LETTER **Communicated by Mariia Golovianko**

Neural Networks With Disabilities: An Introduction to Complementary Artificial Intelligence

Vagan Terziyan
vagan.terziyan@jyu.fi
Olena Kaikova
olena.kaikova@jyu.fi
Faculty of Information Technology, University of Jyväskylä,
40014 Jyväskylän yliopisto, Finland

Machine learning is a good tool to simulate human cognitive skills as it is about mapping perceived information to various labels or action choices, aiming at optimal behavior policies for a human or an artificial agent operating in the environment. Regarding autonomous systems, objects and situations are perceived by some receptors as divided between sensors. Reactions to the input (e.g., actions) are distributed among the particular capability providers or actuators. Cognitive models can be trained as, for example, neural networks. We suggest training such models for cases of potential disabilities. Disability can be either the absence of one or more cognitive sensors or actuators at different levels of cognitive model. We adapt several neural network architectures to simulate various cognitive disabilities. The idea has been triggered by the "coolability" (enhanced capability) paradox, according to which a person with some disability can be more efficient in using other capabilities. Therefore, an autonomous system (human or artificial) pretrained with simulated disabilities will be more efficient when acting in adversarial conditions. We consider these coolabilities as complementary artificial intelligence and argue on the usefulness if this concept for various applications.

1 Introduction

The world around us is developing faster than ever due to emerging sophisticated technologies. We are also developing ourselves to fit into this changing world. Are we doing this fast enough? Are we confident of having all the needed skills and capabilities to explore all the available opportunities for a better life and, concurrently, to do the best in making the world better? How do people with disabilities manage all their challenges? Discussing the Human 2.0 future, Ray Kurzweil asserted that "we are slowly merging with our technology; . . . we are becoming transhuman, with updated abilities, including enhanced intelligence, strength, and awareness" (Sahota, 2018). There is no doubt that we can survive the future challenge only if we change

Neural Computation **34, 255–290** (2022)
https://doi.org/10.1162/neco_a_01449

ourselves radically. However, what about people with disabilities? Are they moving to the same or a similar objective in the same way and with the same efficiency as everybody else? If the world will change faster than we will, will all of us feel disabled?

A disability is any condition that considerably restricts a person from acting or interacting with the world. The World Health Organization (2001) standardized an ontology of disabilities that examines various sensory, intellectual, and functional disabilities among others. Goodley (2016) argued that disability is a global phenomenon that touches everyone.

Disabilities often come with enhanced abilities. So-called coolabilities are the enhanced abilities and strengths that co-occur with disabling conditions (Grundwag, Nordfors, & Yirmiya, 2017). For example, some blind people reorganize and reassign neural pathways in the visual cortex and get echolocation as a coolability (the capability of perceiving and acting unimaginably to the healthy people). Everyone has strengths and weaknesses. Instead of labeling people as disabled according to what they cannot do, it would be fairer to name people as "coolabled" regarding their strengths. Nordfors et al. (2018) argued that the coolability concept can be useful in any context that aims to increase people's values, greatly impacting the labor market. After elaborating further on this topic, Nordfors, Grundwag, and Ferose (2019) discovered that people with coolabilities could spearhead the new market for tailored jobs because they are a particularly underutilized resource.

If "disability" is a global phenomenon, then why is this concept applied mostly (if not only) to humans? What else can be disabled that we should care about? Currently, we believe in the power of emergent collaborative intelligence as a practice of people and autonomous artificial intelligence (AI) agents working and thinking together. According to Wilson and Daugherty (2018), through collaborative intelligence, humans and AI actively enhance each other's complementary strengths: they write about "the leadership, teamwork, creativity, and social skills of the former, and the speed, scalability, and quantitative capabilities of the latter." Therefore, we believe that one must apply the concepts of disability (thus coolability) to artificial, autonomous, smart, self-aware entities (e.g., robots, intelligent agents) in the same (or similar) way we apply these concepts to humans.

It would be difficult (if possible) to hard-code the complex AI capabilities; they must be trained. Machine learning (ML) addresses the learning methods (e.g., supervised, unsupervised, semisupervised, reinforcement, adversarial) and appropriate training processes aiming to provide necessary learning environments and learning content for AI algorithms (models) to train their intellectual capabilities as learning outcomes.

This letter is based on the following major assumption: if the evolution of a person with a disability may induce the appearance of some strong extra-capabilities and skills (coolabilities), then we can assume that intentionally disabled AI algorithms (e.g., ML models) may learn some coolabilities with

potentially higher performance than the same algorithms may learn under normal conditions. This means that artificial disabilities applied during the training process may result in stronger capabilities (coolabilities) of a target algorithm after training. Another assumption would be that if we can observe (learn) the cognitive activity of a person with a disability and notice some gaps (weaknesses) comparably to a target activity, we can automatically train the autonomous AI compensation (autonomous coolability) as a complementary intelligence or artificial personal cognitive assistant for this person. We call appropriate training approaches, methods, and algorithms "complementary artificial intelligence" (CAI).

In this letter, we address the following research questions:

- What kinds of disabilities can be applied to an AI/ML model (particularly for deep neural networks)?
- How can we artificially embed a disability (one or several) into the ML process of a deep neural network and control it until a coolability is trained?
- Is it so that the model (coolability) trained in disabled conditions will have some advantages compared to the models (capabilities) trained in normal conditions?

The rest of the letter is organized as follows. In section 2, we report on related work. In section 3, we describe our approach and methods. In section 4, we present the disability concept regarding neural networks and illustrate various kinds of disabilities and corresponding potential coolabilities with different neural network architectures, including feedforward (shallow and deep) and recurrent and convolutional neural networks. In section 5, we present sustainable neural network architectures, which can be pretrained to get coolabilities ready before the disability happens. We provide some discussion in section 6 and conclude in section 7.

2 Related Work

Various types of disability simulations are applied within both human and machine learning processes. In the human world, disability simulation is often used to modify social attitudes regarding people with disabilities and is provided to healthy people as a teaching about disabilities. Trainees are supposed to perform their everyday activities with a temporary disability, such as a blindfold or earplug (sensory disability), wheelchair (functional disability), or several disabilities simultaneously (Nario-Redmond, Gospodinov, & Cobb, 2017). Silverman (2015) described a sensory disability simulation experiment where students can master braille reading, cooking, and cane travel under blindfold. The results showed that blindfolded experiences lead to skill mastery and confidence; however, to achieve these positive goals, the blindfolded experiences must be carefully guided. Silverman et al. (2018) also reported on functional disability simulation

experiments where students learned to perform simple tasks while simulating paraplegia (the inability to voluntarily move the lower parts of the body) and hemiplegia (lack of control in left or right sides of the body). Such impairment simulations allow students (after some adaptation and training) to experience success in completing activities of daily living with impairments.

Similar kinds of disability simulations have proven to be helpful for athlete training in sports. Efficiency in sports is primarily determined by various neuromuscular factors and the underlying proprioceptive mechanisms. Integrating information from all the senses with the proprioceptive information enables athletes to coordinate the sensor-actuator processes and execute a given movement most appropriately. According to Verkhoshansky and Siff (2009), one possible way of improving proprioceptive efficiency is to block some inputs from the sensory systems, such as the eyes. For example, a blindfolded powerlifting athlete better feels his or her own errors and remembers body position, joint angles, the degree of muscular tension, and the movement patterns during training, and reproduces them more efficiently with open eyes. One interesting recent example was reported in Rice (2020) on the blindfolded training of the Barcelona football team. The new coach, Quique Setien, believes that such training amplifies the senses of the players.

Barnes (2016) noticed that the weaknesses associated with a disability are compensated by special enhanced abilities that ordinary people lack. Grundwag et al. (2017) named these special abilities *coolabilities* and argued that they may enhance the value of disabled people in certain sectors of the labor market. If the coolabilities may be due to an adaptation of disabled people to the natural constraints they face, coolability for all others may appear only as a result of special simulated disability training like the training of athletes.

To find some analogy between human training with simulated disability and simulated disability applied to machine learning algorithms, assume that we used some training content as an input to make an intelligent entity (human or artificial) perform a certain (cognitive) activity after learning. The key assessment criterion used in machine learning to evaluate the resulting skill is *generalization*, a term used to describe an ability to react to new (unseen during training) inputs. The usual goal of machine learning is to train a model that matches the data. Some of the available data are used to train the model (i.e., making the model fit the training data, and others are used to assess (test) the model (i.e., checking how well the trained model fits the testing data or generalizes). Exact fits of a model to training data are not necessarily a guarantee that the model will generalize well. If the model works well on the training data set but fails to generalize (i.e., does not perform well with the test samples), we say it is overfitting (Hawkins, 2004). The usual reason for overfitting is that the training data often contain some noise; thus, much of the overfitting model is actually modeling

the noise (Bartlett & Holloway, 2019). Therefore, the desires to train faster, reduce overfitting, and make better predictions are currently driving machine learning (especially deep learning) research.

One of the popular techniques to combat overfitting and thus improve generalization in deep neural networks is known as *dropout regularization*. This technique reminds us of some objectives of our letter because it involves embedding deliberate disabilities into the trained model—for example, by randomly dropping out some nodes (neurons) of the network to reduce the likelihood of overfitting. Multiple dropouts applied to the same model will result in a set of partially disabled models, which is proven to better generalize as an ensemble than the original complete model alone (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

Another regularization technique, which involves the use of intentionally disabled neural network models during training, is known as *pruning* (Thodberg, 1991). Instead of removing neurons, pruning involves dropping out (or cutting) some connections between the neurons to induce sparsity in connection matrices of the network (i.e., reducing the number of nonzero parameters in the model). Many weights in a neural network, observable as unimportant during the training process, can be removed (pruned) from the network with little to no consequence. Such a way of reducing the number of parameters in a network enables reasonable execution times of models. Zhu and Gupta (2017) explored the efficacy of pruning for model compression and examined the performance of neural networks as a function of sparsity. In their experiments, they found that halving the number of parameters reduced accuracy by only 0.1%. Therefore, pruning is one efficient way to sparsify neural networks (Alford, Robinett, Milechin, & Kepner, 2018) and discover compact and efficient topologies of the decision models. More on pruning as an instrument for sparsification is presented in section 6.

A recent idea (Li, Fan, Pan, Xi, & Zhang, 2020) is related to learnable auto-pruning processes (instead of handcrafted pruning rules) of deep neural networks, which is inspired by recent automatic machine learning—the AutoML initiative (Truong et al., 2019; He, Zhao & Chu, 2021). The approach aims to derive the optimal channel numbers and weights for each layer using a dynamic masking process to describe the corresponding channel evolution. In addition, a new cost function has been introduced that can control the balance of the accuracy and pruning ratio in the entire procedure. This idea remains a topic for further exploration and testing.

Zero-shot learning (Wang, Zheng, Yu, & Miao, 2019) is another type of artificial restriction (temporal disability) intentionally applied to a machine learning algorithm during the training phase. This disability assumes that the learner cannot observe all the class labels during the supervised learning process; however, when tested on instances of unseen classes, the learner needs to infer a category for these instances, such as taking the nearest known class (embedded in a continuous space) as a predicted one (Frome

et al., 2013). This approach enables domain adaptation or transfer learning (Long, Zhu, Wang, & Jordan, 2017), where the models trained on the data related to some object or specific conditions are applied (tested) to some other object or conditions.

The disability of machine learning techniques can also be modeled by artificially creating voids in memory (forgotten, ignorant, or confusion zones) used for training. This learning process will study the boundaries for such gaps and can learn to make decisions based not only on available data but also on knowledge about these ignorance zones. This means that good self-awareness of one's own disability is a feature with added value in making decisions. Terziyan and Nikulin (2021) studied such disabilities as "gray" zones within training data and suggested generating adversarial examples located deep within the largest voids (ignorance or confusion zones of the decision space) to facilitate learning.

Disability is often associated with vulnerability (Gill, 2006). According to Gill (2006), vulnerability is not necessarily a direct consequence of some disability and can be socially constructed (for a disabled person and sometimes even for a healthy one) in unfriendly cognitive environments. Vulnerability and disability also restrict one's cognitive capabilities, thereby requiring training (among other treatments) to develop self-protection skills. Machine learning algorithms are vulnerable to various kinds of adversarial attacks (Ren, Zheng, Qin, & Liu, 2020) and deep fakes (Kietzmann, Lee, McCarthy, & Kietzmann, 2020). Adversarial training has been used as a tool for proactively protecting AI systems against various vulnerabilities and sophisticated attacks. The basic schema of such training includes the simulation of an adversarial environment, which constantly generates adversarial samples for the trainee to address. Classical architecture of this kind is a generative adversarial network (GAN; Goodfellow et al., 2014). Adversarial training has a similarity to disability simulation learning: both are using artificially challenging conditions for training—either an artificial adversarial environment or an artificial disability. In both cases, training leads to pretrained cognitive activity skills in challenging conditions that may happen in the future.

The stress resilience training used for military personnel prior to deployment is reported in Rizzo et al. (2013). Such training aims to create a set of combat simulations so that trainees face challenging virtual combat contexts to develop stress resilience. Similar kinds of resilience training apply to artificial systems. Consider the generic schema of a smart cyberphysical system presented in Figure 1. In normal conditions (see Figure 1a), the system observes the environment via several channels (sensors) and contributes to the environment via several channels (actuators). The decision model drives the behavior of the system—that is, the particular use of actuators given objectives and the sequence of sensory input. Assume that the decision model is a neural network trained to control the system's behavior. If the data used for training have been collected by assuming that all

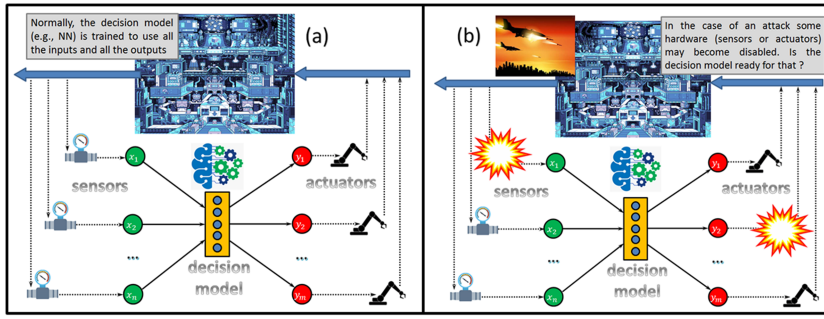

Figure 1: Smart cyberphysical system operating in (a) normal conditions (all healthy sensors and actuators) and (b) conditions with enforced sensory or functional disabilities (due to adversarial attack), or both.

the sensors and the actuators are always available, then the model will also depend on this assumption during execution.

Assume that the system has been attacked, and some of its sensors or actuators become disabled (see Figure 1b). Would the decision model make correct decisions in real time in new conditions with disability (i.e., without having all the input information channels when deciding or without having all the output channels for transforming the decision into behavior)? Certainly not. If the model has not been pretrained for potential adversarial situations, it will not be sustainable or resilient in the face of attacks. Therefore, pretraining even healthy systems to work in adversarial conditions with simulated disabilities will make such systems resilient and able to tolerate enforced structural changes in real time.

3 Approach and Methods

We first formulate our beliefs on the connection between autonomous AI and disability. For being autonomous, one has to be self-aware and self-managed (including awareness of and control over one's own capabilities and disabilities) and behave consciously, an intended feature for the strong AI. Artificial consciousness is still a subject of debate within the heterogeneous academic community (Hildt, 2019). The reason is uncertainty of the closely related "strong AI" concept, as it is difficult to guess what it could be and what it will be. We consider such essential dimensions of artificial consciousness as self-awareness (Chatila et al., 2018) and self-monitoring (Dehaene, Lau, & Kouider, 2017). Both are related to the awareness of an autonomous AI entity (agent, robot, or system) of one's own capabilities and actions and the activity on gaining such awareness. However, we believe that the missing part here is the awareness of one's own (existing or potential) disabilities and the activity in getting such awareness, as well as

planning behavior to train compensatory skills addressing real or potential disabilities.

To avoid complex philosophical debates on artificial consciousness summarized in Hildt (2019), we suggest a softer view of the consciousness concept. We believe that consciousness (for humans or AI) is the ability to understand the boundary between everything within observable "me" (internal environment) and the observable "rest of the world" (external environment) and the activity of keeping a balance between these two. The balance in the long run would mean, for a human, the possibility of completing the personal mission statement and, for an autonomous AI agent, the possibility of completing its design objectives. The assumption is that an autonomous AI entity would consciously use its sensors to get data about the world and about itself, process the data, and decide on the use of actuators aiming at changing the world or oneself.

According to Terziyan (2007), an autonomous AI agent is an entity that can continuously observe and keep the balance between its internal and external environments in such a way that in the case of an imbalance, the agent can:

- Change the external environment (using actuators on the outside world) toward intended balance or/and
- Change the internal environment (using actuators on oneself) toward intended balance or/and
- Move to another place within the external environment where the intended balance occurs without any changes or/and
- Communicate with one or more other agents and create a community, which the integrated internal environment will compensate the external one or/and
- Configure sensors (by updating or deleting) to change the view of the external environment so that the new one will be in balance with the internal environment.

Following such an understanding of an autonomous agent, we can assume that a disability would be a restriction on one of these balance search behavior options. We can define the following main (cognitive) disability categories accordingly:

- *Sensory disability*. The agent cannot get (or to some extent is limited in getting) enough data about itself and the outer environment due to limitations of its internal and external sensors. The agent therefore lacks complete data for making the behavioral decision.
- *Limited self-awareness*. The agent cannot evaluate (or to some extent is limited in evaluating) the balance, that is, in estimating the mismatch (unbalance) between the current states of its external and internal environments. The agent does not understand a need for action.

- *Limited decision-making ability*. The agent cannot decide (or is to some extent limited in deciding) how to act to recover the balance. The agent cannot make a choice on how to act.
- *Functional disability*. The agent cannot act (or to some extent is limited in its behavior actions to change, move, communicate, and so on) toward balance recovery. The agent cannot perform the best-chosen action.

In the rest of the letter, we explore simulating these disabilities by special manipulations with (deep) neural networks, aiming to show that the models pretrained within various disability conditions develop some extra "skills" as coolabilities and will be better prepared for various adversarial scenarios during testing.

We will be looking for AI-driven models and algorithms that explain the coolabilities paradox and enable trained, autonomous artificial cognitive enhancement (also known as coolabilities) for disabled people (and all kinds of artificial disabled cognitive systems).

4 Simulating Disabilities within Neural Networks

In this section, we suggest a special pruning technique for (deep) neural networks and related disability simulation architectures (applied to input, output, and hidden layers), aiming not only at overfitting but also making the network resilient against potential attacks and acquired disabilities.

4.1 A Healthy Neuron versus a Neuron with a Disability. We begin with the atomic component of a neural network, that is, the neuron itself. We consider a neuron from a hidden layer of a neural network to be healthy if it is connected to all the neurons from the previous layer (which is either input or hidden layer) and from the next layer (which is either output or hidden layer). Such a neuron is assumed to be fully capable both sensor-wise and function-wise. The popular term *fully connected layers* used in deep neural network architectures means here that such layers contain only healthy neurons.

For simplicity, we start with just one hidden layer network: a healthy neuron (see Figure 2a) is connected to all the neurons from the input layer of the network and the output layer. We assume our neuron has all the weights for inputs and outputs as trainable ones (i.e., not locked or frozen). A healthy neuron transfers the weighted sum of all the inputs through an appropriate activation function onto all the neurons of the next layer (see Figure 2b). During network training (backward propagation or backpropagation), a healthy neuron gets the weighted feedback (appropriate error values) from all the neurons from the next layer (see Figure 2c).

Let us simulate some sensory or functional disability regarding a neuron. For simplicity, we assume that a neuron (if not a healthy one) may have just

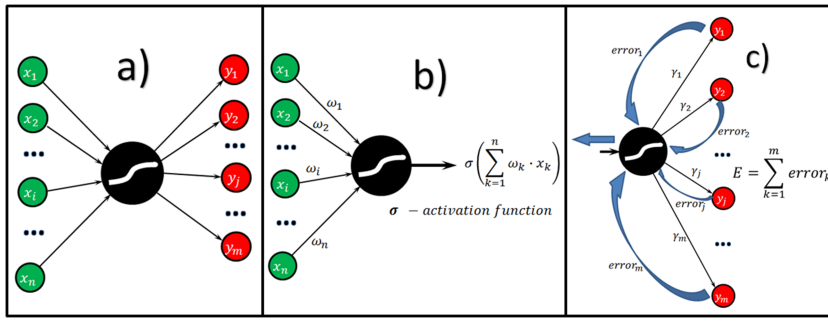

Figure 2: A healthy neuron. (a) The structure of a fully connected and fully capable neuron. (b) Forward propagation through a healthy neuron. (c) Backward propagation through healthy neuron.

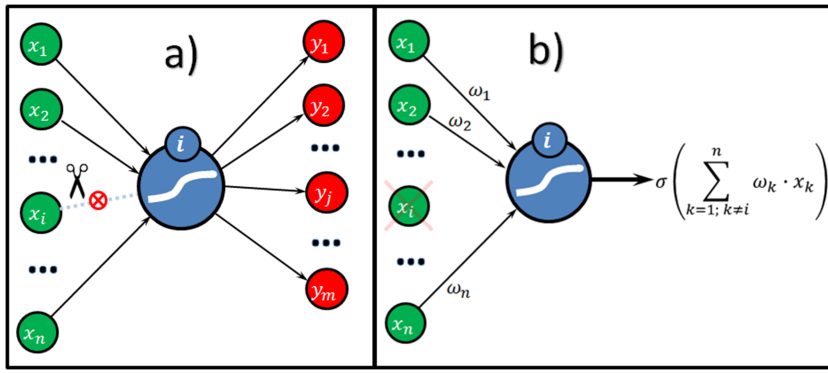

Figure 3: Neuron with a sensory disability. (a) Structure of a neuron with a disabled ith input channel. (b) Forward propagation through the sensory disabled neuron.

one (either sensory or functional) disability. We introduce a neuron with one sensory disability (see Figure 3). We consider a neuron from a hidden layer to have a sensory disability regarding the ith input channel if it is connected to all the neurons from the previous one in our simple case, except just the ith one. This would mean that the corresponding link is cut (also known as pruning) or the corresponding weight is locked (frozen) with zero value. Otherwise, we consider such a neuron as fully capable: it is connected to all other (except the ith one) neurons from the input layer of the network and to all the neurons from the output layer (see Figure 3a). Such a neuron is unaware of x_i, so it outputs a value that does not depend on the unseen input x_i because the weight ω_i is locked to zero (see Figure 3b) and will not

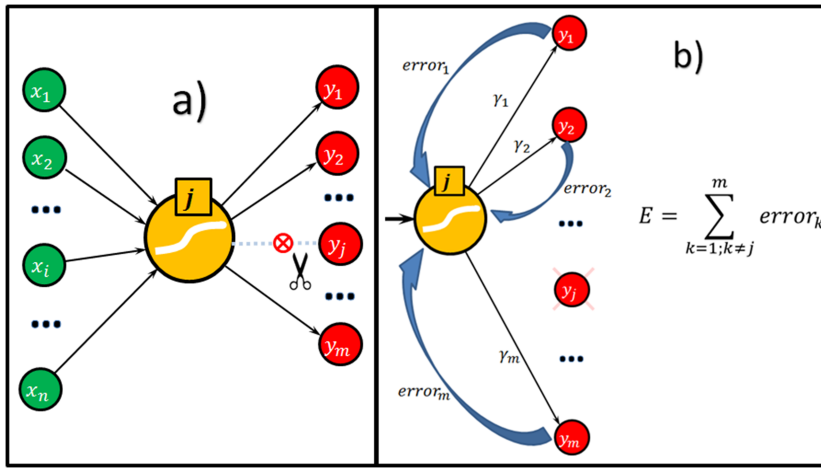

Figure 4: Neuron with a functional disability. (a) Structure of neuron with a disabled jth output channel. (b) Backward propagation through the functionally disabled neuron.

change during backpropagation. Even if we assume that the x_i input is the most informative one for the task, a disabled neuron still learns how to use other inputs efficiently (it recalculates the available input weights during the backpropagation training process) to optimize its own performance.

We introduce a neuron with one functional disability (see Figure 4). We consider a neuron from a hidden layer to have a functional disability regarding the jth output channel if it is connected to all the neurons from the next layer (the output layer in our simple case) except just the jth one. Otherwise we consider such a neuron as fully capable: it is connected to all the neurons from the input layer of the network and to all the neurons from the output layer (except the jth one; see Figure 4a). Such a neuron is forbidden (or is incapable) to use (choose) y_j for a class or action label (due to the corresponding weight γ_j is locked to zero); thus, the neuron never gets feedback (error value) through this output channel during backpropagation (see Figure 4b). Even if we assume that the y_j output is the best option for our neuron to choose as a class or action, a disabled neuron learns how to use other available options efficiently (it recalculates the available output weights during the backpropagation training process) to optimize its own performance.

Therefore, whereas a healthy neuron allows refinement data to flow through it in both directions (forward and back), neurons with disabilities (either sensory or functional) work as a kind of opposite to each other's digital "semiconductors."

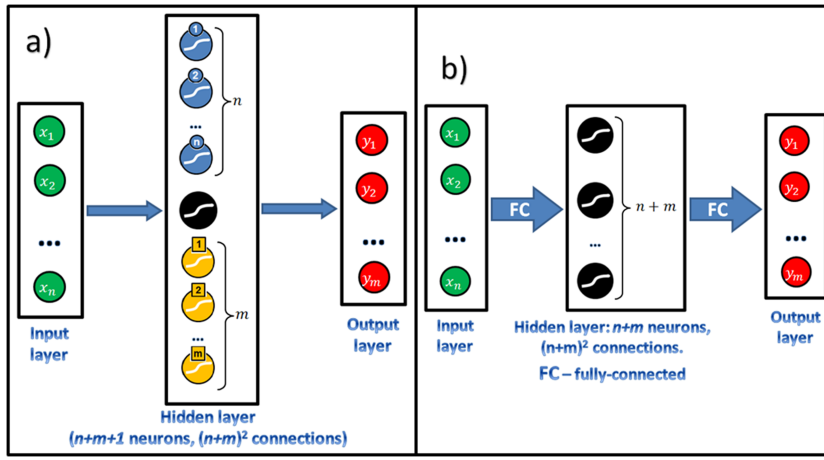

Figure 5: Disability-driven versus healthy network, (a) Architecture of a simple feedforward neural network with one hidden layer, comprising one healthy neuron (center) and the neurons with all kinds of different sensory and functional disabilities, (b) Similar architecture with all the healthy neurons and the same number of connections as the previous one.

4.2 Shallow Neural Network with Partially Disabled Neurons. Now consider the simple architecture of a shallow feedforward neural network (see Figure 5a) that has one hidden layer, and this layer comprises different neurons regarding a disability and one healthy neuron. Such a layer comprises $n + m + 1$ neurons: n neurons with sensory disability for each n input, m neurons with functional disability for each m output, and the healthy neuron. Such a network has $(n + m)^2$ connections, which are $n \cdot (n + m + 1) - n = n \cdot (n + m)$ connections between the input and hidden layers plus $(n + m + 1) \cdot m - m = m \cdot (n + m)$ connections between hidden and the output layers. This network simulates a kind of collective intelligence team of the heterogeneous decision makers (neurons with all kinds of disabilities, which can be considered as coolabilities after training) in such a way that the disability-constrained and therefore coolability-enhanced opinion of each team member would contribute to the group decision together with the healthy team member.

Regarding the number of hyperparameters (neurons, connections), the network in Figure 5a would be almost equivalent to the ordinary network (see Figure 5b) with all healthy $n + m$ neurons at the hidden layer and thus $(n + m)^2$ connections. However, with the same number of connections, the decision performance of a team of $n + m$ disabled and one healthy neuron is expected to outperform that of $n + m$ healthy neurons due to the coolability paradox.

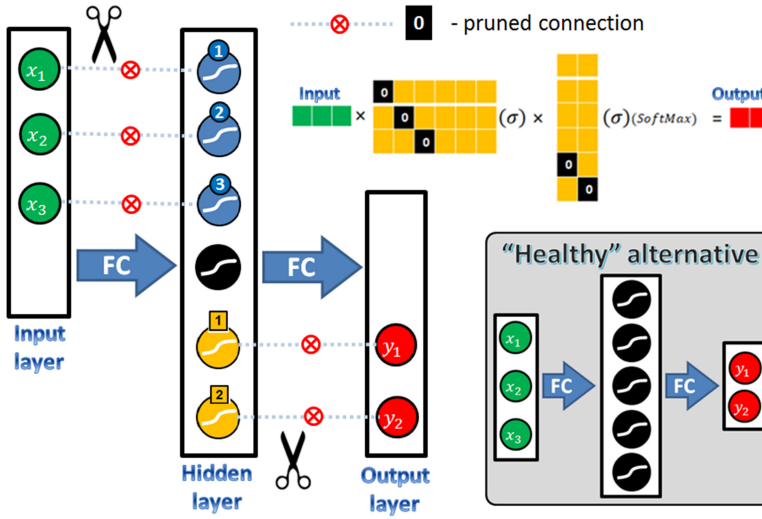

Figure 6: Example of a disability-driven network with three inputs and two outputs and therefore $3 + 2 = 5$ pruned connections. The computation schema for the forward propagation is also shown (the pruned connections become the "locked" zeros in the appropriate matrices). On the right, a healthy alternative is shown, which contains five healthy neurons at the hidden layer. Both networks have 25 connections.

Consider the example of a particular disability-driven neural network with three inputs and two outputs, shown in Figure 6, which illustrates what kind of pruning has been done to construct such network. Three cuts on the left-hand side of the hidden layer add different sensory disabilities to three corresponding neurons from the hidden layer, and two cuts on the right-hand side of the hidden layer add different functional disabilities to two corresponding neurons from the hidden layer. Otherwise the hidden layer is fully connected. Weights for the removed connections in the corresponding matrices are replaced as locked (unchangeable or frozen) zeros. One can also see the healthy network as an alternative, which has five healthy neurons at the hidden layer and 25 connections altogether (the disability-driven alternative has the same number). Emphatically, the 3D data, which train or test such a network, are seen as three different 2D projections by corresponding neurons with sensory disability. In addition, two neurons with functional disabilities do not see corresponding class labels and can vote only for the remaining class. Both of these specifics add information and advantage to the disability-driven network compared to its healthy counterpart, making the former more resilient to sparse, noisy, or adversarial data.

When necessary, one may use more than one healthy neuron at the hidden layer for separating data bounded by complex manifolds. In any case, the architecture in Figure 5a with k healthy neurons ($n + m + k$ neurons together with the disabled ones) will have the same number of connections as the completely healthy architecture shown in Figure 5b with $n + m + k - 1$ healthy neurons. This is possible because pruning $n + m$ connections makes the same impact on the total number of connections as removing one healthy neuron together with its n inputs and m outputs. One can also use more than one disabled neuron of each type at the hidden layer. If each type of disability appears d times at the hidden layer, then altogether, the layer will have $d \cdot (n + m) + k$ neurons, and its completely healthy equivalent (with the same amount of connections) will have $d \cdot (n + m - 1) + k$ neurons.

Our experiments with specially generated synthetic data show that the pruning option (i.e., the hidden layer with embedded disability) gives a lower (interval varies on a case-to-case basis) generalization error than the completely healthy hidden layer with the same number of connections. For example, if some complex case of applying an ordinary shallow network ($n = 3$ inputs, $m = 2$ outputs) needs 11 healthy hidden-layer neurons (i.e., 55 connections altogether) to get the desired accuracy during testing, then one may expect even better accuracy if applying the hidden layer with disability ($d = 2$, $k = 3$) having 13 neurons (6 with sensory disability, 4 with functional disability, and 3 healthy ones), thereby yielding the same number of 55 connections.

4.3 Deep Neural Networks with Partially Disabled Neurons. Can we construct a neural network with several hidden layers (as in Figure 5a) of disabled neurons? To do so requires considering the following evident constraint: if the ith neuron of hidden layer r of a network is a neuron with functional disability (with disabled jth output), then the jth neuron from the next ($r+1$) hidden layer must be a neuron with sensory disability (with disabled ith input). Such a couple of neurons will be called "divorced" because the link between them is absent (pruned; see Figure 7).

Considering this constraint, for the network with n inputs and m outputs, at each odd layer, we would have n neurons with sensory disability and m neurons with functional disability, while at each even layer, we would have m neurons with sensory disability and m neurons with functional disability. Therefore, a consistent deep architecture (with complete teams of disabled neurons) will always contain an odd number of hidden layers. Figure 8 shows that we suggest adding healthy neurons only to each even hidden layer (i.e., the second one in this network). This makes the architecture easily comparable to a fully connected (completely healthy) network with the same number of connections. Such modification of the network (n inputs, m outputs, l is an odd number of hidden layers, $n + m$ disabled neurons in each hidden layer plus one healthy neuron in the even hidden

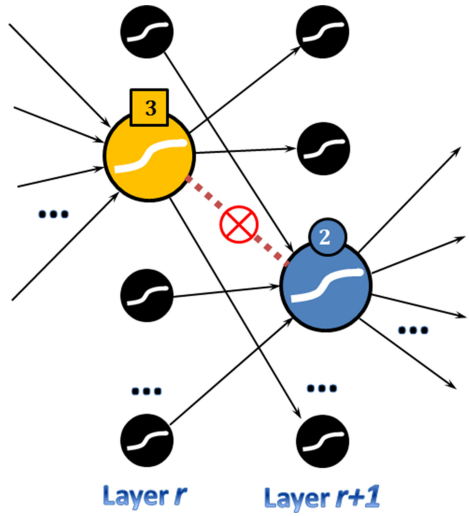

Figure 7: Example of a "divorced couple" of disabled neurons. The pruned link between the hidden layers makes the neuron on the left-hand side have a functional disability and the corresponding neuron on the right-hand side have a sensory disability.

layer) has altogether $l \cdot (n+m) + \frac{l-1}{2}$ neurons (16 in Figure 8) at hidden layers and exactly $l \cdot (n+m)^2$ connections (75 in Figure 8). This configuration can be compared with a similar, completely healthy deep (rectangle form) network, which has n inputs, m outputs, l (an odd number of hidden layers), $n+m$ healthy neurons in each hidden layer, and such architecture will have $l \cdot (n+m)$ neurons (i.e., $\frac{l-1}{2}$ neurons less than the disabled architecture) and the same number $l \cdot (n+m)^2$ of connections. This means that the correct rival for the network from Figure 8 would be the healthy network with 3 hidden layers and 15 neurons altogether at hidden layers (just one fewer neuron). Experimenting with these two architectures against each other on synthetic data shows that the one with embedded disability on average generalizes better than the healthy one, proving the coolability assumption. For this case, the team of 15 neurons with disabilities, together with one healthy neuron (75 connections altogether), appeared to outperform the team of 15 healthy neurons (also with 75 connections).

Interestingly, these two competing architectures have a common parent architecture on top of which two different regularization methods could be applied. Consider the healthy deep neural network in Figure 9, which has three hidden layers with 16 neurons: $m+n$ neurons (5) at odd (first and third) layers and $m+n+1$ (6) neurons at even (second) layer, and 85 connections altogether. If we apply special pruning as a regularization

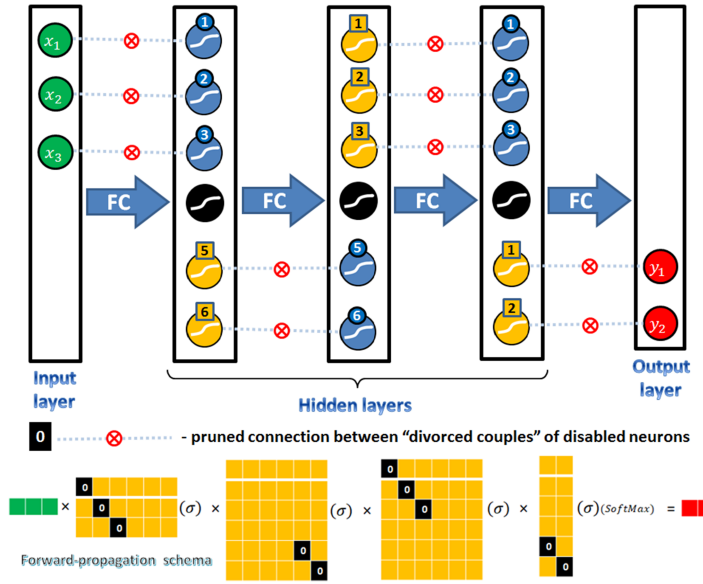

Figure 8: Example of a deep (three inputs, two outputs, three hidden layers with 16 neurons and 75 connections) neural network with five disabled neurons at each layer plus the healthy one only at even (second layer in his case) layers. Notice the difference and symmetry of odd and even layers. See also the computational schema for the forward propagation.

technique, we cut $2 \cdot (m + n)$, that is, 10 connections and get a deep network with embedded disabilities (the left-hand side of Figure 9). If special pruning was applied as a regularization technique and removed one healthy neuron from each even (second in this case) layer, this would mean giving up $2 \cdot (m + n)$, that is, 10 connections, and getting a deep healthy rectangle network (the right-hand side in Figure 9).

4.4 Deep Neural Networks with Hybrid Layers. In previous architectures, we allowed deep disability—disabled neurons that may appear at every layer of a deep neural network. In this section, we simulate a more natural- looking disability: both sensory and functional disabilities may appear only at the cognitive interface layers of a neural network between the "reality" and the "decision model," assuming that internal hidden layers of the decision model will be completely healthy. This means that in simulating appropriate disability, we will apply pruning between the input layer and the first hidden layer (getting neurons with sensory disability only) and between the last hidden layer and the output layer (getting neurons with functional disability only) and having several healthy hidden

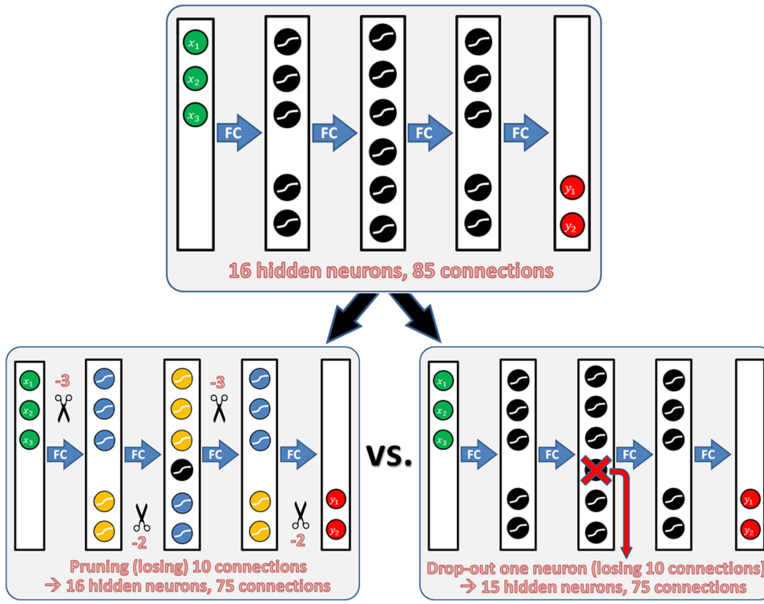

Figure 9: Pruning versus dropout nature of two competing architectures.

layers in between. This architecture is a kind of hybrid that has one layer with sensory disability, one layer with functional disability, and at least one healthy layer. We suggest the following generic rules to construct hybrid networks with n inputs and m outputs and their healthy rivals for comparison. Both networks must have an odd number of hidden layers ($l \geq 3$). Each odd hidden layer must have $n + m$ neurons in the hybrid network and $n + m - 1$ neurons in the healthy rival network. Each even hidden layer must have $n + m - 1$ neurons in the hybrid network and $n + m$ neurons in the healthy rival network. Therefore, a hybrid network always has one fewer neuron than its healthy rival. At the first hidden layer of the hybrid network, n neurons from $n + m$ are the neurons with sensory disability, and at the last hidden layer of the hybrid network, m neurons from $n + m$ are the neurons with functional disability. Constructed in such a way, both rival networks will have the same number of connections (trainable weights): $l \cdot (n + m) \cdot (n + m - 1)$.

Consider the example of hybrid architecture in Figure 10. It has three inputs, two outputs, three hidden layers, 14 neurons (3 with sensory disability, 2 with functional disability, and 9 healthy ones), and 60 connections (see Figure 10a). The first hidden layer contains all different neurons with sensory disabilities, and the last hidden layer contains all different neurons with functional disability. The architecture does not allow sensory and

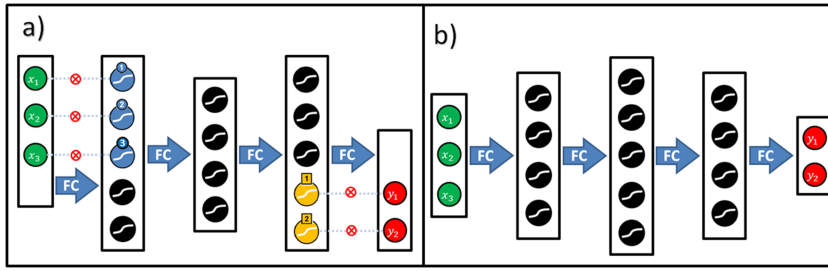

Figure 10: Deep (three layers) disability-driven hybrid versus a healthy network. (a) Architecture of a network with three hybrid hidden layers (it has 14 neurons at hidden layers and 60 connections). Notice that the first layer contains neurons only with sensory disabilities plus healthy ones, the second layer has only healthy neurons, and the third layer contains neurons only with functional disabilities plus healthy one). (b) Rival architecture of completely healthy network with three hidden layers (13 neurons and 60 connections).

functional disabilities to appear at the same layer. There is also a hidden layer in between, and this (even) layer is one neuron shorter than the neighboring (odd) layers. This architecture is comparable to the completely healthy network (see Figure 10b), which has three inputs, two outputs, three hidden layers, 13 neurons (all healthy), and 60 connections. Notice that this healthy network has odd hidden layers shorter than the even ones (opposite the rival network with disabilities). We conducted several experiments on top of specially generated synthetic data sets, which show that (on average) hybrid architectures with embedded disabilities slightly outperform their healthy counterparts.

Embedded disability (special pruning) in different architectures from sections 4.2 to 4.4 can be combined with other regularization methods when necessary to get the intended generalization performance. Therefore, in our experiments, we made comparisons: (network with embedded disability + other regularization) versus (healthy (rival) network + other regularization). We saw that an embedded disability gives some added value (different for different data sets) to the neural networks' performance, which confirms the coolability assumption.

4.5 Recurrent and Convolutional Neural Networks with Disabilities. Previously, we considered fully connected feedforward neural networks where each healthy neuron can take all the inputs from the neurons of the previous layer and send the computed outcome to all the neurons of the next layer. Restrictions for doing that we called disabilities (sensory and functional respectively). Other more powerful networks, which are

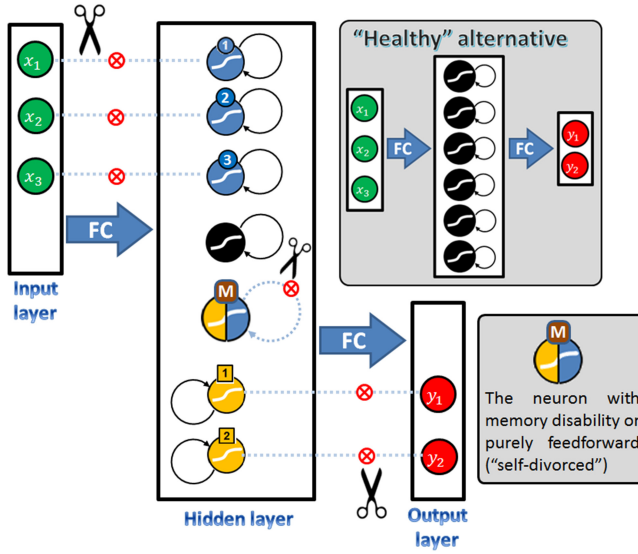

Figure 11: A shallow disability-driven recurrent neural network with three inputs, two outputs, and one hidden layer constructed from seven recurrent neurons: one completely healthy neuron plus six neurons with all kinds of disabilities (three with sensory disability, two with functional disability, and one with memory disability). The network therefore has $3 + 2 + 1 = 6$ pruned connections. A healthy alternative (rival network) is also shown, which contains six healthy recurrent neurons at the hidden layer. Both networks have 36 connections.

recurrent neural networks (RNNs), also enable loops in their architecture; thus, one extra skill for the neurons is the ability to memorize the outcomes at current iteration and use the stored values as additional inputs for the computation at the next iteration (see a critical review on RNNs in Lipton, Berkowitz, & Elkan, 2015). The absence of such a skill can be considered one more disability type; we call it a *memory disability*. This would mean that a healthy neuron in the feedforward neural network (see Figure 2) is a neuron with memory disability in RNNs because it lacks a feedback (memory) loop. Embedding such a disability into RNN architecture means just pruning such a loop for some neurons. We marked such neurons (memory handicaps) with the letter "M" (see Figure 11). We also used double colors for such neurons due to the double nature of memory disability. While these neurons (like the neurons with sensory disability) cannot get information from one important input channel, which is the neurons' own memory, they (like the neurons with functional disability) cannot propagate their outcomes through one important output channel, which is also

the own memory. Unlike pruning the binary connection, which results in a "divorced couple" of neurons (first with functional and second with sensory disabilities—Figure 7), pruning the loop results in a kind of "self-divorced" neuron with memory disability.

Figure 11 is the RNN version of the architecture previously shown in Figure 6. The hidden layer contains neurons with all kind of disabilities, including the memory disability. A completely healthy rival is also shown.

Deep or stacked architectures of RNNs are also possible, and these can be constructed in many ways (Pascanu, Gulcehre, Cho, & Bengio, 2013). Extending our view of deep neural networks with disabilities (see Figures 8 and 9) to the context of RNNs, we provide generic rules for constructing deep RNN architectures with disabilities.

This architecture has n inputs; m outputs; l (an odd number of hidden layers, with each odd hidden layer containing $n+m+1$ neurons); n neurons with sensory disability plus m neurons with functional disability plus one neuron with memory disability); and $n+m+2$ neurons in each even hidden (same as the odd layer plus one completely healthy neuron). Such a deep recurrent network has $l \cdot (n+m+1) + \frac{l-1}{2}$ neurons at hidden layers and exactly $l \cdot (n+m+1)^2 + (l-1) \cdot (n+m+1)$ connections (including "self" loops). This configuration can be compared to a similar completely healthy deep (rectangle form) recurrent network, which has n inputs, m outputs, l (an odd number of hidden layers), and $n+m+1$ healthy neurons in each hidden layer. It will also have $l \cdot (n+m+1)$ neurons (i.e., $\frac{l-1}{2}$ neurons fewer than the disabled architecture) and the same number $l \cdot (n+m+1)^2 + (l-1) \cdot (n+m+1)$ of connections.

Experimenting with these two architectures on synthetic data shows that the one with embedded disability on average generalizes better than the healthy one, proving the coolability assumption. For this case, the team of 15 neurons with disability, together with one healthy neuron (75 connections altogether), appeared to perform better than the team of 15 healthy neurons (also with 75 connections).

A popular and efficient way to manage memory in RNNs is known as long short-term memory (LSTM; Hochreiter & Schmidhuber, 1997; Greff, Srivastava, Koutník, Steunebrink, & Schmidhuber, 2016). In contrast to traditional RNNs, LSTM networks are stronger due to their ability to remember certain information from the past as long as it influences the current prediction and forgets such information if the influence is weak. This ability is supported by special LSTM memory cells. The original LSTM model comprises a single hidden LSTM cell-driven layer, followed by a standard feedforward output layer. It is possible, however, to manage the long-term dependencies within the multilayered structure (Hihi & Bengio, 1996), which enables deep (or stacked) LSTM networks that have multiple hidden LSTM layers where each layer contains multiple memory cells (Graves, Mohamed, & Hinton, 2013).

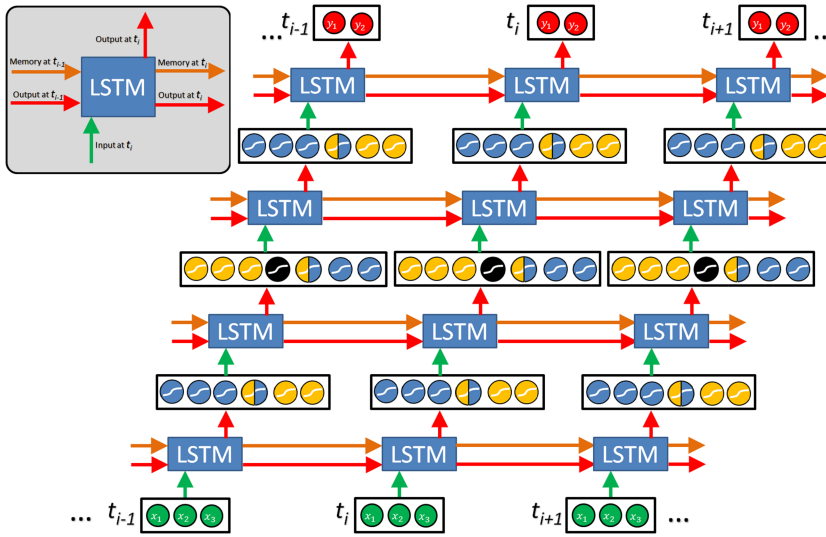

Figure 12: Example of a deep recurrent (LSTM) neural network architecture with embedded disabilities at each hidden layer.

Like our approach to embedding disabilities into deep feedforward neural networks, we do the same for the multilayer LSTM network (see Figure 12), which guarantees us the coolability effect with the intentionally disabled multilayered memory structure.

Consideration of the simulation of memory disabilities complements an excellent study by Brainerd (2021) on deep memory distortions: how the structure of memory could misrepresent the objective structure of the events that humans remember. Four types of such deep distortions have been discussed: overdistribution, super-overdistribution, nonadditivity, and impossible conjunctions.

Due to the current hype on applying deep learning models for image processing and generation, convolutional neural networks (CNNs) and their variations are becoming popular. CNN (LeCun & Bengio, 1998; Rawat & Wang, 2017) can capture locally hidden features by scanning the structural input (e.g., an image) using several special convolution plus pooling layers, then take these features as a new input, and, finally, apply a deep (fully connected) neural network to classify this input. (We will not touch on the convolution plus pooling part of the architecture in this letter, though it is possible and reasonable also.) We embedded the disabilities within the fully connected part of the architecture only, similar to the way we did previously for the deep feedforward neural networks. The generic structure

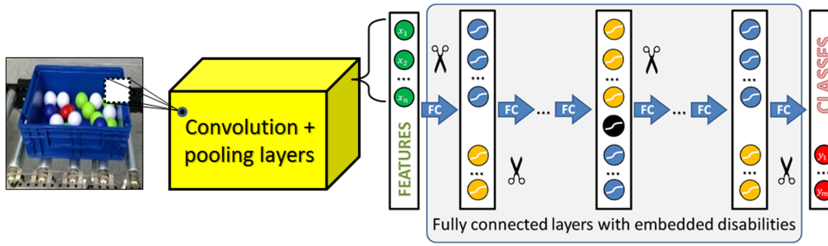

Figure 13: Generic structure of a convolutional neural network architecture with embedded disabilities at fully connected layers.

of CNN architecture is shown in Figure 13. This architecture was tested and compared with the completely healthy, fully connected layers as part of the NATO SPS Cyber-Defence for Intelligent Systems project activity (http://recode.bg/natog5511) in a real laboratory environment, where various kinds of adversarial attacks are generated as images to challenge the AI system (Golovianko, Gryshko, Terziyan, & Tuunanen, 2021). We observed the coolability effect there at the level of 2% to 10% classification precision improvement, depending on the training image set size. We used this effect within the NATO project as one of the features of the digital immune system (Terziyan, Gryshko, & Golovianko, 2021) protecting smart objects and processes of critical and military infrastructure, making them resilient against potential cognitive attacks, such as those described by Terziyan, Golovianko, and Gryshko (2018).

5 Resilient Coolability-Enhanced Neural Network Architectures

In this section, we suggest a special neural network architecture that is deliberately designed as an excessive (multichannel) one but can learn stress resilience in advance to handle consequences of possible attacks (as illustrated in Figure 1) in real time without loss of their classification (prediction) accuracy.

Consider a healthy neural network that has been trained (supervised learning) to address the n-dimensional input $\{x_1, x_2, \ldots, x_n\}$ by deriving the choice among m possible outputs $\{y_1, y_2, \ldots, y_m\}$ (e.g., classes, actions). Assume that at the learning stage, the training set, which comprises labeled training samples in the form $(x_1, x_2, \ldots, x_n \rightarrow y_k)$, has been used. When the trained model starts to operate, it always "expects" that for each decision problem, the complete input will be available (all n input values) and that all m decision options are also allowed. If we deliberately switch off one of the inputs at the operation (testing) stage, especially an important input, the model will drop its decision performance because it has not been

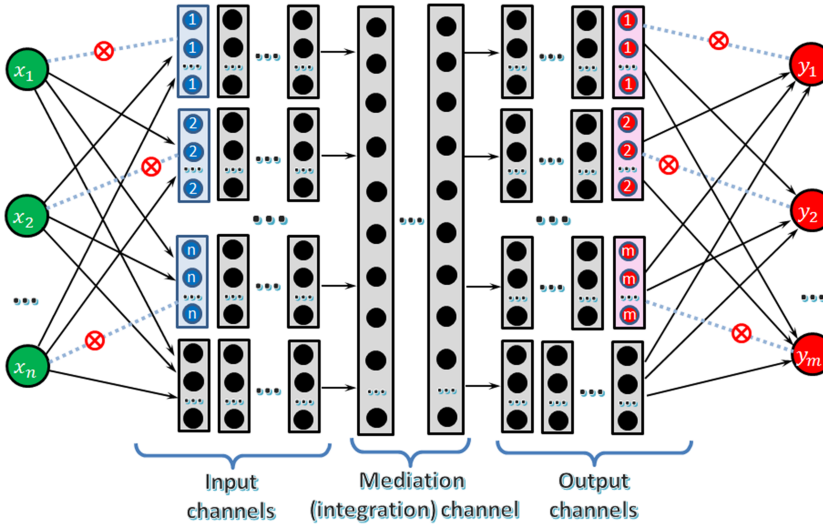

Figure 14: Generic (deep) architecture of a sustainable coolability-enhanced multichannel neural network.

pretrained for this stress. The same will happen if the derived output choice option becomes unexpectedly prohibited. In that situation, the model may not always be capable of finding the second-best choice among the remaining ones. Therefore, we suggest a sustainable multichannel neural network architecture where each channel will be pretrained for a certain possible stress (disability) during the testing stage.

Consider the generic (deep) architecture of such a network in Figure 14. It has a set of $n + 1$ isolated input channels (one for each sensory disability plus the healthy channel), a set of $m + 1$ isolated output channels (one for each functional disability plus the healthy channel), and one integration (mediation) channel of healthy, fully connected hidden layers between these two sets. Each ith input channel is a set of hidden layers started by the obligatory layer comprising $n + m$ neurons with the ith sensory disability, and this layer may be (if necessary) followed by several healthy hidden layers of the same size. Each jth output channel is a set of hidden layers ended by the obligatory layer comprising $n + m$ neurons with the jth functional disability, and this layer may be (if necessary) preceded by several healthy hidden layers of the same size. The (optional!) mediation channel in the middle of the architecture comprises one or more healthier hidden layers (at least $n + m$ neurons each), which are fully connected with each other, with all the input channels and all the output channels.

What makes this architecture in Figure 14 sustainable? Each input channel is a complete network that tries to learn how to get maximum information from the incomplete input (i.e., each channel is lacking one input). Regarding Figure 1, if, during the run-time of the system, the source (e.g., sensor) for the particular input is (e.g., physically) destroyed, the neural network controlling the system can still address the incomplete input due to the pretrained input channel. Each output channel is a complete network that tries to use the incomplete set of available output options as much as possible (i.e., each channel is lacking one output). Regarding Figure 1, if, during the run-time of the system, the enabler (e.g., actuator) for the particular output is destroyed (e.g., physically), the neural network controlling the system can still choose the best actuator among the available ones to address the recommended output due to the pretrained output channel. There is one completely healthy input channel and one completely healthy output channel that together simulate the work of the control system without any disabilities. Keeping each possible pair of input + output channels completely isolated would require $(n+1)\cdot(m+1)$ isolated channels, which makes the architecture too heavy. Therefore, we suggest using a mediation (integration) channel between the input and output channels as a compromise between the efficiency and the sustainability of the neural network architecture.

The simplest (shallow) sustainable architecture is shown in Figure 15. Here, each input and output channel is represented with one layer, and all the input and output layers are fully connected without any mediation layers in between (see Figure 15a). This architecture requires a special order of training. Each input and output channel pair is trained (almost) in isolation from other channels. We use the same training data set to run $(n+1)\cdot(m+1)$ training sessions separately for each pair of channels. Figure 15b shows how the healthy–healthy pair of the architecture is being trained. In that stage, all the weights, which belong to other channels, are locked for training (frozen for change). This would mean that after the training, the network can be used in the future healthy conditions and will be as capable as an ordinary healthy neural network if all the inputs and outputs are available. We trained all other pairs of channels in the same way. For example, a pair of channels (one with disabled ith input and the other with disabled jth output) could be trained (see Figure 15c). Because this pair is isolated from the rest of architecture during training, it is being pretrained to address the absence of the ith input (e.g., destroyed sensor) and the absence of the jth output (e.g., destroyed actuator) in potentially adversarial future exploitation. How will all of these work in the testing phase? The system (e.g., the one from Figure 1) may be healthy at the beginning, and thus all the testing inputs go (forward-propagate) through only the pretrained healthy subarchitecture (i.e., the rest of the links are assumed to be pruned). Then different attacks are simulated, that is, at most one input or one output is assumed as destroyed. For each of these attacks, the architecture

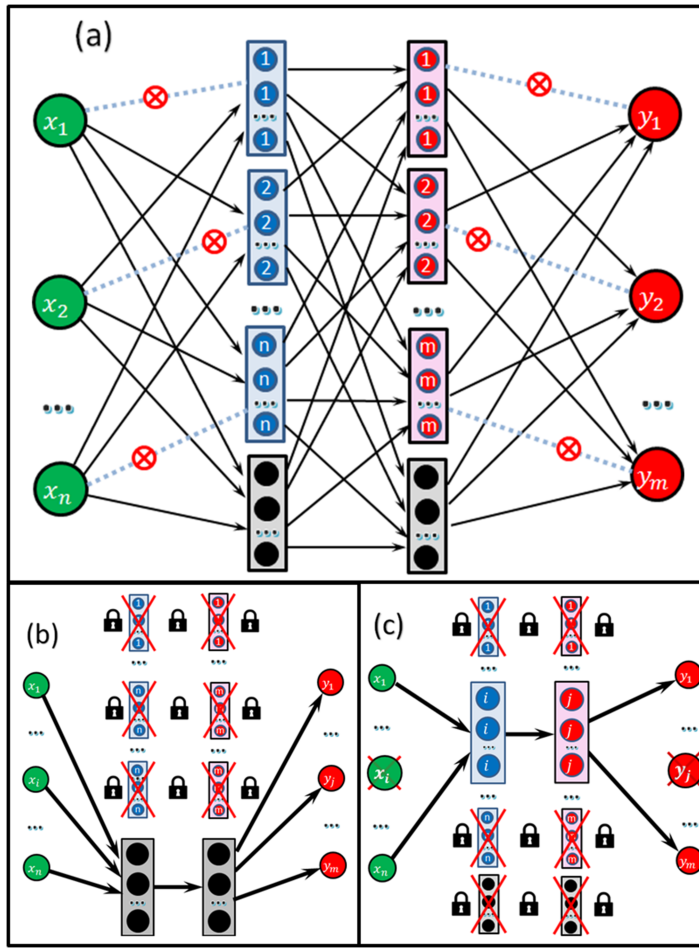

Figure 15: Shallow architecture of a sustainable coolability-enhanced multichannel neural network and its training. (a) The architecture with the "short" disability channels. (b) The stage of training the healthy-to-healthy channel, while other channels (i.e., corresponding weights) are locked for training. (c) The stage of training the *i*th-disabled-to-*j*th-disabled channel (such training stage is applied separately to all pairs of input and output channels), while other channels (i.e., corresponding weights) are locked for training.

unlocks and uses only the appropriate pretrained part of itself (also known as coolability) to address the testing samples, and the rest of the architecture is assumed to be pruned.

Experiments with synthetic data show that in adversarial conditions (with sudden disabilities of inputs or outputs), the testing performance of the pretrained, coolability-driven architecture from Figure 15 or the deep one from Figure 14 significantly (up to 20% in some cases) outperforms that of the ordinary neural network trained in completely healthy conditions. This means that the sustainability of critical infrastructure or various systems acting in adversarial environments can be pretrained.

6 Discussion

The coolability phenomenon has been studied by Nordfors et al. (2018) and Nordfors, Grundwag, and Ferose, (2019), and it is related to the enhanced compensational abilities of people with disabilities. Nordfors et al. (2019) provide an essential set of references showing that neuroscientists have already identified enhanced abilities and compensatory mechanisms for a number of disabling conditions—for example, people with autism spectrum disorder, according to Nordfors et al. (2019), who referred to several sources with the evidence, "usually have such disabilities as difficulties in social interaction, reduced self-awareness, too narrow focus and high sensitivity to sensory stimulation, resistance to change, repetitive behavior, etc.; however, as a kind of compensation, they . . . (b) . . . usually have the enhanced abilities (coolabilities, which accompany the disabilities above) such as extraordinary observation skills, intense focus with attention to detail, expansive long-term memory, affinity to analyzing complex patterns, creativity and exceptional talents in specific areas of interest, success at repetitive tasks, strong systemizing skills, etc.; and, therefore, such people . . . (c) . . . can be potentially unique and great computer programmers (among other careers)."

Coolabilities could be seen also within the context of neural computation and machine learning (particularly artificial neural networks). The objective is to find out whether artificially embedded "disabilities" into various neural network architectures may result in enhanced effectiveness or efficiency of the network (a coolability). Another possibility is to pretrain neural networks to make them resilient within various adversarial or disability contexts so that they will be able to address different disabilities or adversarial attacks that may appear during the run-time of the executable model within a real application. This is a challenging and interesting objective of studying neural architectures for so-called compensational coolabilities (when one or more abilities of a person or a model are strengthened due to the loss of another one) and even singular coolabilities (when unique abilities, which do not exist in other people or models, appear as a response to disability). Therefore, even if to accept the fact that the traditional machine learning process develops just one specific ability as a learning outcome, we can consider such specific coolabilities (additional capabilities that can be

developed as a result of the artificial disabilities) as effectiveness, efficiency, and resilience (among others) of a machine learning model (particularly a neural network).

The coolability concept is a positive connotation of a "disability" concept, which could be interpreted as an opportunity rather than a limitation. The concept can be useful in any context that aims to increase the value of people, resulting in a large impact on the labor market. Therefore, Nordfors et al. (2019) believe that systematically identifying and mapping correlations between disabilities and coolabilities, understanding underlying causalities will be the basis for a wide range of research and practice within different domains.

Regarding training correct social attitudes of healthy people toward people with disabilities, this issue is still a subject of concern in some training programs (see section 2). In a similar way, a healthy neural network could be pretrained for potential cases of disabilities. However, in the context of neural networks, such training is about preparedness for having some unfortunate disability in the future rather than training a correct "social attitude" toward the disability.

Other reasons for training correct social attitudes are applied to people (often sportsmen or military) who are developing skills to act in uncertain, complex, or adversarial conditions. Here we are not talking about training new extra skills but rather about the ability to perform with the usual set of skills but with some possible limitations or previously unseen complex situations enforced by an adversarial environment. In machine learning, a partially similar training paradigm is known as adversarial training (Kwon & Lee, 2021; Bai et al., 2021) by adversarial examples, with the objective of being prepared for adversarial inputs in the future and to promote the model's robustness.

Traditional adversarial training is related to training with adversarial samples, which are the inputs located closer to the decision options' distribution boundaries within the decision space (i.e., between different decision options). Disability simulation may also use the concept of training with adversarial samples; the adversarial samples may be intended to do more than insert some confusion in choosing the decision option but also to limit awareness due to the absence of some essential training data (voids within the decision space) or of the values for certain attributes within training samples needed to train the decision model. Another aspect of disability simulation would be intentionally cutting some important connections within the decision model aiming to pretrain the model to make better decisions with such a disability. Therefore, disability simulation inherits some basic philosophy of adversarial training but adds some specifics, which make the difference while training just a robust model or both a robust and a resilient one.

One special kind of adversarial training could be so-called ignorance-aware machine learning, which means artificial (intentional) creation of

voids within training data (modeling the loss of data) and pretraining the algorithms such as described in Terziyan and Nikulin (2021) to better discover such voids and fill them with the additional artificial training samples for self-supervised learning.

We used pruning as an important instrument for adversarial training. Traditional pruning techniques are applied to cut off the redundant and unimportant connections of a network and leave only the salient and essential structure. After pruning, the succinct network generalizes better and is more efficient. Pruning aims to eliminate weights that lead to overfitting (i.e., learning the noise). Deep neural architectures today expect pruning techniques to be able to choose carefully a layer-wise sparsity and find the acceptable trade-off between sparsity and the network's performance. Pruning is the process of determining which synapses (weights) of the network are not necessary (weakly contribute) to the final decision and removing them. Different pruning techniques are using different mathematical heuristics to determine unnecessary connections.

Consider some classical approaches first (Kavzoglu & Mather, 1998). The magnitude-based pruning technique (Hertz, Krogh & Palmer, 1991) is based on deleting synapses with small saliency (deletion will have the least effect on the training error) assuming that saliency corresponds to the magnitude (weight value) of a synapse. Smaller magnitudes are assumed to have minor effect on the network performance, and therefore the connections with the smallest magnitude will be discovered during initial training and removed. After that, the network will be retrained again, and the same pruning procedure will be performed iteratively until reaching an acceptable limit for the training error. Such a simple idea, however, often leads to the elimination of the wrong weights, as the small weights in some cases could be necessary to achieve a better error rate. The Optimum Brain Damage pruning method (LeCun, Denker & Solla, 1990; Gorodkin, Hansen, Krogh, Svarer, & Winther, 1993) uses second-order derivatives (collected as a Hessian matrix) of the error function. The method iteratively deletes the weights whose deletion results in the least increase in the error. The assumption of the method, which is a diagonal Hessian, may not be always true and may lead to deletion of the wrong weights. Otherwise, the computations of the Hessian matrix are a resource-consuming task. The Optimum Brain Surgeon pruning method (Hassibi & Stork, 1993) outperforms the previous one in robustness because it does not make any assumption about the form of the Hessian matrix. This complex method achieves higher sparsity than other methods with the same error; however, it requires more time and memory because the inverse of the Hessian matrix needs to be computed to assess the salience of every synapse. Pruning can also be viewed as an adaptive regularization (Hansen & Rasmussen, 1994), which is a special way to identify the regularization parameter (i.e., regularize with the infinite weight decay) based on minimization of the expected generalization error.

Pruning remains a popular topic and concern in neural computation. Many sophisticated pruning techniques have appeared. Xie (2020) suggested the redundancy-aware pruning of (convolutional) neural networks, which is performed within special intermediate space to eliminate correlation of neurons. This intermediate space is a result of mapping (by an orthogonal transformation) the input and output of a convolutional layer, and the pruning process in this space takes into account both neuron redundancy and neuron importance.

To make progress in decoding the structural basis of biological neural networks, Gerum, Erpenbeck, Krauss, and Schilling (2020) chose a bottom-up approach, where evolutionarily trained small neural networks are performing some maze tasks that require dynamic decision making with delayed rewards. They show that the random severing of connections (evolutionary pruning), without explicitly rewarding sparsity, leads to general sparsification of the networks and better generalization performance. Another evolutionary pruning strategy (multiobjective: training error versus computational complexity) is reported in Fernandes and Yen (2021). It is related to filter pruning in convolutional neural networks. The approach aims to find a reasonable solution instead of an optimal one; it does not require any extra knowledge during the pruning procedure and returns three pruned models with different trade-offs between performance and computational complexity (with the smallest training error, the smallest computational complexity, and the best trade-off between both). Malach, Yehudai, Shalev-Schwartz, and Shamir (2020) promoted pruning based on the so-called lottery ticket hypothesis (Frankle & Carbin, 2018), which states that a sufficiently overparameterized neural network with random weights contains a subnetwork with roughly the same accuracy as the target network. They show that pruning a random network achieves results that are competitive with optimizing the weights. Their comparison of neuron-based pruning versus weight-based pruning shows that the latter can achieve strictly stronger performance. The iSparse sparsification framework (Garg & Candan, 2020) is based on computing the novel edge significance score to determine the importance of a synapse with respect to the final network output. iSparse is a retraining-free framework that can be applied while training a model or on top of a pretrained model. Therefore, the framework enables minimization of the computational overhead. Luo et al. (2021) proposed pruning as a topological denoising applied on graph neural networks (powerful tools for graph analytics). The method penalizes and prunes task-irrelevant edges, taking into consideration the topology of the entire graph, which results in better generalization of the sparsified graph. Lemhadri, Ruan, Abraham, and Tibshirani (2021) suggested LassoNet as a feature sparsification framework with global feature selection. The approach achieves feature sparsity by adding a skip (residual) layer. Particular features are allowed to participate in the hidden layer of the network if

their skip-layer twins are active. This enables integration of feature selection with the parameter learning resulting to a good performance.

Studies on pruning and sparsity in neural networks combined with studies on functional connectivity within the human brain could be the subject of separate study. The correlation between brain regions might be very high even when they are not directly connected. Das et al. (2017) assumed that the connectivity structure is sparse. They suggested a method that uses a sparse regularized inverse covariance matrix for the connectivity modeling to identify regions that are conditionally independent by measuring strong, direct interactions between pairs of regions while simultaneously removing the influence of the rest of the regions. The demonstrations performed with the generated simple artificial networks show that the method appears to be able to identify functionally connected networks in the human brain. It has also been found that as long as the connectivity structure is sparse, the method can outperform other state-of-the-art methods.

Summarizing the state-of-the-art pruning techniques, we admit that all of them fit well to the objective they are designed for: the discovery and removal of redundant, unimportant synapses of a network, leaving only the salient and essential structure capable of generalizing better or performing more efficiently. In contrast, we try to approach pruning as an enforced disability that may be related to losing important, necessary, and not redundant connections within the model. Our goal is to discover the mechanism of model pretraining with enforced disabilities resulting from resilient architectures (coolabilities) capable of facing potential damage or attack at the run-time.

Consider the following scenario. Assume some neural network model has been designed for a particular decision-making task within a cyber-physical environment. The design process includes both: optimization of the network topology (i.e., pruning/sparsification aiming to remove redundant substructure) and training (i.e., optimizing the weights of the synapses). Assume the model shows the desired performance with the test data. However, if, at the testing stage, someone intentionally cuts some of the synapses, the network (optimal but not prepared for such an intrusion) will drop essentially its decision performance (error/accuracy rates). Such intrusion (attack or enforced disability) may destroy some important input information channels needed for making decisions or some important output channels needed to act in the physical world in accordance with the decisions (see Figure 1). Therefore, if (at the design stage) we assume the possibility of potential attacks on the trained and optimized model, then, in addition to (not instead of!) the traditional pruning for performance, we have to perform adversarial pretraining of several subnetworks with the enforced disabilities (pruning for coolability). After the adversarial training, the ensemble of the pretrained "handicaps" will perform as a resilient (coolability-enhanced) decision model prepared for losing some of its essential structural components on the fly.

7 Conclusion

We started our study based on an excellent discovery made by Nordfors et al. (2018), who demonstrated that the concept of coolability (enhanced capability in disabled conditions) can increase the value of people with disabilities and their impact on the labor market. In this letter, we expanded this concept to the world of AI, assuming that enhanced (cognitive) capabilities may appear in trainable AI models as compensation for possible or intended disabilities. We simulated various types of disabilities in neural networks and studied their coolability effects. We tried various architectures: shallow and deep feedforward neural networks, shallow and deep recurrent neural networks, and convolutional neural networks. We noticed that embedded disabilities in these architectures (classification or prediction models) lack any negative impact on the accuracy of the predictions; moreover, the accuracy is often essentially improved (because the resulting coolability feature operates as a well-structured special regularization technique). We named this positive coolability effect in AI "CAI"—a set of techniques to train or pretrain compensation for disabled AI systems or for AI systems working in adversarial environments, where disability may appear in run-time due to some attacks. We elaborated on the concept of a sustainable multichannel neural network architecture that can be trained in advance to face various disabilities in the future. Therefore, we argued that disability in AI is not only a problem but also an opportunity if addressed correctly. Our task was to present the concept and several possible architectures around it. The experiments with synthetic data provide some optimism regarding the usefulness of the coolability-driven CAI concept. Experiments with the real image data package (within the NATO SPS project Cyber-Defence for Intelligent Systems, Military Logistics Laboratory) show that the coolabilities within CNN architectures can be pretrained and used as a kind of immune system protecting some intelligent object or process (of critical or military infrastructure) from future attacks.

Therefore, we have addressed our research questions as follows. Regarding the types of considered disabilities in the neural architectures, we focused on sensor, functional, and memory disabilities related, respectively, to the broken input, output, or recurrent synapses of the neurons at different layers of the neural network. Regarding artificial embedding of the disabilities into neural architectures, we show how to enforce disabilities by special pruning and adversarial training of the pruned models. Regarding the benefits of such training, we consider (as the main advantage) the resilience of the neural architecture against potential attacks and acquired or enforced disabilities.

We are at the exploratory stage of the idea; however, by this letter, we suggest a road map for the deeper study of the coolability phenomenon, and we expect AI experts and data scientists to engage in further experiments

with real data for real problems. There is much to be done in this area, and we welcome other researchers to join us.

References

Alford, S., Robinett, R., Milechin, L., & Kepner, J. (2018). Pruned and structurally sparse neural networks. In *Proceedings of the IEEE MIT Undergraduate Research Technology Conference* (pp. 1–4). Piscataway, NJ: IEEE. 10.1109/URTC45901.2018.9244787

Bai, T., Luo, J., Zhao, J., Wen, B., & Wang, Q. (2021). *Recent advances in adversarial training for adversarial robustness.* In *Proceedings of the International Joint Conference on Artificial Intelligence* (IJCAI-21).

Barnes, E. (2016). *The minority body: A theory of disability*. New York: Oxford University Press.

Bartlett, J., & Holloway, E. (2019). Generalized information. A straightforward method for judging machine learning models. *Communications of the Blyth Institute*, *1*(2), 13–21. 10.33014/issn.2640-5652.1.2.bartlett.1

Brainerd, C. J. (2021). Deep memory distortions. *Cognitive Psychology*, *126*, 101386. 10.1016/j.cogpsych.2021.101386, PubMed: 33887617

Chatila, R., Renaudo, E., Andries, M., Chavez-Garcia, R.-O., Luce-Vayrac, P., Gottstein, R., . . . Khamassi, M. (2018). Toward self-aware robots. *Frontiers in Robotics*, *5*, 88. 10.3389/frobt.2018.00088

Das, A., Sampson, A. L., Lainscsek, C., Muller, L., Lin, W., Doyle, J. C., . . . Sejnowski, T. J. (2017). Interpretation of the precision matrix and its application in estimating sparse brain connectivity during sleep spindles from human electrocorticography recordings. *Neural Computation*, *29*(3), 603–642. 10.1162/NECO_a_00936, PubMed: 28095202

Dehaene, S., Lau, H., & Kouider, S. (2017). What is consciousness, and could machines have it? *Science*, *358*(6362), 486–492. 10.1126/science.aan8871, PubMed: 29074769

Fernandes, F. E., & Yen, G. G. (2021). Pruning deep convolutional neural networks architectures with evolution strategy. *Information Sciences*, *552*, 29–47. 10.1016/j.ins.2020.11.009

Frankle, J., & Carbin, M. (2018). *The lottery ticket hypothesis: Finding sparse, trainable neural networks.* arXiv:1803.03635.

Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Ranzato, M. A., & Mikolov, T. (2013). DeViSE: A deep visual-semantic embedding model. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *26* (pp. 2121–2129). Red Hook, NY: Curran.

Garg, Y., & Candan, K. S. (2020). iSparse: Output informed sparsification of neural network. In *Proceedings of the 2020 International Conference on Multimedia Retrieval* (pp. 180–188). New York: ACM. 10.1145/3372278.3390688

Gerum, R. C., Erpenbeck, A., Krauss, P., & Schilling, A. (2020). Sparsity through evolutionary pruning prevents neuronal networks from overfitting. *Neural Networks*, *128*, 305–312. 10.1016/j.neunet.2020.05.007

Gill, C. J. (2006). Disability, constructed vulnerability, and socially conscious palliative care. *Journal of Palliative Care*, *22*(3), 183–189. 10.1177/082585970602200309, PubMed: 17058757

Golovianko, M., Gryshko, S., Terziyan, V., & Tuunanen, T. (2021). Towards digital cognitive clones for the decision-makers: Adversarial training experiments. *Procedia Computer Science*, *180*, 180–189. 10.1016/j.procs.2021.01.155

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *27* (pp. 2672–2680). Red Hook, NY: Curran.

Goodley, D. (2016). *Disability studies: An interdisciplinary introduction*. Thousand Oaks, CA: Sage.

Gorodkin, J., Hansen, L. K., Krogh, A., Svarer, C., & Winther, O. (1993). A quantitative study of pruning by optimal brain damage. *International Journal of Neural Systems*, *4*(2), 159–169. 10.1142/S0129065793000146, PubMed: 8044376

Graves, A., Mohamed, A., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Proceedings of the 38th International Conference on Acoustics, Speech and Signal Processing* (pp. 6645–6649). Piscataway, NJ: IEEE. 10.1109/ICASSP.2013.6638947

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232. 10.1109/TNNLS.2016.2582924, PubMed: 27411231

Grundwag, C., Nordfors, D., & Yirmiya, N. (2017). *"Coolabilities": Enhanced abilities in disabling conditions*. SocArXiv. 10.31235/osf.io/stgd4

Hansen, L. K., & Rasmussen, C. E. (1994). Pruning from adaptive regularization. *Neural Computation*, *6*(6), 1223–1232. 10.1162/neco.1994.6.6.1223

Hassibi, B., & Stork, D. G. (1993). Second order derivatives for network pruning: Optimal Brain Surgeon. In S. J. Hanson, J. D. Cowan& C. L. Giles (Eds.), *Advances in neural information processing systems, 5* (pp. 164–171). San Mateo, CA: Morgan Kaufmann.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, *44*(1), 1–12. 10.1021/ci0342472, PubMed: 14741005

He, X., Zhao, K., & Chu, X. (2021). AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, *212*, 106622. 10.1016/j.knosys.2020.106622, PubMed: 33519100

Hertz, J., Krogh, A. & Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Reading, MA: Addison-Wesley.

Hihi, S. E., & Bengio, Y. (1996). Hierarchical recurrent neural networks for long-term dependencies. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems*, *8* (pp. 493–499). Cambridge, MA: MIT Press.

Hildt, E. (2019). Artificial intelligence: Does consciousness matter? *Frontiers in Psychology, 10*, 1535. 10.3389/fpsyg.2019.01535

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. 10.1162/neco.1997.9.8.1735, PubMed: 9377276

Kavzoglu, T., & Mather, P. M. (1998). Assessing artificial neural network pruning algorithms. In *Proceedings of the 24th Annual Conference and Exhibition of the Remote*

Sensing Society (pp. 9–11). https://www.academia.edu/download/37919129/RSS98.PDF

Kietzmann, J., Lee, L. W., McCarthy, I. P., & Kietzmann, T. C. (2020). Deepfakes: Trick or treat? *Business Horizons*, *63*(2), 135–146. 10.1016/j.bushor.2019.11.006

Kwon, H., & Lee, J. (2021). Diversity adversarial training against adversarial attack on deep neural networks. *Symmetry*, *13*(3), 428. 10.3390/sym13030428

LeCun, Y., & Bengio, Y. (1998). Convolutional networks for images, speech, and time series. In M. A. Arbib (Ed.), *The handbook of brain theory and neural networks* (pp. 255–258). Cambridge, MA: MIT Press.

LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal Brain Damage. In D. Touretzy (Ed.), *Advances in neural information processing systems*, *2* (pp. 598–605). San Mateo, CA: Morgan Kaufmann.

Lemhadri, I., Ruan, F., Abraham, L., & Tibshirani, R. (2021). LassoNet: A neural network with feature sparsity. *Journal of Machine Learning Research*, *22*(127), 1–29.

Li, B., Fan, Y., Pan, Z., Xi, T., & Zhang, G. (2020). *AutoPruning for deep neural network with dynamic channel masking.* arXiv:2010.12021.

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). *A critical review of recurrent neural networks for sequence learning.* arXiv:1506.00019.

Long, M., Zhu, H., Wang, J., & Jordan, M. I. (2017). Deep transfer learning with joint adaptation networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning* (vol. 70, pp. 2208–2217). 10.5555/3305890.3305909

Luo, D., Cheng, W., Yu, W., Zong, B., Ni, J., Chen, H., & Zhang, X. (2021). Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining* (pp. 779–787). New York: ACM. 10.1145/3437963.3441734

Malach, E., Yehudai, G., Shalev-Schwartz, S., & Shamir, O. (2020). Proving the lottery ticket hypothesis: Pruning is all you need. In *Proceedings of Machine Learning Research*, *119*, 6682–6691.

Nario-Redmond, M. R., Gospodinov, D., & Cobb, A. (2017). Crip for a day: The unintended negative consequences of disability simulations. *Rehabilitation Psychology, 62*(3), 324–333. 10.1037/rep0000127

Nordfors, D., Dasgupta, S. S., Subramanian, G., Ferose, V. R., Grundwag, C., & Zandi, B. (2018). Coolabilities API. In *Proceedings of the Fifth IEEE International Conference on Data Science and Advanced Analytics* (pp. 491–495). Piscataway, NJ: IEEE. 10.1109/DSAA.2018.00063

Nordfors, D., Grundwag, C., & Ferose, V. R. (2019). A new labor market for people with "coolabilities." *Communications of the ACM*, *62*(7), 26–28. 10.1145/3332809

Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2013). *How to construct deep recurrent neural networks.* arXiv:1312.6026.

Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, *29*(9), 2352–2449. 10.1162/neco_a_00990, PubMed: 28599112

Ren, K., Zheng, T., Qin, Z., & Liu, X. (2020). Adversarial attacks and defenses in deep learning. *Engineering*, *6*(3), 346–360. 10.1016/j.eng.2019.12.012

Rice, S. (2020, February 9). Blind side: Barcelona players forced to run around in training blindfolded in bizarre "chicken" drill by new coach Setien. *Sun*. Retrieved from https://www.thesun.co.uk/sport/10926320/barcelona-players-setien-blindfolds-training/

Rizzo, A., John, B., Newman, B., Williams, J., Hartholt, A., Lethin, C., & Buckwalter, J. G. (2013). Virtual reality as a tool for delivering PTSD exposure therapy and stress resilience training. *Military Behavioral Health*, *1*(1), 52–58. 10.1080/21635781.2012.721064

Sahota, N. (2918). Human 2.0 is coming faster than you think. Will you evolve with the times? *Forbes*, October 1.

Silverman, A. M. (2015). The perils of playing blind: Problems with blindness simulation and a better way to teach about blindness. *Journal of Blindness Innovation and Research*, *5*(2). https://www.nfb.org/images/nfb/publications/jbir/jbir15/jbir050201.html. 10.5241/5-81

Silverman, A. M., Pitonyak, J. S., Nelson, I. K., Matsuda, P. N., Kartin, D, & Molton, I. R. (2018). Instilling positive beliefs about disabilities: Pilot testing a novel experiential learning activity for rehabilitation students. *Disability and Rehabilitation*, *40*(9), 1108–1113. 10.1080/09638288.2017.1292321, PubMed: 28637146

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958. 10.5555/2627435.2670313

Terziyan, V. (2007). Challenges of the "global understanding environment" based on agent mobility. In V. Sugumaran (Ed.), *Application of agents and intelligent information technologies* (pp. 121–152). Hershey, PA: IGI Global.

Terziyan, V., Golovianko, M., & Gryshko, S. (2018). Industry 4.0 intelligence under attack: From cognitive hack to data poisoning. In K. Dimitrov (Ed.), *Cyber defence in industry 4.0 systems and related logistics and IT infrastructure* (pp. 110–125). Amsterdam: IOS Press.

Terziyan, V., Gryshko, S., & Golovianko, M. (2021). Taxonomy of generative adversarial networks for digital immunity of industry 4.0 systems. *Procedia Computer Science*, *180*, 676–685. 10.1016/j.procs.2021.01.290

Terziyan, V., & Nikulin, A. (2021). Semantics of voids within data: Ignorance-aware machine learning. *ISPRS International Journal of Geo-Information*, *10*(4), 246. 10.3390/ijgi10040246

Thodberg, H. H. (1991). Improving generalization of neural networks through pruning. *International Journal of Neural Systems*, *1*(04), 317–326. 10.1142/S0129065791000352

Truong, A., Walters, A., Goodsitt, J., Hines, K., Bruss, C. B., & Farivar, R. (2019). Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools. In *Proceedings of the 31st IEEE International Conference on Tools with Artificial Intelligence* (pp. 1471–1479). Piscataway, NJ: IEEE. 10.1109/ICTAI.2019.00209

Verkhoshansky, Y., & Siff, M. C. (2009). *Supertraining*. Verkhoshansky SSTM.

Wang, W., Zheng, V. W., Yu, H., & Miao, C. (2019). A survey of zero-shot learning: settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology*, *10*(2), 1–37. 10.1145/3293318

WHO. (2001). *International classification of functioning, disability and health: ICF*. Geneva: World Health Organization.

Wilson, H. J., & Daugherty, P. R. (2018). Collaborative Intelligence: Humans and AI are joining forces. *Harvard Business Review*, *96*(4), 114–123.

Xie, G. (2020). Redundancy-aware pruning of convolutional neural networks. *Neural Computation*, *32*(12), 2532–2556. 10.1162/neco_a_01330

Zhu, M., & Gupta, S. (2017). *To prune, or not to prune: Exploring the efficacy of pruning for model compression*. arXiv:1710.01878.

Received May 28, 2021; accepted July 15, 2021.