were evolved without pruning, we performed 3×4 additional evaluations, introducing pruning after $t_p = 20$ s with the previously mentioned 3 criteria and 4 rates ρ.

Figure 4 shows the outcome of this experiment, i.e., v_x on the re-assessment plotted against the validation pruning rate ρ for both individuals evolved with (solid line) and without (dashed line) pruning. The foremost finding is that individuals evolved with pruning visibly outperform the ones whose ancestors have not experienced pruning, for almost all pruning rates. This corroborates the explanation we provided above, that is, VSRs whose ancestors evolved experiencing pruning are more robust to pruning than VSRs that evolved without pruning.

Besides analyzing the aggregate results, we also examined the behavior of a few evolved VSRs in a comparative way, i.e., with and without pruning in re-assessment. We found that, interestingly, in some cases the VSR starts to move effectively only after the pruning: this might suggest that pruning shaped the evolutionary path at the point that the lack of pruning becomes detrimental, similarly to what happens in the brain of complex animals (see

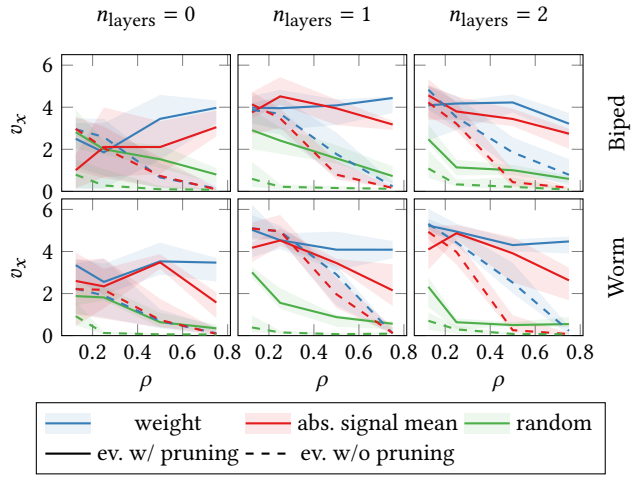


Figure 4: Median, lower quartiles, and upper quartiles of validation velocity v_x vs. validation pruning rate ρ of individuals evolved with and without pruning for different ANN topologies and VSR morphologies.

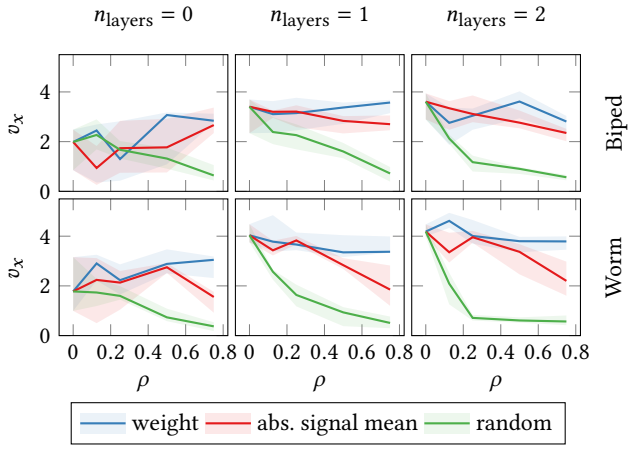


Figure 5: Median, lower quartiles, and upper quartiles of validation velocity v_x vs. validation pruning rate ρ averaged across validation terrains fir different pruning criteria, VSR morphologies, and ANN topologies.

Section 2). We provide videos of a few VSRs exhibiting a change in their behaviour after pruning at https://youtu.be/-HCHDEb9azY, https://youtu.be/oOtJKri6vyw, and https://youtu.be/uwrtNezTrx8.

5.3 RQ2: impact on the adaptability

For the sake of this research question, we defined VSR controllers as *adaptable* if they are able to effectively accomplish locomotion on terrains they "have never seen before"—i.e., terrains that none of their ancestors ever experienced locomotion on. Hence, to assess the adaptability of evolved controllers, we measured the performance in locomotion of the best individuals of the last generations on a set of different terrains. We experimented with the following terrains: (a) flat, (b) hilly with 6 combinations of heights and distances between hills, (c) steppy with 6 combinations of steps heights and widths, (d) downhill with 2 different inclinations, and (e) uphill with 2 different inclinations. As a result, each individual was re-assessed on a total of 17 different terrains. Note that, in this experiment, controllers were not altered in between evolution and re-assessment, i.e., they were re-evaluated with the same pruning criterion, if any, and pruning rate ρ as experienced during evolution.

Figure 5 displays the outcome of this experiment. Namely, for each of the different VSR morphologies, ANN topologies, and pruning criteria, the validation velocity v_x (i.e., the re-assessment velocity averaged on the 17 terrains) is plotted against the pruning rate ρ. The results in Figure 5 are coherent with the findings of Section 5.2: comparing the subplots, we can conclude that the morphology of the VSR is not relevant in determining the impact of pruning on the adaptability, whereas the ANN topology plays a key role. More in details, for shallow networks, pruning seems to enhance adaptability, whereas it has a slightly detrimental effect for deeper networks (that are, however, in general better than shallow ones). Anyway, for controllers evolved employing weight or absolute signal mean pruning criteria, the validation results are comparable to those of controllers evolved without pruning. We performed a Mann-Whitney U test with the null hypothesis that, for each combination of VSR morphology, ANN topology, pruning criterion, and pruning rate ρ, the distribution of the average validation velocities across all terrains is the same as the one obtained from the validation of the corresponding baseline controller, i.e., with the same VSR morphology and ANN topology, evolved without pruning, and we found that the p-value is greater than 0.05 in 38 out of 72 cases, 12 of which are with $\rho \leq 0.25$.

6 CONCLUDING REMARKS

We investigated the effects of pruning on evolved neural controllers for Voxel-based Soft Robots (VSR). In particular, we aimed at evaluating whether this biologically inspired technique could impact artificial agents similarly to living creatures, i.e., favouring adaptability, or if it would prove detrimental for the resulting individuals. To do so, we considered the task of locomotion and we evolved the controller of VSRs with two morphologies (biped and worm) and three ANN topologies employing several pruning criteria and pruning rates. Our experimental results show that the application of pruning with a limited rate and a proper criterion during evolution can result in individuals that are comparable to those obtained without pruning, as well as more robust to pruning than the latter ones. In addition, we have shown that individuals evolved with pruning do not appear significantly less adaptable to different tasks, i.e., locomotion on unseen terrains, than those evolved without pruning.

As an extension of this work, it might be noteworthy to explore the effects of pruning on more biologically plausible neural controllers as, e.g., those based on spiking neural networks [28]. Moreover, the relationship between pruning and forms of regeneration of the controller [16] might be studied.

ACKNOWLEDGMENTS

The experimental evaluation of this work has been partially supported by a Google Faculty Research Award granted to E.M.. This work was partially funded by the Norwegian Research Council (NFR) through their IKTPLUSS research and innovation action under the project Socrates (grant agreement 270961) and Young Research Talent program under the project DeepCA (grant agreement 286558). G.N. was partially supported by NordSTAR - Nordic Center for Sustainable and Trustworthy AI Research (OsloMet Project Code 202237-100).

REFERENCES

[1] Alessio Ansuini, Eric Medvet, Felice Andrea Pellegrino, and Marco Zullich. 2020. Investigating Similarity Metrics for Convolutional Neural Networks in the Case of Unstructured Pruning. In *International Conference on Pattern Recognition Applications and Methods*. Springer, 87–111.

[2] Alessio Ansuini, Eric Medvet, Felice Andrea Pellegrino, and Marco Zullich. 2020. On the Similarity between Hidden Layers of Pruned and Unpruned Convolutional Neural Networks.. In *ICPRAM*. 52–59.

[3] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. 2017. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 13, 3 (2017), 1–18.

[4] Christopher M Bishop. 1995. *Neural Networks for Pattern Recognition*. Clarendon Press, Chapter 9.5.3 - Saliency of Weights.

[5] Gal Chechik, Isaac Meilijson, and Eytan Ruppin. 1999. Neuronal regulation: A mechanism for synaptic pruning during brain maturation. *Neural computation* 11, 8 (1999), 2061–2080.

[6] William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *arXiv preprint arXiv:2101.03961* (2021).

[7] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJl-b3RcF7

[8] Wulfram Gerstner and Werner M Kistler. 2002. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.

[9] Richard C Gerum, André Erpenbeck, Patrick Krauss, and Achim Schilling. 2020. Sparsity through evolutionary pruning prevents neuronal networks from overfitting. *Neural Networks* 128 (2020), 305–312.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press, Chapter 7.1.2 - L1 Regularization.

[11] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both Weights and Connections for Efficient Neural Network. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 1135–1143.

[12] Donald Olding Hebb. 2005. *The organization of behavior: A neuropsychological theory*. Psychology Press.

[13] Kristine Heiney, Ola Huse Ramstad, Vegard Fiskum, Nicholas Christiansen, Axel Sandvig, Stefano Nichele, and Ioanna Sandvig. 2021. Criticality, connectivity, and neural disorder: A multifaceted approach to neural computation. *Frontiers in computational neuroscience* 15 (2021), 7.

[14] Jonathan Hiller and Hod Lipson. 2012. Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics* 28, 2 (2012), 457–466.

[15] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *arXiv preprint arXiv:2102.00554* (2021).

[16] Kazuya Horibe. 2021. Regenerating Soft Robots through Neural Cellular Automata. In *Genetic Programming: 24th European Conference, EuroGP 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings*. Springer Nature, 36.

[17] Javier Iglesias, Jan Eriksson, François Grize, Marco Tomassini, and Alessandro EP Villa. 2005. Dynamics of pruning in simulated large-scale spiking neural networks. *Biosystems* 79, 1-3 (2005), 11–20.

[18] Yohannes Kassahun and Gerald Sommer. 2005. Efficient reinforcement learning through Evolutionary Acquisition of Neural Topologies.. In *ESANN*. 259–266.

[19] Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. 2020. Soft Threshold Weight Reparameterization for Learnable Sparsity. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Hal Daumé III and Aarti Singh (Eds.), Vol. 119. PMLR, 5544–5555. http://proceedings.mlr.press/v119/kusupati20a.html

[20] Simon B Laughlin, Rob R de Ruyter van Steveninck, and John C Anderson. 1998. The metabolic cost of neural information. *Nature neuroscience* 1, 1 (1998), 36–41.

[21] Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. 1989. Optimal brain damage.. In *NIPs*, Vol. 2. Citeseer, 598–605.

[22] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2019. Rethinking the Value of Network Pruning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJlnB3C5Ym

[23] Wolfgang Maass. 1997. Networks of spiking neurons: the third generation of neural network models. *Neural networks* 10, 9 (1997), 1659–1671.

[24] Eric Medvet, Alberto Bartoli, Andrea De Lorenzo, and Giulio Fidel. 2020. Evolution of distributed neural controllers for voxel-based soft robots. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 112–120.

[25] Eric Medvet, Alberto Bartoli, Andrea De Lorenzo, and Stefano Seriani. 2020. 2D-VSR-Sim: A simulation tool for the optimization of 2-D voxel-based soft robots. *SoftwareX* 12 (2020).

[26] M Naumov, L Chien, P Vandermersch, and U Kapasi. 2010. Cusparse library. In *GPU Technology Conference*.

[27] Urte Neniskyte and Cornelius T Gross. 2017. Errant gardeners: glial-cell-dependent synaptic pruning and neurodevelopmental disorders. *Nature Reviews Neuroscience* 18, 11 (2017), 658.

[28] Sidney Pontes-Filho and Stefano Nichele. 2019. Towards a framework for the evolution of artificial general intelligence. *arXiv preprint arXiv:1903.10410* (2019).

[29] Dhruva Venkita Raman, Adriana Perez Rotondo, and Timothy O'Leary. 2019. Fundamental bounds on learning performance in neural circuits. *Proceedings of the National Academy of Sciences* 116, 21 (2019), 10537–10546.

[30] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092* (2021).

[31] Simon M Reader and Kevin N Laland. 2002. Social intelligence, innovation, and enhanced brain size in primates. *Proceedings of the National Academy of Sciences* 99, 7 (2002), 4436–4441.

[32] Alex Renda, Jonathan Frankle, and Michael Carbin. 2020. Comparing Fine-tuning and Rewinding in Neural Network Pruning. In *International Conference on Learning Representations*.

[33] Fadil Santosa and William W Symes. 1986. Linear inversion of band-limited reflection seismograms. *SIAM J. Sci. Statist. Comput.* 7, 4 (1986), 1307–1330.

[34] Yuhan Shi, Leon Nguyen, Sangheon Oh, Xin Liu, and Duygu Kuzum. 2019. A soft-pruning method applied during training of spiking neural networks for in-memory computing applications. *Frontiers in neuroscience* 13 (2019), 405.

[35] Nils T Siebel, Jonas Botel, and Gerald Sommer. 2009. Efficient neural network pruning during neuro-evolution. In *2009 International Joint Conference on Neural Networks*. IEEE, 2920–2927.

[36] Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. 2019. Designing neural networks through neuroevolution. *Nature Machine Intelligence* 1, 1 (2019), 24–35.

[37] Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.

[38] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. *arXiv preprint arXiv:1906.02243* (2019).

[39] Jacopo Talamini, Eric Medvet, Alberto Bartoli, and Andrea De Lorenzo. 2019. Evolutionary Synthesis of Sensing Controllers for Voxel-based Soft Robots. In *Artificial Life Conference Proceedings*. MIT Press, 574–581.

[40] Georg Thimm and Emile Fiesler. 1995. Evaluating pruning methods. In *Proceedings of the International Symposium on Artificial neural networks*. Citeseer, 20–25.

[41] Haoran You, Chaojian Li, Pengfei Xu, Yonggan Fu, Yue Wang, Xiaohan Chen, Richard G Baraniuk, Zhangyang Wang, and Yingyan Lin. 2019. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint arXiv:1909.11957* (2019).

[42] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. 2019. Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/1113d7a76ffceca1bb350bfe145467c6-Paper.pdf

[43] Wendy Zukerman and Andrew Purcell. 2011. Brain's synaptic pruning continues into your 20s.

[44] Marco Zullich, Eric Medvet, Felice Andrea Pellegrino, and Alessio Ansuini. 2021. Speeding-Up Pruning for Artificial Neural Networks: Introducing Accelerated Iterative Magnitude Pruning. *2020 25th International Conference on Pattern Recognition (ICPR)* (2021).