SCIENTIFIC REPORTS
nature research

OPEN

Brain experiments imply adaptation mechanisms which outperform common AI learning algorithms

Shira Sardi[1], Roni Vardi[2], Yuval Meir[1], Yael Tugendhaft[1], Shiri Hodassman[1], Amir Goldental[1] & Ido Kanter[1,2] ✉

Attempting to imitate the brain's functionalities, researchers have bridged between neuroscience and artificial intelligence for decades; however, experimental neuroscience has not directly advanced the field of machine learning (ML). Here, using neuronal cultures, we demonstrate that increased training frequency accelerates the neuronal adaptation processes. This mechanism was implemented on artificial neural networks, where a local learning step-size increases for coherent consecutive learning steps, and tested on a simple dataset of handwritten digits, MNIST. Based on our on-line learning results with a few handwriting examples, success rates for brain-inspired algorithms substantially outperform the commonly used ML algorithms. We speculate this emerging bridge from slow brain function to ML will promote ultrafast decision making under limited examples, which is the reality in many aspects of human activity, robotic control, and network optimization.

Machine learning is based on Donald Hebb's pioneering work; seventy years ago, he suggested that learning occurs in the brain through synaptic (link) strength modifications[1]. A synaptic strength modification typically lasts tens of minutes[2] while the clock speed of a neuron (node) ranges around one second[3]. Although the brain is comparatively slow, its computational capabilities outperform typical state-of-the-art artificial intelligence algorithms. Following this speed/capability paradox, we experimentally derive accelerated learning mechanisms based on small datasets, where their utilization on gigahertz processors is expected to lead to ultrafast decision making.

Unlike modern computers, a well-defined global clock does not govern brain dynamics; instead, they are a function of relative event timing (e.g., stimulations and evoked spikes)[4]. According to neuronal computational, using decaying input summation via its ramified dendritic trees, each neuron sums the asynchronous incoming electrical signals and generates a short electrical pulse (spike) when its threshold is reached. For each neuron, synaptic strength is slowly modified based on the relative timing of inputs from other synapses; if a signal is induced from a synapse without generating a spike, its associated strength is modified based on the relative timing to adjacent spikes from other synapses on the same neuron[5].

Recently it was experimentally demonstrated that each neuron functions as a collection of independent threshold units[6]. After signals arrive via one of the dendritic trees, each threshold unit is activated. Additionally, a new type of adaptive rule was experimentally observed based on dendritic signal arrival timing[7], which is similar to the slow adaptation mechanism currently attributed to synapses (links). This dendritic adaptation occurs on a faster timescale: it requires approximately five minutes, while synaptic modification requires tens of minutes or more.

Results

In this study, dendritic adaptation was experimentally examined at a higher stimulation frequency, 5 Hz, using the training pattern of previous experiments run at 0.5 to 1 Hz. We planted neuronal cultures on a multi-electrode-array with added synaptic blockers, which extracellularly stimulated a patched neuron via its dendrites (Fig. 1a and Materials and Methods). The adaptation process consisted of a training set: 50 pairs of

[1]Department of Physics, Bar-Ilan University, Ramat-Gan, 52900, Israel. [2]Gonda Interdisciplinary Brain Research Center and the Goodman Faculty of Life Sciences, Bar-Ilan University, Ramat-Gan, 52900, Israel. ✉e-mail: ido.kanter@biu.ac.il

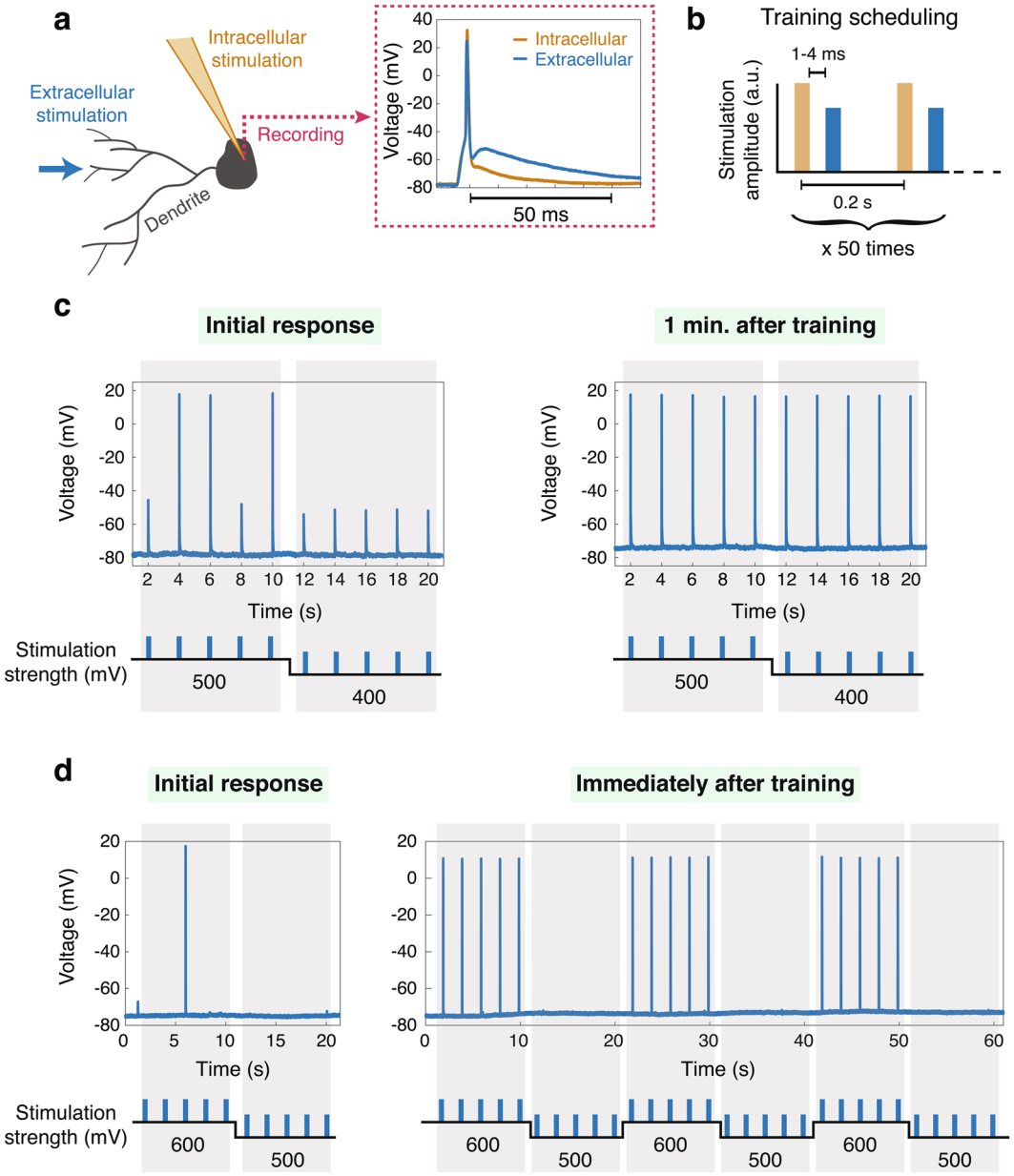

Figure 1. Experimental results indicate that adaptation rates increase with training frequency. (**a**) The experimental scheme where a patched neuron is stimulated intracellularly via its dendrites (Materials and Methods) and a different spike waveform is generated for each stimulated route. (**b**) The scheduling for coherent training consists of repeated pairs of intracellular stimulations (orange) generating a spike followed by an extracellular stimulation (blue) with the lack of a spike. (**c**) An example of the first type of experiments, where decreasing extracellular stimulation amplitude is used to estimate the threshold using intracellular recording (left), and enhanced responses measured a minute after the termination of the training, **b** (right). (**d**) An example of the second type of experiment, similar to **c**, where enhanced responses are observed 10 seconds after the termination of the training (Materials and Methods).

stimulations. After an above-threshold intracellular stimulation, an extracellular stimulation that did not evoke a spike arrived with a predefined delay, typically 1 to 4 ms (Fig. 1b). We primarily take into account differing extra- and intra-spike waveforms (Fig. 1a, right), which presumably activated the neuron from two independent dendritic trees[7].

By comparing the amplitudes of intracellular responses and extracellular stimulation before and after the training procedure, we quantified the effect of the neuronal adaptation. To quantify the initial response, the extracellular stimulation amplitude was decreased until no reliable evoked spikes were observed (Fig. 1c,d, left).

In the first type of experiment, one minute after the training terminated, we measured the enhanced responses and witnessed dendritic adaptation (Fig. 1c, right). When compared with the visible adaptation time for the 1 Hz training, that of the 5 Hz training was substantially faster. Occasionally, the visible effect of adaptation was further

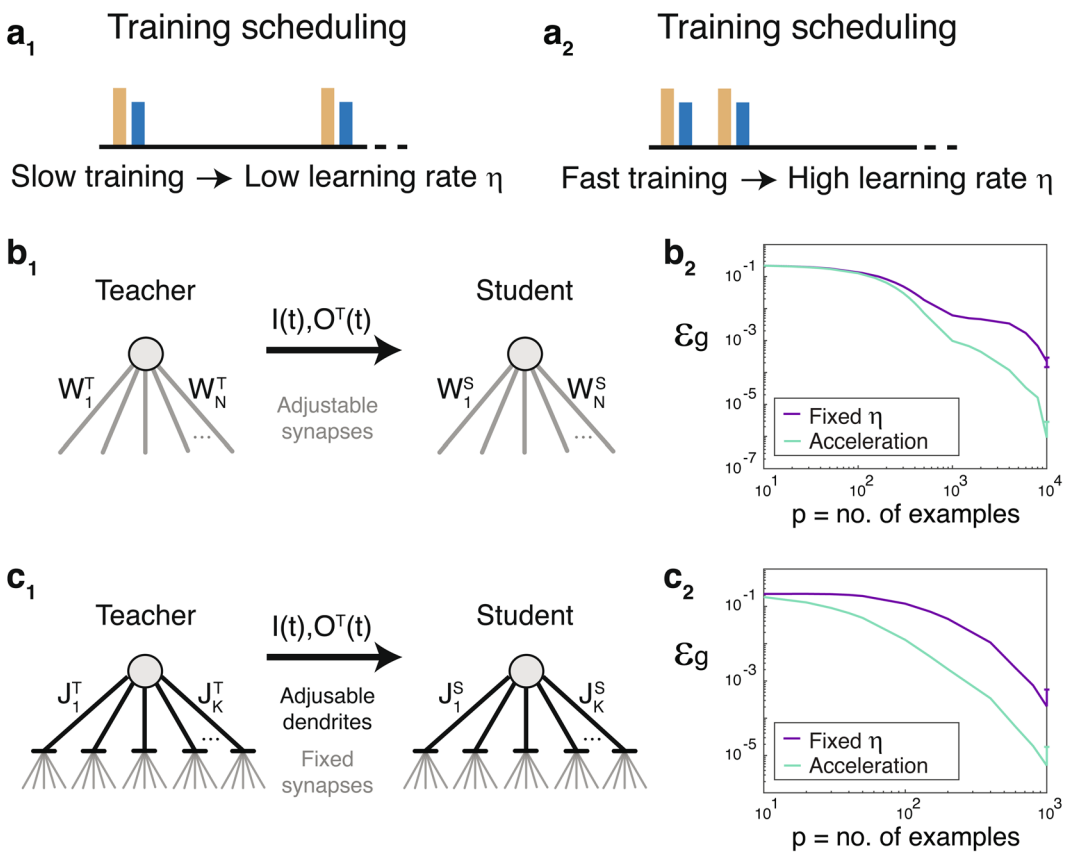

Figure 2. Acceleration of supervised realizable learning rules based on the biologically inspired mechanism. (**a**) The implication of the biological mechanism (Fig. 1) indicating that training scheduling with low/high frequency (**a**$_1$/**a**$_2$) results in a low/high learning rate, η. (**b**) Synaptic learning. (**b**$_1$) A perceptron with 1000 asynchronous inputs and a leaky integrate-and-fire output unit. The synapses of the teacher/student, W^T/W^S, are dynamically updated following a biological adaptation rule, a spike-time-dependent-plasticity (Materials and Methods). (**b**$_2$) The generalization error, ε_g, for the training of the student in **b**$_1$ using fixed learning step $\eta = 0.001$ (purple) and for the accelerating scenario $\eta^{t+1} = \eta^t \cdot \exp(-0.1) + 0.01 \cdot sign(O^T - O^S)$ (green), where O^T/O^S stands for a teacher/student output, and the initial $\eta = 0.001$. (**c**) Dendritic learning. (**c**$_1$) Dendritic learning, similar to **b**$_1$, where the 1000 synapses are fixed and the 200 dendrites of the teacher/student, J^T/J^S, are updated using similarly adaptive rule as in **b** (Materials and Methods). (**c**$_2$) ε_g for the trained student, **c**$_1$, using the same scenarios for η as in **b**$_2$. Results, **b**$_2$ and **c**$_2$, are averaged over 30 training datasets and the STD of ε_g is presented for the maximal presented p.

enhanced after more time passed (Supplementary Fig. 1), suggesting by extrapolation that adaptation might occur in much less than one minute. Because after training, the initialization and compilation of subsequent experiments required a minimal time lag (one minute), the feasibility of such ultrafast adaptation was impossible to examine.

To overcome this limitation, we introduced a second type of experiment (Fig. 1d): shortly after the end of the training procedure, the neuron was extracellularly stimulated using two predefined amplitudes with unreliable responses. Without requiring any new experimental methods or compilations (Materials and Methods), this procedure pinpointed dendritic adaptation within only 10 seconds of the training termination (Fig. 1d, right).

With the increased training frequency, the adaptation process substantially accelerated (Fig. 2a), potentially implying a time-dependent decaying adaptation step-size:

$$\eta_{adap}^{t+1} = \eta_{adap}^{t} \cdot e^{-\frac{\tau}{\tau_0}} + \Delta \tag{1}$$

where the current adaptation step, η_{adap}^{t+1}, is equal to the previous one with a decaying weight, t stands for a discrete time step, τ_0 is a constant, $1/\tau$ stands for the training frequency, and Δ is a constant representing the incremental effect of the current training step. This type of decay process occurs in many biological scenarios and represents, for instance, the decaying concentration of active material due to diffusion. As a generalization of Eq. (1), incoherent consecutive training steps are also allowed, decreasing the dendritic strength and resulting in a $-\Delta$ term in Eq. (1).

Using supervised on-line learning of realizable rules and binary classification (Fig. 2b,c), we first examined the impact of the time-dependent adaptation steps (Eq. 1) on accelerating biological learning processes. The teacher

provided the student asynchronous-input and binary-output relations[8], where both had the same architecture of the simplest classifier, the perceptron[9], and the output nodes were represented by a leaky integrate-and-fire neuron[10]. Two scenarios were examined: synaptic adaptation and dendritic adaptation (Fig. 2b,c and Materials and Methods). Results clearly indicate that the generalization error, ε_g, of the experimentally-inspired time-dependent η (Eq. 1) substantially outperformed the fixed η scenario (Fig. 2b,c). This accelerated learning stems from the fact that weights in synaptic learning converge to the extreme limits, vanishing or above-threshold weights[7]. Hence, the learning step-sizes in coherent dynamics increased toward these extremes, accelerating the learning scenario (Fig. 2b). Similarly, in the dendritic case (Fig. 2c), weights oscillated and synchronized via repeatedly hitting the boundary values[7]. Hence, a faster decay of ε_g also resulted from learning step acceleration (Fig. 2c).

Next, we examined the experimentally-inspired time-dependent learning step mechanism on the supervised learning of an unrealizable rule using the MNIST database[11] tested on a neural network. This database consists of a large number of examples of handwritten digits (Fig. 3a) and is commonly used as a prototypical problem for quantifying the generalization performance of machine learning algorithms for various image processing tasks. In this study we use a small subset of the MNIST database without any data extension methods[12–14]. The commonly used trained networks consisted of 784 inputs representing the 28×28 pixels of a digit, one hidden layer (30 units in this study), and ten outputs representing the labels (Fig. 3a). The commonly used learning approach is the backpropagation strategy[15]:

$$W^{t+1} = W^t - \eta \cdot \nabla_{W^t} C \tag{2}$$

where weight at time-step t, W^t, is modified with a step-size η towards the minus sign of the gradient of the cost function, C. An improved approach is the momentum strategy[5,16,17] and regularization of the weights[18,19]:

$$W^{t+1} = (1 - \alpha) \cdot W^t + V^{t+1} \tag{3}$$

$$V^{t+1} = \mu \cdot V^t - \eta_0 \cdot \nabla_{W^t} C$$

where the momentum, μ, and the regularization, α, are constants in the region [0, 1] and η_0 is a constant. We optimized the performance of the momentum strategy (Eq. 3) over $(\mu,\ \alpha,\ \eta_0)$ for a limited training dataset using the cross-entropy cost function (Materials and Methods) and compared its performance with the following two experimentally-inspired learning mechanisms consisting of time-dependent η.

In the first approach, acceleration, the time-dependent η, and the update rules for weight are given by these equations:

$$W^{t+1} = (1 - \alpha) \cdot W^t - |\eta^{t+1}| \cdot \nabla_{W^t} C \tag{4}$$

$$\eta^{t+1} = \eta^t \cdot e^{-\tau} + A_{1/2} \cdot tanh(\beta_{1/2} \cdot \nabla_{W^t} C)$$

where τ is the positive decaying factor, A_1 and β_1 are constants representing the amplitude and the gain between the input and the hidden layers, respectively, and A_2 and β_2 represent the same between the hidden and the output layers. It is evident that coherent consecutive gradients of weight, i.e., with the same sign, increased its conjugate η. Note, in the limit $\beta \to \infty$, the equation for η was simplified, $\eta^{t+1} = \eta^t \cdot \exp(-\tau) + A \cdot sign(\nabla_{W^t} C)$. The second approach, advanced acceleration, combines the two previous approaches (Eqs. 3, 4):

$$W^{t+1} = (1 - \alpha) \cdot W^t + V^{t+1} \tag{5}$$

$$V^{t+1} = \mu \cdot V^t - |\eta^{t+1}| \cdot \nabla_{W^t} C$$

$$\eta^{t+1} = \eta^t \cdot e^{-\tau} + A_{1/2} \cdot tanh(\beta_{1/2} \cdot \nabla_{W^t} C)$$

For the two experimentally-inspired accelerated-approaches (Eqs. 4, 5), the changes in weights and η depend on the higher moments of gradients, in contrast with the linear dependence of the momentum approach (Eq. 3). Given a limited subset of the dataset examples the performance of the accelerated approaches was maximized over six (Eq. 4) and seven (Eq. 5) parameters (Materials and Methods).

The on-line training set consisted of 300 randomly chosen examples: each label appeared 30 times within a random order. After 300 learning steps, the accelerated approaches outperformed the momentum method by more than 25%, and the test accuracy increased from about 0.43 to 0.54, respectively (Fig. 3b). Note, the acceleration approach (Eq. 4) showed similar performance to the advanced acceleration approach (Eq. 5). These improved results were found to be robust also for on-line training based on 6000 examples (30 batches of size 200) (Fig. 3c) and 1200 examples (24 batches of size 50) (Fig. 3d). For both cases (Fig. 3c,d), the advanced acceleration approach (Eq. 5) offered the best performance. We repeated the training using the same 60 examples 5 times ($60 \times 5 = 300$ training in total); compared with using 300 examples for training once, the 60×5 approach yielded better performance: using the advanced acceleration approach test accuracy increased from 0.54 to 0.57 (Fig. 3e and Supplementary Fig. 2). For a given number of network updates, results demonstrate that smaller example sets yield more information. This result stems from the random training order of a randomly selected small dataset, e.g., 60 or 300 examples, consisting of a balanced appearance for each label. Around equalized trained label appearances, there are more temporal fluctuations for a dataset involving 300 examples with 30 appearances for

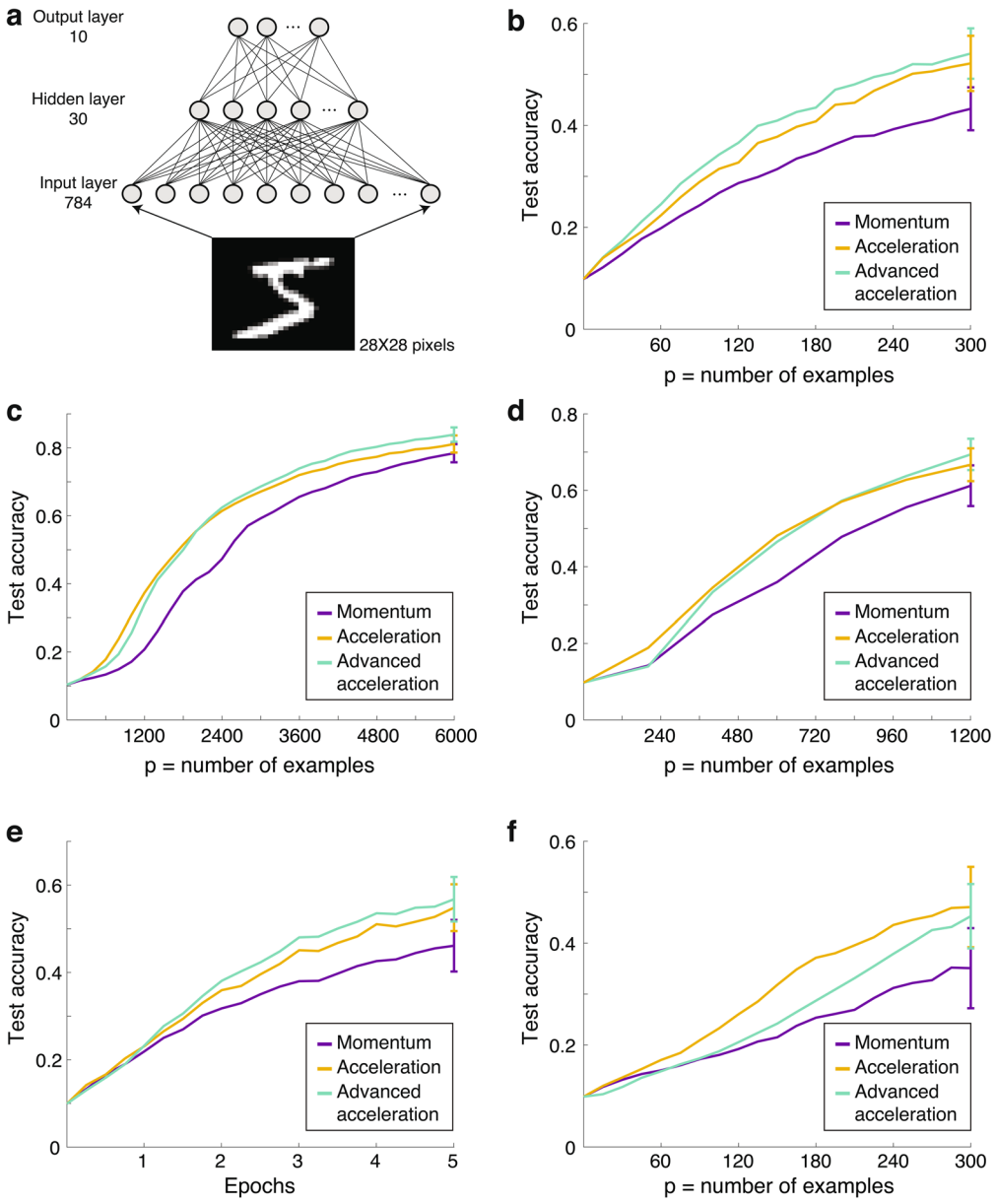

Figure 3. Biological-inspired accelerating learning for the MNIST database in comparison with a common existing learning method. (**a**) The trained network, using backpropagation and cross-entropy cost function, consists of 784 (28×28) input units representing the pixels of an MNIST digit, 30 hidden units, and 10 outputs representing the probabilities for the possible labels (Materials and Methods). (**b**) The maximal attainable test accuracy for 300 examples trained only once by the network in **a**, using the following methods; momentum, Eq. (3) (purple), acceleration, Eq. (4) (orange) and advanced acceleration, Eq. (5) (green). (**c**) Similar to **b** using 6000 examples composed of 30 batches of 200. (**d**) Similar to **b** using 1200 examples composed of 24 batches of 50. (**e**) 60 examples are trained 5 times. The total number of presented examples to the network, **a**, is 300 (60×5) as in **b**. (**f**) Similar to **b** using mean-square-error cost-function. Results, panels **b**-**f**, are averaged over 100 training datasets and the STD of the test accuracy is presented for the last trained example.

each label than for one involving 5 sets of 60 examples with 6 appearances for each label. Indeed, for a training set of 300 distinct examples composed of 5 subsets of 60 balanced examples, a test accuracy of 0.57 was achieved (Supplementary Fig. 3), and for one with 30 subsets of 10 examples where each label appears once, the test accuracy increased further to 0.67 and to 0.7 for a fixed label order (Supplementary Fig. 4). Results indicate that in order to maximize the test accuracy for on-line scenarios and especially for small datasets, the balanced set of examples and their balanced temporal training order are important ingredients.

Conclusions

Based on increased η with coherent consecutive gradients, the brain-inspired accelerated-learning mechanism outperforms existing common ML strategies for small sets of training examples[20]. Consistent results occur across various cost functions, e.g., square cost-function, however, with a relatively diminished performance (Fig. 3f). Because the performance maximization for a given dataset depends on the selected acceleration approach (Fig. 3b,c), adapting the learning approach during the training process may improve performance. Nevertheless, in addition to possible advanced nonlinear functions for updating η, given the number of network updates, the ultimate scheduling of acceleration approaches and the ordering of trained examples to maximize the performance deserves further research. The presented bridge from experimental neuroscience to ML is expected to further advance decision making using limited databases, which is the reality in many aspects of human activity[21], robotic control[22,23], and network optimization[24,25].

Materials and Methods

***In-Vitro* experiments.** The experimental methods are similar to our previous studies[6,7] and only the modifications are presented. All procedures were in accordance with the National Institutes of Health Guide for the Care and Use of Laboratory Animals and Bar-Ilan University Guidelines for the Use and Care of Laboratory Animals in Research and were approved and supervised by the Bar-Ilan University Animal Care and Use Committee.

Experiments protocol. The details of the experimental protocol of Figure 1 are as follows. The neuronal response latency (NRL) was used to accurately adjust the time-lag between intracellular evoked spikes or EPSPs originated from consecutive intra- and extra- cellular stimulations to be in the range of 1–4 ms. We note that an above-threshold extracellular stimulation given shortly, e.g. 2 ms, after an above-threshold intracellular stimulation, does not result in an evoked spike, and can be used to enhance adaptation. The thresholds and NRL were rechecked at the end of the experiment, in order to ensure their stability.

Statistical analysis. The demonstrated results were repeated tens of times on many cultures.

Simulations of biological neural networks. The simulation methods are similar to our previous studies[8] and only the used parameters and modifications are presented.

Inputs generation. Each input was composed of N/2 randomly stimulated input units. For each stimulated unit a random delay and a stimulation amplitude were chosen from given distributions. The delays were randomly chosen from a uniform distribution with a resolution of 1 (0.001) ms, such that the average time-lag between two consecutive stimulations was 2 (10) ms for the synaptic (dendritic) scenario. Stimulation amplitudes were randomly chosen from a uniform distribution in the range [0.8, 1.2]. Note that the reported results are qualitatively robust to the scenario where all the non-zero amplitudes equal 1. In the dendritic scenario, the five W_m connected to the same dendrite were stimulated sequentially in a random order and with an average time-lag of 10 ms between consecutive stimulations.

Calculating the generalization error. The estimation consisted of up to 20,000 inputs presented to the teacher and the student, where each input generates about 30/200 evoked spikes for the synaptic/dendritic scenario. The generalization error is defined as

$$\varepsilon_g = \frac{total\ no.\ of\ mismatch\ firing}{total\ no.\ of\ stimulations}.$$

The details of the simulations in Figure 2 are as follows. Panel B: $\{W_m\}$ were chosen from a uniform distribution in the range [0.1, 0.2]. The adaptation and learning steps were A = 0.05 and η = 1/1000, respectively. W_m was bounded from above by 1.5 and from below by 10^{-4}. The fixed learning rate was compared to the accelerating method using adaptive learning step:

$$\eta^{t+1} = \eta^t \cdot e^{-\tau} + B \cdot sign(O^T - O^S)$$

using τ = 0.1, B = 0.01 and η was initiated as 1/1000.

Panel C: $\{W_m\}$ were chosen from a uniform distribution in the range [0.1, 0.9] and then were normalized to a mean equals to 0.5. $\{J_i\}$ were chosen from a uniform distribution in the range [0.5, 1.5]. Stimulations with low amplitudes (0.01) were given to the N/2 unstimulated input units, resulting in non-frozen J_i. The adaptation and learning steps were A = 0.05 and η = 1/1000, respectively. J_i was bounded from below by 0.1 and from above by 3. The fixed learning rate was compared to the accelerating method using adaptive learning step:

$$\eta^{t+1} = \eta^t \cdot e^{-\tau} + B \cdot sign(O^T - O^S)$$

using τ = 0.1, B = 0.01 and η was initiated as 1/1000.

Simulations of neural network. *Architecture.* The feedforward neural network contains 784 input units, 30 hidden units and 10 output units in a fully connected architecture. Each unit in the hidden and the output layers has an additional input from a bias unit. Weights from the input layer to the hidden layer, W_1, and from the hidden layer to the output layer, W_2, were randomly chosen from a Gaussian distribution with a zero average and standard deviation equals 1. All weights were normalized such that all input weights to each hidden unit have

an average equals 0 and a STD equals 1. The initial value of the bias of each weights was set to 1. We trained the network on the handwritten digits dataset, MNIST, using gradient descent. The inputs, examples from the train dataset, contain 784 pixel values in the range [0, 255]. We normalized the inputs such that the average and the STD are equal to 0 and 1, respectively.

Forward propagation. The output of a single unit in the hidden layer, a_j^1, was calculated as:

$$z_j^1 = \sum_i (W_{ij}^1 \cdot X_i)$$

$$a_j^1 = \frac{1}{1 + e^{-Z_j^1}}$$

where W_{ij}^1 is the weight from the ith input to the jth hidden unit, X_i is the ith input, and b$^1{}_j$ is the bias for the jth hidden unit.

For the output layer, the output of a single unit, a_j^2 was calculated as:

$$z_j^2 = \sum_i (W_{ij}^2 \cdot a_i^1)$$

$$a_j^2 = \frac{1}{1 + e^{-Z_j^2}}$$

where W_{ij}^2 is the weight from the ith hidden unit to the jth output unit, a_i^1 is the output of the ith unit in the hidden layer, and b$^2{}_j$ is the bias for the jth output unit.

Back propagation. We used two different cost functions; the first was the cross entropy:

$$C = -\frac{1}{N}\sum_n \left[y * \log(a) + (1 - y) * \log(1 - a) \right]$$

and the second was the mean square error (MSE):

$$C = \frac{1}{2N}\sum_n (y - a)^2$$

where y are the desired labels and a stands for the current 10 output units of the output layer. The summation is over all training examples, N.

The backpropagation method computes the gradient for each weight with respect to the chosen cost function. The weights and biases were updated according to 3 different methods:

(1) Momentum
The weights update:

$$W^{t+1} = (1 - \alpha) \cdot W^t + V^{t+1}$$

$$V^{t+1} = \mu \cdot V^t - \eta_0 \cdot \nabla_{W^t} C$$

where t is the discrete time-step W are the weights, α is a regularization constant, η is the fixed learning rate, and $\nabla_{W^t} C$ is the gradient of the cost function for each weight at time t. V was initialized as: $-\eta_0 \cdot \nabla_W C_{first}$, where $\nabla_W C_{first}$ is the first computed gradient and the biases update:

$$V_b^{t+1} = \mu \cdot V_b^t - \eta_0 \cdot \nabla_{b^t} C$$

$$b^{t+1} = b^t + V_b^{t+1}$$

where $\nabla_b C$ is the gradient of the cost function of each bias with respect to its weight, b, and V_b was initialized as: $-\eta \cdot \nabla_b C_{first}$, where $\nabla_b C_{first}$ is the first computed bias gradient.

(2) Acceleration
The weights update:

$$W^{t+1} = (1 - \alpha) \cdot W^t - |\eta^{t+1}| \cdot \nabla_{W^t} C$$

$$\eta^{t+1} = \eta^t \cdot e^{-\tau} + A_{1/2} \cdot tanh(\beta_{1/2} \cdot \nabla_{W^t} C)$$

where η is defined for each weight, A_1 and β_1 are constants representing the amplitude and the gain

between the input and the hidden layers, respectively, and A_2 and β_2 represent the same between the hidden and the output layers. η was initialized as: $A_{1/2} \cdot tanh\left(\beta_{1/2} \cdot \nabla_W C_{first}\right)$, where $\nabla_W C_{first}$ is the first computed gradient and the biases update:

$$b^{t+1} = b^t - \left|\eta_b^{t+1}\right| \cdot \nabla_{b^t} C$$

$$\eta_b^{t+1} = \eta_b^t \cdot e^{-\tau} + A_{1/2} \cdot tanh(\beta_{1/2} \cdot \nabla_{b^t} C)$$

(3) Advanced acceleration:

The weights update

$$W^{t+1} = (1 - \alpha) \cdot W^t + V^{t+1}$$

$$V^{t+1} = \mu \cdot V^t - \eta^{t+1} \cdot \nabla_{W^t} C$$

$$\eta^{t+1} = \eta^t \cdot e^{-\tau} + A_{1/2} \cdot tanh(\beta_{1/2} \cdot \nabla_{W^t} C)$$

and the biases update:

$$b^{t+1} = b^t + V_b^{t+1}$$

$$V_b^{t+1} = \mu \cdot V_b^t - \eta_b^{t+1} \cdot \nabla_{b^t} C$$

$$\eta_b^{t+1} = \eta_b^t \cdot e^{-\tau} + A_{1/2} \cdot tanh(\beta_{1/2} \cdot \nabla_{b^t} C)$$

Testing the network. The network classification accuracy was tested on the MNIST dataset for testing, containing 10,000 inputs. The test inputs were also normalized to have an average of each equals to 0 and a STD equals to 1.

Optimization. For each update method the parameters were chosen to maximize the test accuracy. For optimization we first used a grid of the adjustable parameters followed by a fine tuning with higher resolution for each parameter. The optimization was performed over 3 parameters for the momentum method (μ, η_0, α), 6 parameters for the acceleration method (A_1, A_2, β_1, β_2, τ, α) and for 7 parameters for the advanced acceleration method (A_1, A_2, β_1, β_2, τ, α, μ).

The details of the simulations in Figure 3 are as follows. Panel B: The feed forward neural network was presented with 300 examples with equally number of appearance of each digit. The cross entropy cost function was used and the following parameters for each method: momentum with $\mu = 0.9$, $\eta_0 = 0.02$, $\alpha = 0$, acceleration with $A_1 = 0.1$, $A_2 = 0.27$, $\beta_1 = 5000$, $\beta_2 = 900$, $\tau = 0.598$, $\alpha = 0.003$, and advanced acceleration with $A_1 = 0.07$, $A_2 = 0.07$, $\beta_1 = \infty$, $\beta_2 = \infty$, $\tau = 0.29$, $\alpha = 0.005$, $\mu = 0.5$. Results are presented as the average of 100 different runs, and typical error bars are presented for the last point.

Panel C: The feed forward neural network was presented with 6000 examples randomly taken from the 60000 examples in the training dataset, and presented to the network with 30 mini-batches of size 200. The cross entropy cost function was used and the following parameters for each method: momentum with $\mu = 0.68$, $\eta_0 = 0.65$, $\alpha = 0.08$, acceleration with $A_1 = 3$, $A_2 = 0.7$, $\beta_1 = 1500$, $\beta_2 = 1000$, $\tau = 0.4$, $\alpha = 0.065$, and advanced acceleration with $A_1 = 0.5$, $A_2 = 0.5$, $\beta_1 = \infty$, $\beta_2 = \infty$, $\tau = 0.1$, $\alpha = 0.1$, $\mu = 0.55$. Results are presented as the average of 100 different runs, and typical error bars are presented for the last point.

Panel D: The feed forward neural network was presented with 1200 examples randomly taken from the 60000 examples in the training dataset, and presented to the network with 24 mini-batches of size 50. The cross entropy cost function was used and the following parameters for each method: momentum with $\mu = 0.75$, $\eta_0 = 0.6$, $\alpha = 0.11$, acceleration with $A_1 = 1.15$, $A_2 = 0.6$, $\beta_1 = 4500$, $\beta_2 = 3500$, $\tau = 0.1$, $\alpha = 0.055$, and advanced acceleration with $A_1 = 0.55$, $A_2 = 0.55$, $\beta_1 = \infty$, $\beta_2 = \infty$, $\tau = 0.1$, $\alpha = 0.1$, $\mu = 0.55$. Results are presented as the average of 100 different runs, and typical error bars are presented for the last point.

Panel E: The feed forward neural network was presented with 60 with equally number of appearance of each digit. The 60 examples were presented to the network 5 times. The cross entropy cost function was used and the following parameters for each method: momentum with $\mu = 0.87$, $\eta_0 = 0.035$, $\alpha = 0.005$, acceleration with $A_1 = 0.11$, $A_2 = 0.26$, $\beta_1 = 2000$, $\beta_2 = 2500$, $\tau = 0.45$, $\alpha = 0.008$, and advanced acceleration with $A_1 = 0.035$, $A_2 = 0.02$, $\beta_1 = 4500$, $\beta_2 = 5$, $\tau = 0.0619$, $\alpha = 0.01$, $\mu = 0.6$. Results are presented as the average of 100 different runs, and typical error bars are presented for the last point.

Panel F: The feed forward neural network was presented with 300 examples with equally number of appearance of each digit. The mean-square-error cost function was used and the following parameters for each method: momentum with $\mu = 0.6$, $\eta_0 = 0.35$, $\alpha = 0.005$, acceleration with $A_1 = 0.95$, $A_2 = 0.25$, $\beta_1 = 5000$, $\beta_2 = 40$, $\tau = 0.15$, $\alpha = 0.008$, and advanced acceleration with $A_1 = 0.06$, $A_2 = 0.09$, $\beta_1 = 2100$, $\beta_2 = 1$,

$\tau = 0.015$, $\alpha = 0.005$, $\mu = 0.8$. Results are presented as the average of 100 different runs, and typical error bars are presented for the last point.

The feed forward neural network was presented with 300 examples with equally number of appearance of each digit. The examples are composed of 5 subsets of 60 balanced examples, where in every subset each label appears exactly 6 times. The cross entropy cost function was used and the following parameters for each method: momentum with $\mu = 0.87$, $\eta_0 = 0.035$, $\alpha = 0.005$, acceleration with $A_1 = 0.11$, $A_2 = 0.26$, $\beta_1 = 2000$, $\beta_2 = 2500$, $\tau = 0.45$, $\alpha = 0.008$, and advanced acceleration with $A_1 = 0.035$, $A_2 = 0.02$, $\beta_1 = 4500$, $\beta_2 = 5$, $\tau = 0.0619$, $\alpha = 0.01$, $\mu = 0.6$. Results are presented as the average of 100 different runs, and typical error bars are presented for the last point.

The feedforward neural network was presented with 300 examples with equally number of appearance of each digit. The examples are composed of 30 subsets of 10 balanced examples, where in every subset each label appears exactly one time. Results are also presented for the advanced acceleration method where for each subset of 10 examples, the labels were in a fixed order. The cross entropy cost function was used and the following parameters for each method: momentum with $\mu = 0.87$, $\eta_0 = 0.035$, $\alpha = 0.005$, acceleration with $A_1 = 0.11$, $A_2 = 0.26$, $\beta_1 = 2000$, $\beta_2 = 2500$, $\tau = 0.45$, $\alpha = 0.008$, and advanced acceleration with $A_1 = 0.035$, $A_2 = 0.02$, $\beta_1 = 4500$, $\beta_2 = 5$, $\tau = 0.0619$, $\alpha = 0.01$, $\mu = 0.6$. Results are presented as the average of 100 different runs, and typical error bars are presented for the last point.

Received: 5 February 2020; Accepted: 31 March 2020;
Published online: 23 April 2020

References

1. Hebb, D. The organization of behavior. *A Neuropsychological* (1949).
2. Dan, Y. & Poo, M.-m Hebbian depression of isolated neuromuscular synapses *in vitro*. *Science* **256**, 1570–1573 (1992).
3. Turrigiano, G. G. & Nelson, S. B. Homeostatic plasticity in the developing nervous system. *Nature Reviews Neuroscience* **5**, 97 (2004).
4. Markram, H., Gerstner, W. & Sjöström, P. J. Spike-timing-dependent plasticity: a comprehensive overview. *Frontiers in synaptic neuroscience* **4**, 2 (2012).
5. Perantonis, S. J. & Karras, D. A. An efficient constrained learning algorithm with momentum acceleration. *Neural Networks* **8**, 237–249 (1995).
6. Sardi, S., Vardi, R., Sheinin, A., Goldental, A. & Kanter, I. New Types of Experiments Reveal that a Neuron Functions as Multiple Independent Threshold Units. *Sci Rep-Uk* **7**, 18036 (2017).
7. Sardi, S. *et al*. Adaptive nodes enrich nonlinear cooperative learning beyond traditional adaptation by links. *Sci Rep-Uk* **8**, 5100 (2018).
8. Uzan, H., Sardi, S., Goldental, A., Vardi, R. & Kanter, I. Biological learning curves outperform existing ones in artificial intelligence algorithms. *Sci Rep-Uk* **9**, 11558, https://doi.org/10.1038/s41598-019-48016-4 (2019).
9. Watkin, T. L., Rau, A. & Biehl, M. The statistical mechanics of learning a rule. *Reviews of Modern Physics* **65**, 499 (1993).
10. Brette, R. & Gerstner, W. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of neurophysiology* **94**, 3637–3642 (2005).
11. LeCun, Y. *et al*. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective* **261**, 276 (1995).
12. Zhang, Y. & Ling, C. A strategy to apply machine learning to small datasets in materials science. *Npj Computational Materials* **4**, 25 (2018).
13. Shorten, C. & Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of Big Data* **6**, 60 (2019).
14. Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. In *Advances in neural information processing systems*. 3320–3328 (2014).
15. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436 (2015).
16. Sutskever, I., Martens, J., Dahl, G. & Hinton, G. In *International conference on machine learning*. 1139–1147 (2013).
17. LeCun, Y. A., Bottou, L., Orr, G. B. & Müller, K.-R. In *Neural networks: Tricks of the trade* 9–48 (Springer, 2012).
18. Jin, Y., Okabe, T. & Sendhoff, B. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*. 1-8 (IEEE).
19. Bishop, C. M. *Neural networks for pattern recognition. (Oxford university press, 1995)*.
20. Hoffmann, J. *et al*. Machine learning in a data-limited regime: Augmenting experiments with synthetic data uncovers order in crumpled sheets. *Science advances* **5**, eaau6792 (2019).
21. Kearns, M., Suri, S. & Montfort, N. An experimental study of the coloring problem on human subject networks. *Science* **313**, 824–827 (2006).
22. Edelman, B. *et al*. Noninvasive neuroimaging enhances continuous neural tracking for robotic device control. *Science Robotics* **4**, eaaw6844 (2019).
23. Saridis, G. Intelligent robotic control. *IEEE Transactions on Automatic Control* **28**, 547–557 (1983).
24. Mateo, D., Horsevad, N., Hassani, V., Chamanbaz, M. & Bouffanais, R. Optimal network topology for responsive collective behavior. *Science Advances* **5**, eaau0999 (2019).
25. Liu, L., Cheng, Y., Cai, L., Zhou, S. & Niu, Z. In *2017 IEEE international conference on communications (ICC)*. 1-6 (IEEE).

Acknowledgements

We would like to thank Editage (www.editage.com) for English language editing.

Author contributions

S.S. and R.V. contributed equally to all experimental result aspects while Y.T. prepared the tissue cultures and materials for the experiments. S.S. and Y.M. contributed equally to the simulation results. S.H. confirmed some of the simulation results. A.G. contributed to conceptualization and for the data analysis. I.K. initiated the study and supervised all aspects of the work. All authors discussed the results and commented on the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41598-020-63755-5.

Correspondence and requests for materials should be addressed to I.K.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.