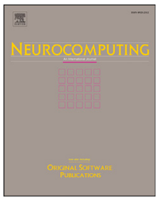
Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Towards biologically plausible DNN optimization: Replacing backpropagation and loss functions with a top-down credit assignment network

Jianhui Chen[a,b], Tianming Yang[a], Cheng-Lin Liu[b,c,*], Zuoren Wang[a,d,e,**]

[a] Institute of Neuroscience, State Key Laboratory of Brain Cognition and Brain-inspired Intelligence Technology, Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai, China
[b] State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[c] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[d] School of Future Technology, University of Chinese Academy of Sciences, Beijing, China
[e] School of Life Science and Technology, ShanghaiTech University, Shanghai, China

ARTICLE INFO

Communicated by M. Bianchini

Keywords:
Deep neural networks
Top-down projection
Biologically plausible
Learning strategies

ABSTRACT

Since the biological brain is capable of fast and energy-efficient learning, one way to improve current neural network optimization techniques is by drawing inspiration from the brain's learning paradigm. The biological implausibility of current learning processes based on loss functions and backpropagation (BP) may limit their optimization potential. In the brain, top-down modulation might act as a counterpart to loss functions and BP. Inspired by this mechanism, we propose a Top-Down Credit Assignment Network (TDCA-network) to optimize bottom-up networks, serving as a substitute for conventional loss functions and backpropagation. Each TDCA-network configuration corresponds to specific combinations of loss functions and learning rules. Applications of the TDCA-network to non-convex function optimization, supervised learning, and reinforcement learning reveal that a well-trained TDCA-network outperforms traditional learning methods across various paradigms, network architectures, and datasets. Additionally, we introduce a brain-inspired credit diffusion mechanism—a capability unattainable by BP—to further demonstrate the TDCA-network's potential. This mechanism enables the TDCA-network to achieve lower computational costs than BP while maintaining high performance. The most significant advantage of the TDCA-network lies in its ability to represent diverse loss functions and learning rules, offering a promising framework for the future exploration of fast and efficient learning algorithms for tasks that remain unconceptualized by humans.

1. Introduction

Understanding brain intelligence origins is vital in human technology. Artificial neural networks (ANNs), inspired by the biological brain, derive principles from biological neural networks where neurons form an information processing network. These models have made significant progress in different tasks, especially in computer vision and natural language processing. However, optimization of ANNs is still not as fast and efficient as the biological brain. Aiming to fill in the gap, biologically plausible learning algorithms have drawn wide attention in the research community.

In traditional machine learning, a loss function is typically set for the learning process, followed by a back-propagation algorithm to optimize this loss. Therefore, a biologically plausible learning algorithm should specifically focus on modifications to the loss function and the backpropagation process. The loss function, which is designed and explicitly defined by humans, possesses desirable properties that make it suitable for optimization using gradient descent methods. However, research in neuroscience suggests that the brain does not operate like artificial intelligence systems, where an explicitly represented numerical

* Corresponding author at: State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China.
** Corresponding author at: Institute of Neuroscience, State Key Laboratory of Brain Cognition and Brain-inspired Intelligence Technology, Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai, China.
Email addresses: jhchen@ion.ac.cn (J. Chen), tyang@ion.ac.cn (T. Yang), liucl@nlpr.ia.ac.cn (C.-L. Liu), zuorenwang@ion.ac.cn (Z. Wang).

https://doi.org/10.1016/j.neucom.2025.132330
Received 29 July 2025; Received in revised form 26 November 2025; Accepted 3 December 2025
Available online 4 December 2025

loss signal is used to guide learning. Instead, its optimization objectives may be implicit and distributed, rather than explicitly represented in a symbolic form [1,2]. The backpropagation algorithm, calculating each neuron's synaptic weights and updating gradients in the network through symbolic chain differentiation, is widely used for training neural network models [3]. However, the BP algorithm is also not found in the brain [4,5]. From both artificial intelligence and neuroscience perspectives, understanding how the brain represents loss function and learning rule is a significant research question. Previous methods on biologically plausible algorithms primarily focus on the biologically plausibility of back-propagation [6–9] by addressing the issues of weight transport[10,11], non-local plasticity [12,13], update locking [14,15], and global loss function [1,16–18]. However, they have not considered that a symbolic loss function itself may also be biologically implausible. Another line of work focuses on meta-learning loss functions, where some approaches parameterize the loss function using neural networks [19,20,20–25]. While these methods partially address the biological plausibility of loss functions, they are not explicitly designed with biological constraints in mind, leaving several important questions unresolved. A detailed review of related work is provided in Section 2. In contrast, our approach emphasizes the biological plausibility of both the loss function and the backpropagation process.

To address the issue of biological feasibility in the neural network training process, we need to refer to the ways in which brains implement learning. In neuroscience, top-down connections involve projections from brain areas responsible for higher-order cognitive functions to preceding cortical areas. Research shows that these top-down mechanisms play a key role in learning [2,4,26–31]. Thus, we used a top-down network to replace both the loss function and back-propagation, as shown in Fig. 1, making the learning process more biologically plausible. The top-down network directly generates the parameter gradient by monitoring the state of bottom-up network. This paradigm enables full network learning without a manually defined symbolic loss function or back-propagation. Given that neural networks are universal approximators [32], the TDCA-network has the potential to encode all possible loss functions and learning methods. Conducting parameter searches on these TDCA-networks is equivalent to exploring new loss functions and optimization algorithms. Meanwhile, it addresses the issues of weight transport, weight symmetry, non-local plasticity, and precise gradient calculation associated with traditional BP algorithms. Although we still need a measurement to evaluate the optimization quality during the parameter search for the TDCA-network, this measurement is no longer required to be differentiable and can be a discrete value by leveraging the benefits of evolution strategies. In experiments, we compared the performance of the top-down network-based optimization with traditional methods across various datasets, network architectures, and different learning paradigms. TDCA-networks can learn faster than traditional methods on non-convex function optimization, supervised learning and reinforcement learning. We also establish a brain-inspired credit-diffusion mechanism, a mechanism that can never be optimized by traditional learning paradigms, to further demonstrate the power of TDCA-network, making the framework more efficient and biologically plausible.

The TDCA framework represents a paradigm shift from traditional optimization methods by replacing explicit loss functions and backpropagation with a learnable credit assignment network. This approach offers several theoretical advantages:

- **Universal Approximation of Optimization Strategies**: As neural networks are universal function approximators, the TDCA-network can in principle encode any loss function and learning rule, providing a unified framework for optimization strategy discovery.
- **Emergence of Biologically Plausible Mechanisms**: Our framework naturally incorporates biological principles like heterosynaptic plasticity and credit diffusion, which are difficult to express within traditional gradient-based optimization.

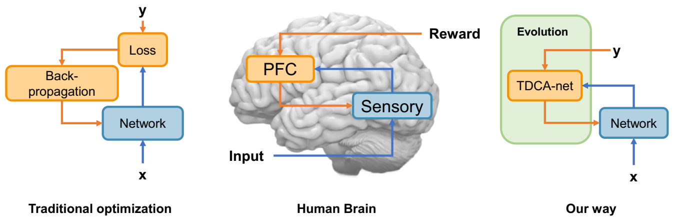


Fig. 1. Illustration of the different optimization paradigms. The TDCA-network mimics hierachical top-down modulation by providing high-level feedback signals, analogous to contextual modulation from higher-order brain regions. It can take the place of loss function and backpropagation in traditional optimization.

- **Escape from Local Optima**: The TDCA-network learns gradient fields that avoid local minima, suggesting it can discover optimization strategies beyond what traditional methods offer.

The rest of this paper is organized as follows. Section 2 describes related works. Section 3 introduces the proposed top-down network-based optimization method. Section 4 presents experimental results, and Section 6 provides concluding remarks.

2. Related works

2.1. Meta learning loss function

Meta-learning, often described as “learning to learn,” involves the enhancement of a learning algorithm through repeated learning episodes [33]. Loss function learning is an emerging subfield of meta-learning that focuses on leveraging past learning experiences to derive an optimal loss function directly from the data [19,20]. Some approaches were inspired by the idea of neural architecture search, where different combinations of mathematical symbols were explored to derive a new loss function [19,21–24]. Another type of approach involves searching for loss functions in the Taylor polynomial function space [34–36]. The most relevant works to ours utilize a feedforward neural network to represent the loss function [20,25]. However, since the primary goal of this line of work is not to address the biological plausibility of loss functions, their methods still rely on a manually designed loss function as a starting point. They then use backpropagation to meta-train the loss function network, which does not fully resolve the biological plausibility issues of both the loss function and backpropagation. In contrast, our work not only employs neural networks to represent the loss function but also leverages evolutionary algorithms to extend the dependency on symbolic and differentiable loss functions to metrics that only need to be quantifiable. This approach expands the search space, enabling the discovery of better optimization algorithms.

2.2. Biologically plausible backpropagation

The Backpropagation algorithm's biological implausibility has prompted significant research efforts for enhancement. Within BP, symmetric weight matrices from symbolic differentiation contradict biological brain characteristics [37]. Efforts to eliminate this symmetry abound [6,38–40]. Feedback Alignment (FA) replaces BP's feedback weight matrix with a random one, supporting error back-propagation in deep learning [6]. Direct Feedback Alignment (DFA) further streamlines the process by providing feedback from the output layer directly to each hidden neuron, achieving comparable performance to FA [38]. However, FA family, as a BP approximation, retains drawbacks, such as local optima susceptibility, and uses a random feedback matrix with extra constraints that lack biological plausibility. The BP algorithm also demands global weight information during error back-propagation, driving efforts to localize learning signals [41]. Therefore, equilibrium propagation uses an energy-based model to propagate gradient information between neighboring neurons [41]. Despite bypassing explicit global information collection, its requirement for local gradient information propagation necessitates more iterations for network learning, slowing adaptation. Likewise, the Difference Target-Propagation (DTP) algorithm back-predicts the current layer's output value based on the next layer's output and updates the weights accordingly [42]. Despite avoiding weight symmetry and global information, DTP introduces numerous extra parameters, deviating from the efficiency principles of the biological brain.

Recent advancements in algorithmic design have further enhanced the biological plausibility of optimization algorithms [17,40,43–46]. These studies exhibit two primary limitations. Firstly, they are all approximations of the Backpropagation algorithm, their performance cannot surpass that of BP theoretically. Secondly, they still focus solely on the biologically plausible aspect within BP. Our research approaches summarize the experiences from previous studies, pointing out that backpropagation is essentially a mapping from the loss value to the gradient vector. Therefore, it is quite natural to replace both BP and loss function with one top-down network, addressing its biological plausibility in a comprehensive manner.

2.3. Other bio-inspired top-down models

In biological brains, top-down projections are responsible for a range of high-level functions, such as attention, decision-making planning, and multi-modal sensory integration [28]. Consequently, the machine learning field has also made attempts in these functional directions. Traditionally, in machine learning, the attention mechanism is assumed to be the primary function of top-down projections. Therefore, a significant amount of work has focused on attention, such as using top-down signals to generate attention masks to enhance image classification and object localization performance in complex visual scenes [47,48]. Attention can also be used to achieve cross-modal information regulation [49] or multi-modal information integration [50]. Additionally, some models are also considered to be related to the flow of top-down information, such as Deep Boltzmann Machines, the Helmholtz Machine, and Transformers [51–53]. Overall, the top-down mechanism has been more frequently used to model attention mechanisms, with less work focused on expressing learning signals. Although some research has shown that transformer attention is equivalent to a one-step update over in-context learning data [54], no models have been specifically designed based on top-down projections to demonstrate that they can be used to replace the loss function and learning rule to optimize another neural network.

3. Top-down credit assignment framework

We now introduce a comprehensive networked optimization framework inspired by three brain principles. This framework comprises two distinct networks: a bottom-up network that executes tasks, and a top-down network that typically replaces both the loss function and back-propagation by directly generating modulation signals for the bottom-up network. We also introduce how individual neurons integrate information from the top-down network through a neuron learning rule, and how a single signal can regulate multiple neurons via a credit diffusion mechanism. The top-down network can operate independently or in conjunction with these two biological mechanisms.

3.1. Brain-inspired top-down network design

In the biological brain, top-down modulation refers to the influence of higher-order brain regions, such as the prefrontal cortex, on lower-level sensory or motor areas. Learning is closely related to top-down mechanisms because it helps shape and refine neural circuits involved in learning. Emerging evidence suggests that in the brain, high-level regions like the prefrontal cortex (PFC) and anterior cingulate cortex (ACC) act as a biological implementation of a loss function [2,55,56]. They evaluate task performance and outcomes, and then guide learning in lower-level sensory regions through top-down feedback pathways. For example, top-down projections in the visual system enable neurons to dynamically adjust their receptive fields based on task demands, enhancing the processing of task-relevant sensory inputs [4,29]. Similarly, in the hippocampus, prefrontal cortex projections can modulate neural activity to improve the encoding of spatial and contextual information [30,31]. These mechanisms collectively highlight the brain's ability to top-down modulate neural activity, directing learning signals to the appropriate neural circuits based on high-level goals or behavioral contexts.

Inspired by this biological paradigm, the TDCA-network, serving as a surrogate for conventional loss functions and backpropagation (as shown in Fig. 2), models a similar mechanism for deep learning optimization. Fig. 3 depicts the optimization framework with two loops: an inner loop optimizing the bottom-up network and an outer loop working on the TDCA-network. This design, inspired by meta-learning, allows the TDCA-network to learn a generalizable credit assignment function across tasks (outer loop) while simultaneously supporting task-specific optimization of the bottom-up network (inner loop).

In the inner loop, the bottom-up network $f(Input;\theta)$, controlled by parameter θ, produces outputs based on $Input$. The TDCA-network then generates parameter gradients $\Delta\theta$ for the bottom-up network, based on state $S(f(Input;\theta))$ (or $S(f)$) and the environment $Signal$. The state $S(f)$ typically includes information that is useful for optimization. We will explore its specific definition for different tasks. The TDCA-network as function $T(S(f), Signal;\beta)$ uses parameter β to control the learning strategy.

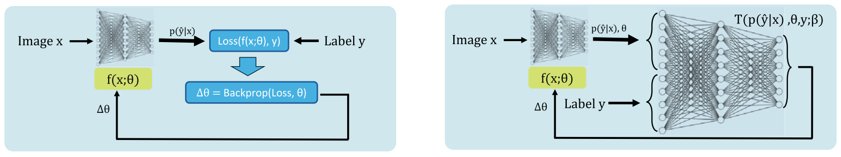
The outer loop, colored in orange in Fig. 3, optimizes parameter β using a hyper-optimizer, based on performance measurements calculated from $S(f)$ and $Signal$. The bottom-up network is re-initialized at the beginning of each inner loop; the final state $S(f)$ transmits to the outer loop after each run. Due to the complex dynamics of the TDCA-network, we utilize a black-box optimization algorithm as the hyper-optimizer to circumvent differentiating hyper-parameter β in this dynamic system. The Policy Gradients with Parameter-based Exploration (PGPE) [57], a black-box optimization technique, has demonstrated success in optimizing large-scale neural networks, matching the performance of leading optimization algorithms [58]. PGPE is a member of the large family of evolutionary algorithms, which are a subset of population-based optimization techniques inspired by the principles of natural selection and are biologically plausible [59,60]. The pseudocode is presented in Algorithm 1. An example of applying PGPE on supervised learning is detailed in Appendix A.

Despite matching state-of-the-art results, the PGPE algorithm has a higher computational cost [61]. Two key points need attention. First, PGPE can handle discrete fitness functions, but continuous functions provide richer information and require fewer sampling points. Second, as the number of parameters increases, so does the optimization difficulty, the number of sampling points, and computational power. To enhance practicality and reduce power requirements, we introduce two more biologically plausible mechanisms in the following subsections.

3.2. Local neuron-level learning rule design

Heterosynaptic plasticity is a form of synaptic plasticity in which the strengthening of a synapse occurs as a result of activity at neighboring synapses, rather than direct activity at the synapse itself [62, 63]. This phenomenon contrasts with homosynaptic plasticity, where changes in synaptic strength are driven by activity at the same synapse. Heterosynaptic plasticity highlights the influence of local synaptic networks and intersynaptic interactions in modulating neuronal adaptation. [64–67].

Due to the heterosynaptic plasticity nature of neurons, we can treat neurons as individual entities and regulate them using one signal. For a single neuron with input X and output $\hat{y}$ in of the bottom-up network (Fig. 4). We assume that each input weight will determine its update amount based on the neuron's output and the regulatory signal. In this case, we refer to the Hebbian learning rule, the update gradient for



(a) Backpropagation (b) Top-down network

Fig. 2. Example of training paradigm in classification task. (a) The traditional paradigm based on loss function and backpropagation. (b) Top-down networks replace the combination of the loss function and backpropagation. This approach maximizes the biological plausibility of training the bottom-up network.

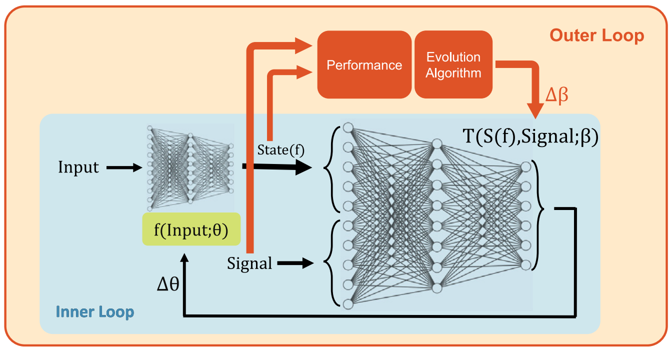


Fig. 3. Schematic diagram of the top-down learning framework. There are two neural networks in the framework. The bottom-up network f is responsible for handling the task; the top-down network T optimizes the bottom-up network by integrating the state information $S(f)$ of the bottom-up network and the external reward information $Signal$ from the environment. The framework consists of two parts: (1) the inner loop optimizes the parameter θ of the bottom-up network. (2) the outer loop uses an evolutionary algorithm to optimize the parameters β of the top-down network.

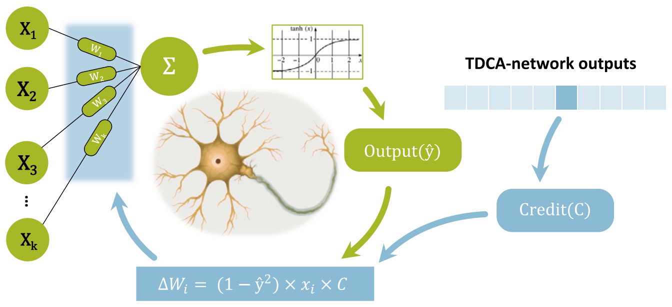


Fig. 4. Learning rule of a single neuron. Each neuron's weight is updated according to its own input and the neuron's output and modulated by a scalar credit C from the TDCA-network. The credit is generated per neuron by the TDCA-network.

each weight is calculated by multiplying the corresponding input x_i and Credit C. Given that our neurons use Tanh as the activation function, to rule out the effects of this activation function inside the neuron, the gradient incorporates Tanh's derivative as a coefficient. The synaptic weights are then updated according to:

$$\Delta W_i = (1 - \hat{y}^2) * x_i * C \tag{1}$$

This local update rule can help reduce the output dimension of the TDCA-network. For a neuron that has m weight parameters, instead of controlling each parameter individually with the TDCA-network, we assign a single credit to adjust all m parameters of each neuron. As a result, the output dimension of the TDCA-network is reduced by a factor of m.

Algorithm 1 TDCA-network optimization.

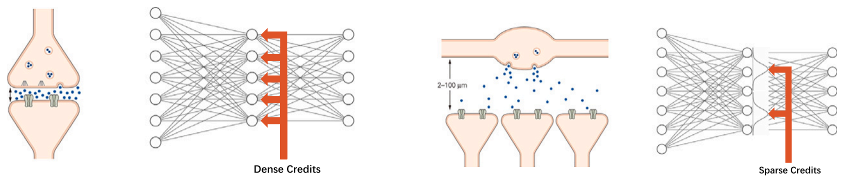
```
Input: dataset $D$, number of generations $G$, number of samples $N$, sampling variance $\sigma$, TDCA-net $T(\beta)$, bottom-up learning rate $lr_b$, TDCA-net learning rate $lr_t$
Initialize TDCA-network parameters $\beta$
for $i = 1$ to $G$ do
  Sample $\{\beta_n\}_{n=1}^{N}$, $\beta_n = \beta + \sigma\epsilon_n, \epsilon_n \sim \mathcal{N}(0, I)$
  Initialize bottom-up network parameters $\theta_1$
  for $n = 1$ to $N$ do
    for $j = 1$ to $M$ do
      $\Delta\theta_{batch} = 0$
      for each data point $(x, y) \in D$ do
        Get output of the bottom-up network: $\hat{y} = f(x;\theta)$
        Gather the state of the bottom-up network: $S(f) = [x, \hat{y}, \theta]$
        Generate the gradient: $\Delta\theta = T(S(f), y; \beta_n)$
        Accumulate the gradient $\Delta\theta_{batch} = \Delta\theta_{batch} + \Delta\theta$
      end for
      Update the bottom-up network: $\theta_{j+1} \leftarrow \theta_j + lr_b * \Delta\theta_{batch}$
    end for
    Evaluate the bottom-up network, for example:
      $$F(\beta_n) = \sum_{D}^{x,y} Accuracy(y, f(x;\theta_M))$$
  end for
  Estimate the gradient according to PGPE algorithm:
    $$\Delta\beta = \mathbb{E}_{\beta_n \sim \mathcal{N}(\beta,\sigma I)}\left[\nabla_{\beta_n} F(\beta_n)\right] = \frac{1}{\sigma}\mathbb{E}_{\epsilon_n \sim \mathcal{N}(0,I)}\epsilon_n F(\beta + \sigma\epsilon_n)$$
  Update the TDCA-network: $\beta \leftarrow \beta + lr_t * \Delta\beta$
end for
```

3.3. Credit diffusion mechanism

To further showcase capability of the TDCA-network and shrink its output dimension, we designed a biologically plausible mechanism inspired by how neurotransmitters regulate a group of neurons.

Neuroscience provides evidence of synapse crosstalk, and corresponding computational models exist [68]. Critical neurotransmitters in the central nervous system, glutamate and GABA, can function extra-synaptically [69]. These neurotransmitters, when released, can diffuse and modulate nearby neurons. Dopamine and serotonin, neurotransmitters vital for learning and memory, contact multiple synapses extensively, influencing proximate neurons [70,71]. Hence, a neuron's neurotransmitters commonly modulate neighboring neurons in a paracrine manner.

Borrowing from these paracrine mechanisms, we have integrated a credit diffusion mechanism into the TDCA framework. We define a neighboring structure to emulate signal spread among neurons within the same layer. As shown in Fig. 5(b), credits are initially sparsely allocated to neuron subsets, unlike the dense assignment in Fig. 5(a) and then diffused across the entire layer. A pre-set Gaussian kernel



(a) Dense credit-assignment (b) Sparse credit diffusion mechanism

Fig. 5. Schematic diagram illustrating dense credit assignment and sparse credit diffusion mechanisms for neurons. (a) Each neuron obtaining one credit value. (b) Credits are sparsely assigned to neurons, with groups of neurons sharing one credit value. This value decreases with distance, mimicking natural diffusion.

Table 1
Comparison of TDCA framework components with biological correlations.

Component of the framework	Function	Biological correlation
Top-Down Network	Implicitly encode traditional loss functions and learning methods (backpropagation) into a network, and directly generate parameter gradients for the bottom-up network.	Mimics top-down projections from higher-order cognitive areas (e.g., prefrontal cortex) to lower-level sensory or motor areas in the brain, which play a crucial role in learning.
Neuron Learning Rule	Each neuron updates its weights based on the credit signal provided by the TDCA-network.	Simulates heterosynaptic plasticity in the brain, where synaptic weight updates of a neuron are influenced by the activity of neighboring synapses, rather than solely by its own activity.
Credit Diffusion Mechanism	Credit signals diffuse among neurons, reducing the output dimension of the TDCA-network and improving computational efficiency.	Emulates paracrine signaling of neurotransmitters (e.g., dopamine, serotonin) in the brain, which can diffuse and modulate the activity of nearby neurons.
Evolutionary Algorithm	Population-based method optimizing the parameters of a top-down network. Encode the learning potential for specific tasks into the top-down network. Sometimes, the encoded learning strategies can generalize to unseen situations.	The more advanced an organism becomes in evolution, the stronger its learning ability. This potential to master complex tasks stems from the more diverse and sophisticated higher cognitive regions and top-down projections, which are shaped by the course of population-based biological evolution.

function controls the attenuation process, reflecting distance variations between neurons during diffusion. For a credit C_i assigned to neuron i, its diffusion to neuron j within the same group is:

$$C_j = C_i * e^{\frac{-(Coord(i)-Coord(j))^2}{\sigma^2}}, \tag{2}$$

where $Coord(\cdot)$ returns the neuron structure's coordinates and σ sets the credit diffusion distance. The definition of coordinate is

$$coord(i) = \begin{cases} i, Dim = 1 \\ [\left\lfloor \frac{i-1}{s} \right\rfloor, (i-1) \bmod s], Dim = 2 \end{cases} \tag{3}$$

Here, i represents the index of neuron, Dim is the dimension of the structure, and s denotes the size of square grid in the two-dimensional case. For example, in the one-dimensional case, 100 neurons can be arranged in a 1×100 structure, while in the two-dimensional case, they can be organized into a 10*10 grid structure.

This method arranges the neurons within the same layer into a structured grid, allowing a group of neighboring bottom-up neurons to be updated with a single credit, thereby reducing the number of credits needed to update the bottom-up network. This facilitates more efficient bottom-up model training with a smaller TDCA-network. If a group contains k neurons, the TDCA-network's credit dimension further reduces by a factor of k. We can either use a smaller TDCA-network with the same bottom-up network structure, or train larger bottom-up networks with the existing TDCA-network structure for improved performance.

Finally, we provide a detailed comparison of the framework's key components with their corresponding biological mechanisms in Table 1, to highlight how each component of the framework is inspired by and mimics specific biological processes in the brain.

3.4. Applications

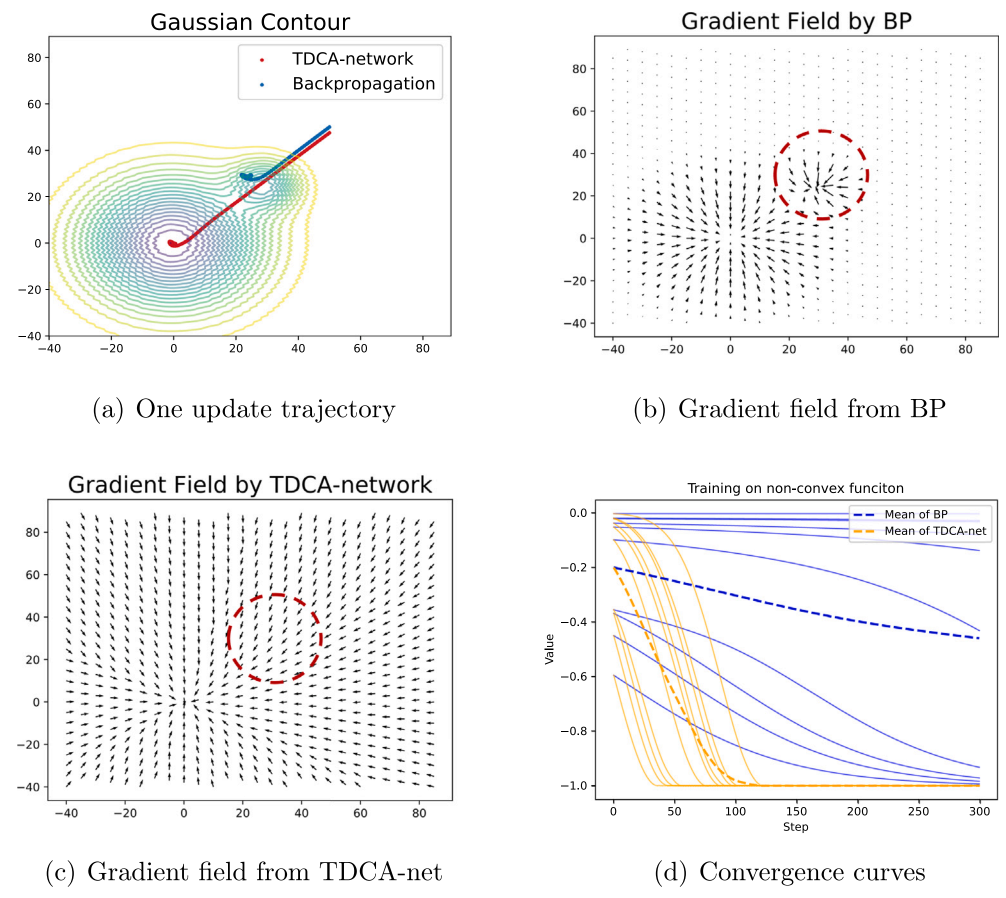
We will evaluate the effectiveness of our framework in the contexts of non-convex function optimization, supervised learning, and reinforcement learning. Our framework will adapt to the specific requirements of each learning paradigm accordingly. You can find related details in Appendix B.

4. Experiments

Our experimental design follows a proof-of-concept approach, focusing on demonstrating the fundamental capabilities of the TDCA framework across multiple learning paradigms. While larger-scale problems are valuable for future work, the current experiments on MNIST, Fashion-MNIST, and reinforcement learning tasks serve to establish:

1. The basic feasibility of replacing both loss functions and backpropagation with a learned network.
2. The framework's versatility across different learning paradigms.
3. The emergence of biologically plausible mechanisms not possible with traditional methods.
4. Competitive performance with specialized optimization algorithms.

We applied the TDCA optimization framework to three different optimization scenarios, to demonstrate that the TDCA-network can provide learning strategies that converge faster than traditional methods. We first tried to find better learning strategies in non-convex optimization problems. We compared the TDCA-network and the BP algorithm on optimizing Gaussian functions, and visualized their gradient fields to illustrate why the model controlled by TDCA-network converges faster. In a supervised learning context, we tried to find out better learning strategies based on the TDCA-network across various bottom-up network structures and datasets. Moreover, we showcased the capability of TDCA-network in handling the brain-inspired sparse credit diffusion mechanism, which in turn reduces computational complexity. Finally,



(a) One update trajectory
(b) Gradient field from BP
(c) Gradient field from TDCA-net
(d) Convergence curves

Fig. 6. Optimization on 2-dimensional mixed Gaussian function. (a) This panel compares the update trajectories of coordinate trained by BP (blue) and the TDCA-network (red) within the 2-dimensional Gaussian contour. (b) Under BP-based optimization, gradients are oriented towards both local and global minima. (c) With TDCA-network, gradients in the field lead directly to the global minimum, disregarding the local minimum. (d) Convergence curves of BP (blue) and TDCA-network (orange) in optimizing 10 randomly chosen initial points in 2D space. The solid lines correspond to different initialization points, while the dashed line represents the average performance across 10 optimization curves. Under the control of the TDCA-network, convergence is significantly faster than BP.

in reinforcement learning, we used the TDCA-network to find out faster learning strategies that outperform traditional reinforcement learning algorithms.

4.1. *Generating gradient field for non-convex Gaussian function*

To assess our top-down framework, we evaluate the TDCA-network's capability in solving non-convex optimization problems. Additionally, we explore its multitasking potential by simultaneously optimizing two distinct Gaussian functions. Detailed information on this application can be found in Appendix B.1.

4.1.1. *Non-convex Gaussian function*

As shown in Fig. 6, We create a two-dimensional non-convex optimization task using Gaussian function, consisting of a dominant Gaussian function with a smaller Gaussian function overlay. The optimization parameters are 2-dimensional coordinate values, with a global and a local minimum.

We compare the update trajectories of the BP algorithm and TDCA-network optimization for Gaussian functions in Fig. 6(a). The update trajectory (in blue) of BP algorithm stops after reaching the center of the small-scale Gaussian function, indicating that the parameters are trapped in a local minimum. However, the update trajectory (in red) by the TDCA-network moves to the center of the large-scale Gaussian function after crossing the center of the small-scale Gaussian function. This suggests that the TDCA-network's optimization strategy can avoid local optimum entrapment, achieving global optimization.

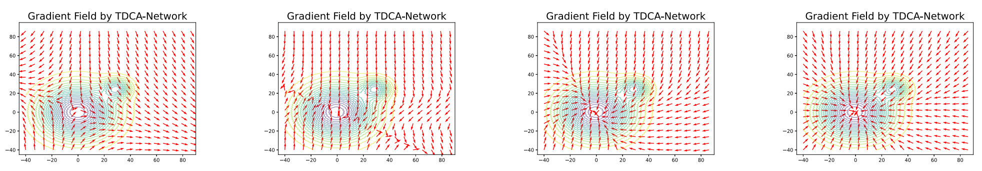
To fully analyze parameter updates guided by the TDCA-network, we present the gradient fields of different algorithms on the Gaussian surface. Fig. 6(b) shows the BP algorithm's update gradient field. Besides gradients pointing towards the global optimum, a group of negative feedback gradients exists at the local optimum (inside the red circle). These gradients trap parameters in this area, thus the BP algorithm fails to guide towards the global optimal solution. Fig. 6(c) displays the TDCA-network's update gradient, free of negative feedback gradients at the red circle. Unlike the BP algorithm, all gradients from the TDCA-network point towards the global optimum. Moreover, on much of the Gaussian surface in Fig. 6(b), gradient magnitudes approach zero, while the gradient in Fig. 6(c) maintains an adequate intensity, leading to faster convergence as depicted in Fig. 6(d).

To offer a clearer visual representation of the learning process of the TDCA-network, we depicted the gradient fields at various stages of the evolutionary algorithm in Fig. 7. It is evident that during random initialization, the gradient field is in a divergent state. However, as training advances, the gradient field swiftly adapts and ultimately converges towards the optimal value of the Gaussian function.

Overall, in optimizing 2-dimensional function, the gradients generated by TDCA-network have two advantages. Firstly, the TDCA-network's gradients do not saturate. Secondly, the TDCA-network can ignore the local optimum and accelerate parameter iteration towards the global optimum.

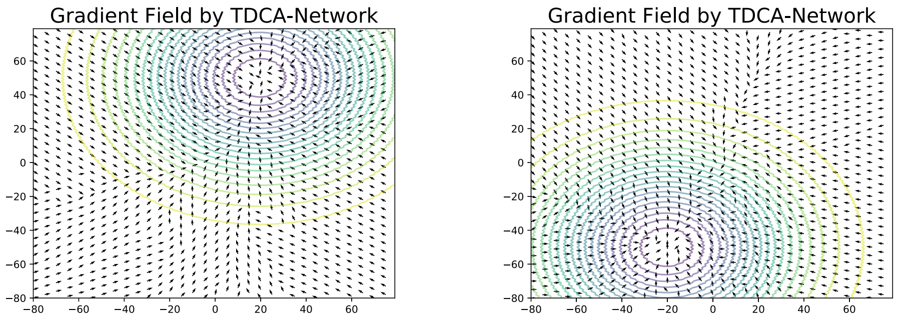
4.1.2. *Multi-Gaussian function*

To emphasize the proposed framework's capabilities, we test its optimization of two Gaussian functions simultaneously using a single



(a) Random Initialize (b) Step 3 (c) Step 5 (d) Step 20

Fig. 7. Evolving gradient field during TDCA-network training. The TDCA-network begins with random initialization, and as training progresses, the generated gradient field gradually aligns with the task landscape and stabilizes after 20 steps. Ultimately, we achieve a gradient field that optimally facilitates the task. At this stage, the TDCA-network is equivalent to a combination of a loss function and a learning rule capable of producing such an optimal gradient field.



(a) Gaussian 1 (b) Gaussian 2

Fig. 8. One TDCA-network optimizes two separate Gaussian functions. (a) and (b) are the contour maps of two distinct Gaussian functions and the corresponding gradient field generated by the TDCA-network. All gradient points to the optimum of their respective tasks.

TDCA-network, the framework is shown in Fig. 8. It displays contours of two distinct Gaussian functions, along with the TDCA-network's gradient fields at the corresponding positions. The TDCA-network generates different gradient fields pointing towards optimal values based on the task ID vector, demonstrating its multitasking optimization ability.

4.2. *Application in supervised learning*

We demonstrate the TDCA optimization framework's ability to find better learning strategies by examining its supervised learning performance in an image classification task. Detailed adaptation of the TDCA framework to supervised learning and experimental details can be found in Appendix B.2. The experiments include: 1) comparing TDCA-network's performance with Cross-Entropy (CE) loss function and BP method for optimizing the classification task across various network architectures and datasets, demonstrating its effectiveness in tasks that traditional methods can handle; 2) integrating the biologically-plausible sparse credit diffusion mechanism into the TDCA-network framework, showcasing its ability to optimize tasks that traditional methods cannot achieve; 3) analyzing the convergence process (in Section 4.2.5), computational complexity (in Section 4.2.6) and parameter update trajectory of the bottom-up neural network with both the TDCA framework and the BP algorithm in Appendix D. In the table, each cell presents the classification accuracy and standard deviation for both training and test sets.

4.2.1. *TDCA-network outperforms other biologically plausible methods*

In our early exploratory experiments (Appendix C), we found that in supervised learning, the TDCA-network exhibited better performance on MNIST compared to BP when Zero initialized. Additionally, the TDCA-network trained on MNIST could be transferred to optimize the classification of images in Fashion-MNIST.

To further validate our model, we assessed the TDCA-network's optimization of the bottom-up network on multiple datasets: MNIST, and Fashion-MNIST. Table 2 compares the performance of TDCA-network

Table 2
Classification performance on different datasets.

Method	MNIST	Fashion-MNIST
Kaiming BP	98.20 ± 0.15 %/96.16 ± 0.14 %	91.26 ± 0.38 %/87.19 ± 0.41 %
Kaiming FA	95.85 ± 0.12 %/94.79 ± 0.15 %	88.17 ± 0.16 %/85.79 ± 0.23 %
Kaiming DFA	95.95 ± 0.05 %/94.90 ± 0.10 %	88.13 ± 0.13 %/85.77 ± 0.20 %
Kaiming DTP	97.97 ± 0.17 %/96.47 ± 0.16 %	90.42 ± 0.19 %/87.20 ± 0.27 %
CE + Zero BP	99.04 ± 0.066 %/96.35 ± 0.15 %	92.33 ± 0.11 %/87.48 ± 0.12 %
ML^3	95.04 ± 0.12 %/94.52 ± 0.10 %	81.11 ± 3.26 %/80.20 ± 2.87 %
TDCA-net	99.08 ± 0 %/96.82 ± 0 %	93.36 ± 0.01 %/87.57 ± 0.01 %
Kaiming + SGD	91.13 ± 0.07 %/91.64 ± 0.07	83.25 ± 0.11 %/81.88 ± 0.19 %
Zero + SGD	11.24 ± 0 %/11.35 ± 0 %	43.11 ± 0.57 %/44.42 ± 0.56 %
TDCA-net + SGD	97.56 ± 0 %/96.47 ± 0 %	87.46 ± 0 %/85.67 ± 0 %

with BP, biologically plausible variants of BP (FA [6], DFA [38], DTP [42]) and ML^3 [20], which employs a neural network to meta-learn the loss function. The results indicate that the TDCA-network surpasses BP algorithm and its variants in fitting and generalization abilities. These results highlight the TDCA optimization framework's superiority over traditional loss function and back-propagation methods. The better performance of the TDCA-network compared to ML^3 demonstrates that co-searching for both loss functions and learning rules is more effective than searching solely for loss functions. The default optimizer for bottom-up network is Adam. Since the Adam optimizer inherently has a certain ability to avoid local optima, we also conducted comparative experiments using the most basic SGD optimizer. The results revealed that initialization becomes increasingly important for backpropagation, whereas TDCA-Net is relatively less affected by initialization. The optimization capabilities of TDCA-network become even more pronounced.

4.2.2. *TDCA-network optimizing deeper network architectures*

We verified the TDCA optimization framework on a deep fully connected neural network ("Deep FC") and a simple ResNet-6 network. The

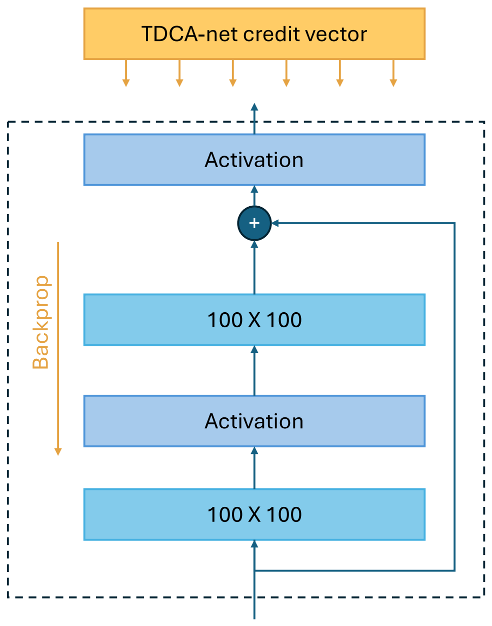


Fig. 9. The structure of the ResNet block, where each block contains two linear layers. When receiving credit from TDCA-Net, the block can perform backpropagation based on the gradient of its output layer. While this approach sacrifices some degree of biological plausibility, it offers a practical scale-up solution for the popular block-wise design used in large-scale models.

Deep FC has 4 layers with 100 neurons in each hidden layer. Since block-wise design has become increasingly popular in large-scale models. We conducted experiments on a smaller-scale ResNet. When backpropagation is still used within each block, TDCA-Net assigns credit to the neurons at the block output layer, which is then used for backpropagation within the block. The simplified block structure is illustrated in Fig. 9.

Table 3 shows the results. Zero BP underperforms Kaiming BP, suggesting that parameters initialized at the origin tend towards local optima, making the origin a poor initialization point for BP-based optimization. However, the “Top-down” initialized at the origin outperforms both Zero BP and other biologically plausible methods with Kaiming initialization. This finding suggests that even with advanced initialization methods, the algorithms are still limited by BP algorithm flaws and their final optimization performance is suboptimal. In contrast, the TDCA-network performs better, even with worse initial points, showing its ability to overcome the local optima challenge.

Table 4
Generalization performance across datasets.

Architecture	Method	MNIST-to-Fashion-MNIST	Fashion-MNIST-to-MNIST
MLP	ML^3	78.70 ± 4.63 %/78.04 ± 4.48 %	87.43 ± 4.61 %/87.20 ± 4.54 %
	TDCA-net	85.94 ± 0 %/83.91 ± 0 %	93.998 ± 0 %/93.21 ± 0 %
Deep FC	ML^3	58.30 ± 4.90 %/58.93 ± 7.06 %	40.79 ± 17.67 %/38.62 ± 16.13 %
	TDCA-net	81.25 ± 0 %/79.79 ± 0 %	76.42 ± 0 %/75.91 ± 0 %

As the complexity of the network structure increases, optimization becomes more challenging. At the same time, the gap between the TDCA-network and other biologically plausible methods continues to widen under Fashion-MNIST. The successful optimization of ResNet-6 suggests that the TDCA-Net method possesses the capability to handle more challenging optimization scenarios, a hypothesis we aim to investigate further in future studies involving more complex optimization problems.

4.2.3. *Transferability of well-trained TDCA-network*

We tested the TDCA-network’s transferability on individual datasets (MNIST, Fashion-MNIST). Each TDCA-network was evolved using the PGPE algorithm on the mentioned datasets individually. Table 4 presents the performance of a TDCA-network transferred to an unseen dataset. TDCA-networks demonstrate generalization abilities across datasets. And mean performance improves with higher dataset difficulty in Fashion-MNIST. This trend suggests that evolving TDCA-networks on more complex, multiple training datasets enhances the TDCA-network’s optimization ability.

We also conducted experiments by splitting the MNIST dataset into two subsets: digits 0–4 and digits 5–9. At each PGPE step, we randomly sampled 3 classes from the 0–4 subset to train the TDCA-network, and then evaluated its generalization performance on the 5–9 subset. During testing, we averaged the results over 10 randomly sampled tasks, as summarized in Table 5. The TDCA network meta-trained on digits 0–4 generalizes well to unseen digits 5–9, achieving generalization performance comparable to that of backpropagation.

4.2.4. *TDCA-network handle credit diffusion mechanism*

We evaluated the credit diffusion mechanism, which BP can never utilize, within the TDCA framework using MNIST, and Fashion-MNIST datasets. The given three-layer network structure constrained us to implement credit diffusion solely on the hidden layer. We defined a neighboring structure for the hidden layer neurons to implement credit diffusion, exploring a one-dimensional line and a two-dimensional grid structure, with each neuron having two and four neighbors, respectively. We set the hidden layer’s “sparsity”—the ratio of credit to neuron

Table 3
Classification performance on deeper network structure.

Model	Method	MNIST	Fashion-MNIST
Deep FC	Kaiming BP	99.83 ± 0.05 %/96.78 ± 0.20 %	91.98 ± 0.66 %/86.75 ± 0.49 %
	Kaiming DTP	98.65 ± 0.05 %/96.49 ± 0.08 %	90.64 ± 0.20 %/87.29 ± 0.19 %
	Kaiming FA	$94.93 \pm 0.08/94.22 \pm 0.03$ %	86.29 ± 0.34 %/84.44 ± 0.43 %
	Kaiming DFA	96.09 ± 0.11 %/94.89 ± 0.12 %	88.44 ± 0.19 %/85.85 ± 0.11 %
	Zero BP	98.97 ± 0.15 %/95.96 ± 0.21 %	90.51 ± 0.58 %/86.00 ± 0.39 %
	ML^3	93.05 ± 1.03 %/92.96 ± 0.46 %	71.04 ± 6.03 %/67.40 ± 7.47 %
	TDCA-net	99.06 ± 0 %/96.71 ± 0 %	93.27 ± 0 %/87.71 ± 0 %
	Kaiming BP + SGD	91.10 ± 0.10 %/91.55 ± 0.08 %	$82.60 \pm 0.15/81.23 \pm 0.22$ %
	Zero BP + SGD	11.24 ± 0 %/11.35 ± 0 %	10 ± 0 %/10 ± 0 %
	TDCA-net + SGD	97.3 ± 0 %/95.99 ± 0 %	88.27 ± 0 %/86.33 ± 0 %
ResNet-6	Kaiming BP	99.95 ± 0.02 %/96.60 ± 0.07 %	92.03 ± 1.0 %/87.13 ± 0.57 %
	Kaiming TDCA-net	99.39 ± 0.17 %/95.90 ± 0.08 %	93.45 ± 0.42 %/87.04 ± 0.07 %
	Kaiming BP + SGD	92.10 ± 0.10 %/92.35 ± 0.18 %	83.55 ± 0.20 %/82.06 ± 0.22 %
	Kaiming TDCA-net + SGD	96.52 ± 0.35 %/95.39 ± 0.19 %	86.39 ± 0.48 %/84.53 ± 0.33 %

Table 5
Generalization performance across digits.

Architecture	Method	Meta-Trainig (digit 0–4)	Meta-Testing (digit 5–9)
MLP	BP	98.72 ± 0.63 %/98.98 ± 0.71 %	97.97 ± 0.63 %/97.41 ± 0.71 %
	TDCA-net, meta-batch = 1	98.76 ± 0.69 %/98.89 ± 0.86 %	97.80 ± 1.04 %/96.96 ± 1.2 %
	TDCA-net, meta-batch = 4	99.17 ± 0.41 %/99.24 ± 0.53 %	98.55 ± 0.71 %/97.79 ± 0.72 %

Table 6
Performance of TDCA-network with credit diffusion mechanism (credit-to-neuron).

Dataset	100-to-100	10-to-100	100-to-1000	36-to-900	6*6-to-30*30
Sparseness	1.0	0.1	0.1	0.04	0.04
MNIST	99.08 % / 96.82 %	99.20 % / 96.96 %	99.67 % / 96.50 %	98.01 % / 96.51 %	99.87 % / 96.45 %
Fashion-MNIST	93.37 % / 87.56 %	92.80 % / 87.51 %	92.81 % / 87.59 %	34.87 % / 36.23 %	93.80 % / 87.86 %

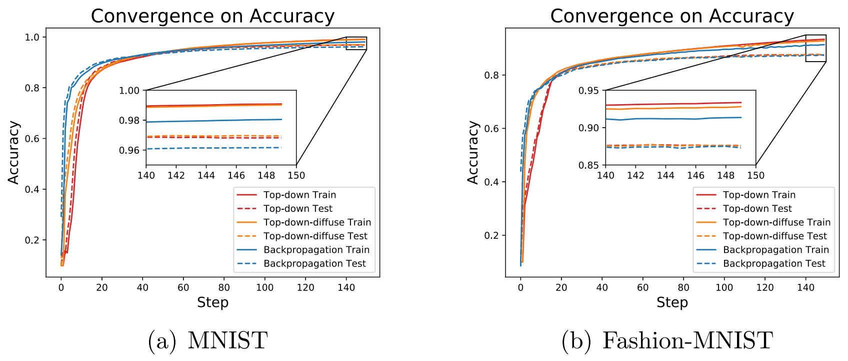


Fig. 10. Convergence curves of the bottom-up network across various datasets. The bottom-up network trained by Zero-initialized back-propagation (blue line) exhibits rapid convergence but falls short in final performance. In contrast, the top-down line (red) converges more slowly but yields superior performance in the final, while the top-down diffuse line (orange) shows intermediate characteristics.

count—from 0.1 (10/100, 100/1000) to 0.04 (36/900), using a σ of 5 for the Gaussian kernel.

As Table 6 shows, "10-to-100" performs similarly to that of "100-to-100" which suggests that sparse credit diffusion mechanism can achieve credit sparsity with minimal performance loss. Moreover, "100-to-1000" outperforms "100-to-100" indicating that increasing the bottom-up network size while keeping the credit count constant enhances performance. Therefore, the TDCA framework shows promise for controlling large bottom-up network updates using smaller TDCA-networks.

Even extreme credit sparsity (0.04) has little effect on MNIST ("36-to-900" in Table 6), demonstrating this frameworks' robust adaptability and indicating that dense gradient information isn't essential for optimization.

We also tested a two-dimensional grid structure in the hidden layer, with a 6*6 credit structure controlling a 30*30 neuron structure (6*6-to-30*30). The results show better performance than 36-to-900 and even the 100-to-1000 configuration with denser credit and more neurons. This suggests that higher-dimensional credit diffusion can further enhance the TDCA-network's performance.

4.2.5. *Convergence analysis*

Fig. 10 shows the convergence curves of the bottom-up network. Our findings show that the bottom-up network, trained by traditional BP, converges faster than the TDCA-network initially (before 40 steps), but the final performance turns out to be worse. This indicates that backpropagation quickly falls into the local optimum near the initialization. In contrast, the TDCA-network takes a long-term perspective, leading to superior performance on both the training and testing sets. These results show that the TDCA-network might bypass the local optimum early on, providing a more effective update strategy than BP.

4.2.6. *Complexity analysis*

Our study quantitatively evaluates the reduction in parameters and computational complexity of the BP, DTP, FA, DFA and three TDCA-network-based optimization methods: credit parameters, credit neurons, and credit groups. For bottom-up networks, the credit parameters method often struggles due to the excessive number of parameters. Hence, we only provide the TDCA-network parameter count for comparison without evaluating its computational complexity. In Table 7, each row corresponds to an optimization method, showcasing parameter quantities and computational complexities. Computational complexity is expressed as floating-point operations (FLOPs).

We analyzed three-layer networks with a hidden layer of either 100 or 900 neurons. In the 100-neuron bottom-up network, we observed a decrease in the TDCA-network's size, parameter quantity, and computational complexity when viewing neurons as a whole and using a credit diffusion mechanism. This suggests they can effectively simplify the TDCA-network. In the 900-neuron bottom-up network, the credit group's computational complexity dropped to nearly one-sixth of the BP algorithm. As the number of neurons increases 9-fold, the computational complexity of the BP algorithm increases linearly (×9) with the neuron count, but the increase for "credit groups" can be minimal (×1.4). Furthermore, our comparisons on ResNet-6 (with two hidden blocks) reveal that, TDCA-network requires only half the computational cost of standard backpropagation. Therefore, the computational complexity of the TDCA-network can be much lower than that of the BP algorithm for large-scale computations, demonstrating its advantage in efficiency.

While the current TDCA-network implementation demonstrates higher computational requirements during meta-training, this cost should be considered in the context of the framework's unique capabilities:

Table 7
Complexity of different methods for optimizing various bottom-up networks.

	Settings	MNIST/FashionMNIST	
		Param	MFLOPs
100-hid-MLP	BP/DTP	0	240
	FA/DFA	1000	240
	Credit each parameter	4,892,794	-
	Credit each neuron	4970	576
	Credit each group	2180	252
900-hid-MLP	BP/DTP	0	2160
	FA/DFA	9000	2160
	Credit each group	2986	345.6
ResNet-6	BP	0	2640
	Credit each neuron	11,170	1296

[1]DTP requires derivative calculation for the last hidden layer to converge. This means DTP back-propagates one layer fewer parameters than FA and DFA. For MLP with a single hidden layer, DTP has 0 parameters. For ResNet-6, TDCA-Net directly feeds the gradient to the block output layers. Therefore, we only account for the FLOPs of backpropagation between blocks to ensure a fair evaluation.

Table 8
Comparison of optimizing reinforcement tasks.

Method	CartPole-v1	Pendulum-v0	BipedalWalker-v3
Reward Range	[0,200]	[−3254.7,0]	[−400, 300]
Policy Gradient	199 (success)	–	–
TD3	–	−647 (success)	−88 (False)
TDCA-net	200 (success)	−119 (success)	288.4 (success)

- **One-time Investment**: The TDCA-network training is a one-time cost that produces a reusable optimization strategy.
- **Credit Diffusion Efficiency**: As shown in Section 4.2.4, the credit diffusion mechanism significantly reduces computational requirements while maintaining performance.
- **Hardware Optimization Potential**: The localized nature of TDCA-network updates makes the framework amenable for parallelization and hardware acceleration.

The computational benefits become particularly apparent when considering tasks where traditional methods struggle, such as optimization with the credit diffusion mechanism which backpropagation cannot naturally express.

4.3. Application in reinforcement learning

The TDCA optimization framework's adaptability is highlighted through its use in various reinforcement learning tasks. We replace

Table 9
Multi agents controlled by the same TDCA-network.

Measurement	Button-press-wall-v2	Dial-turn-v2	Door-close-v2
Training (3 init)	2581.9 (0 %)	1497.5 (100 %)	3451.2 (100 %)
Test (10 init)	1932.2 (0 %)	1493.2 (80 %)	3385.5 (90 %)

the conventional gradient sampling from agent-generated actions and back-propagation with a TDCA-network (Detailed information on this application and experimental settings can be found in Appendix B.3).

Traditional reinforcement learning algorithms fall into two categories: discrete and continuous action. Numerous algorithms cater to these, such as the Policy Gradient (PG) for discrete action, and the Twin Delayed Deep Deterministic Policy Gradient (TD3) for continuous action [72,73]. Table 8 compares the performance of the TDCA-network with established algorithms. It reveals the TDCA-network's success and efficiency, scoring close to the maximum across various tasks. Two key advantages of the TDCA framework emerge. It optimizes both discrete and continuous action agents, overlooking the action data type differences, indicating its broad applicability. Moreover, the learning method discovered by the TDCA-network converges very quickly. Despite the challenges posed by BipedalWalker in continuous action reinforcement learning, the TDCA-network successfully completes the task and achieves near-optimal scores after only 200 iterations. In contrast, the TD3 algorithm does not converge during this time.

We compare the TDCA-network with the advanced TD3 algorithm and plot the convergence process of the algorithms in Fig. 11, the TDCA-network achieves faster convergence. In the BipedalWalker task, the TDCA-network can converge with an MLP agent network that has a single hidden layer of 40 neurons, while the TD3 cannot. The TD3 requires larger model capacity, a fully connected network with two hidden layers, each with a width of 400, to achieve slow convergence.

We also tested one TDCA-network's ability to optimize multiple reinforcement learning tasks concurrently using the Meta-World environment. Experiment details can be found in Appendix B.3.1. Table 9 shows the score and corresponding success rate, where the TDCA-network can successfully learn to optimize nine different tasks (three task types, each with three different initial states), and can generalize to 30 unseen tasks (three types * ten initializations) with different target positions effectively. This demonstrates the TDCA-network's ability to develop distinct optimization strategies per task category, and to generalize within the same category.

5. Discussion

Technical Features: Compared with previous brain-inspired model studies, the TDCA-network replaces the entire optimization process with a network, further enhancing the biological plausibility of the model. Previous studies have mostly focused on how to make the

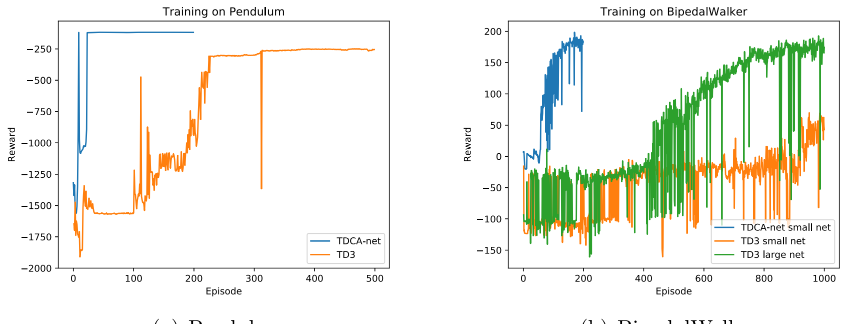


Fig. 11. Comparison of the TDCA-network and TD3 in optimizing agent networks. The episode length is set 200 for Pendulum and 500 for BipedalWalker. The TDCA-network achieves faster convergence than TD3 and can successfully converge on smaller networks where TD3 cannot.

backpropagation algorithm more biologically plausible, while this paper starts from a higher level, simulating the top-down modulation mechanism in the brain to directly replace the traditional loss function and backpropagation with a network, making the entire optimization process closer to the learning methods of the biological brain.

Technical Innovation: The technical innovation of this paper lies in the first use of a top-down network to simultaneously replace the loss function gradient calculation and other optimization tools. This innovation not only breaks the limitations of traditional optimization methods but also provides an effective means for discovering new optimization methods. By transforming the optimization process into a network parameter search problem, the TDCA-network can explore more optimal optimization strategies in a larger search space, bringing a new perspective and method to the field of machine learning.

Theoretical Innovation: From a theoretical perspective, this paper demonstrates the feasibility of using a top-down network for meta-learning the entire optimization process. This not only provides a new theoretical basis for optimization algorithm research but also indirectly explains the potential learning mechanisms of the brain. The top-down structure of the brain is likely the source of its efficient learning ability, and this study provides new clues and evidence for understanding the learning mechanisms of the brain.

Application Prospects: The application prospects of the TDCA-network are very broad. First, it can be used to design and discover new machine learning algorithms. By simulating the brain's learning methods, the TDCA-network can explore more efficient and faster optimization strategies, injecting new vitality into the development of the machine learning field. Second, the TDCA-network can also be based on neurophysiological data to model the dynamic learning processes of the brain. This will help us better understand the learning mechanisms of the brain and provide new ideas and methods for the interdisciplinary research of neuroscience and artificial intelligence.

Challenges: One potential concern is the adaptability of the TDCA-network to different network sizes and architectures. While the TDCA-network has shown promise in optimizing smaller networks, its performance in very large networks, such as deep convolutional networks (VGG, ResNet) or transformers, remains to be fully explored. The increased complexity of these networks could challenge the TDCA-network's ability to effectively generate credit signals and maintain high accuracy.

Another issue is that although we observed faster convergence and improved training set performance during the experiment, the model's generalization performance actually decreased as the bottom-up network complexity increased. The current method does not take the model's generalization performance into account. Perhaps in the future, we can set the validation set performance as the fitness for the evolutionary algorithm, or use other regularization methods or optimizers to reduce the risk of overfitting in the TDCA-network [74].

6. Conclusion

This study presents the TDCA-network, an innovative optimization approach inspired by the brain's top-down modulation processes. Unlike traditional methods that rely on loss functions and backpropagation, the TDCA-network employs a biologically plausible learning strategy that significantly enhances optimization efficiency and performance across various learning paradigms. This framework not only increases biological plausibility but also serves as a powerful tool for discovering new optimization strategies through network-based parameter searches.

The TDCA-network excels in optimizing non-convex functions, supervised learning, and reinforcement learning, surpassing conventional methods in terms of convergence speed and performance. Its credit diffusion mechanism further underscores the TDCA-network's potential to discover new optimization strategies, while simultaneously reducing computational complexity and maintaining high performance, making it highly scalable for large-scale applications.

In conclusion, the TDCA-network offers a compelling alternative to traditional optimization methods, bridging the gap between neuroscience and artificial intelligence. It provides valuable insights into the brain's learning mechanisms and sets the stage for future research in developing more efficient and biologically plausible learning algorithms.

CRediT authorship contribution statement

Jianhui Chen: Writing – original draft, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Tianming Yang:** Supervision. **Cheng-Lin Liu:** Supervision, Conceptualization. **Zuoren Wang:** Writing – original draft, Conceptualization.

Code availability

Code available at: https://github.com/alexxchen/Top-down-credit-assignemnt-network

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been supported by the Strategic Priority Research Program of the Chinese Academy of Sciences Grant No. XDB1010201, National Science and Technology Innovation 2030 Major Program, No 2022ZD0205100.

Appendix A. Evolving parameters of TDCA-network in supervised learning

We apply the Policy Gradients with Parameter-based Exploration (PGPE) algorithm in this study to approximate gradient calculations via sampled fitness [75].

Regarding the PGPE algorithm, we define the Gaussian distribution from which we sample as:

$$p_\beta(\beta_i) = \mathcal{N}(\beta, \sigma I);\ \ \beta_i = \beta + \sigma\epsilon, \epsilon \sim \mathcal{N}(0, I), \tag{A.4}$$

where σ is the sampling variance. The PGPE algorithm draws N samples $\{\beta_i\}_{i=1}^N$ from the distribution $p_\beta(\beta_i)$ to estimate the gradient expectation at β.

We use the fitness function $F(\beta_i)$, based on the sample performance. In a classification task, for instance, gradients are generated for the bottom-up network to improve classification performance. Hence, the fitness function is defined as:

$$F(s) = Accuracy(f(x;\theta_M), y), \tag{A.5}$$

where subscript M denotes the iteration times of inner loop, and

$$\theta_{j+1} = \theta_j + T(S(f(Input;\theta_j)), y; \beta_i), j \in 1, 2, 3 \ldots, M \tag{A.6}$$

Taken together, the expectation of fitness on β is

$$\mathcal{F}(\beta) = \mathbb{E}_{\beta_i \sim p_\beta(\beta_i)} F(\beta_i) = \int_{\beta_i} F(\beta_i) p_\beta(\beta_i) d\beta_i. \tag{A.7}$$

Finally, the gradient of β is (please refer to [58] for more details)

$$\begin{aligned}\nabla_\beta \mathcal{F}(\beta) &= \nabla_\beta \mathbb{E}_{\beta_i \sim \mathcal{N}(\beta,\sigma I)} F(\beta_i) \\ &= \nabla_\beta \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} F(\beta + \sigma\epsilon) \\ &= \frac{1}{\sigma} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} \epsilon F(\beta + \sigma\epsilon).\end{aligned} \tag{A.8}$$

Appendix B. Application details on different learning paradigms

B.1. Non-convex function optimization

We adjust the framework to use the TDCA-network in a two-dimensional Gaussian function optimization (Fig. S1). Here, the inner loop's optimization objective is the Gaussian function $Gaussian(x, y)$, with x and y coordinates representing the optimization parameters. The Gaussian function's mean and variance are fixed. Without additional input, the Gaussian function's state is the current coordinate pair $[x, y]$. The TDCA-network generates the corresponding gradients Δx and Δy and seeks the Gaussian functions' extremum points. Concurrently, the outer loop's evolutionary algorithm updates the parameter β based on the Gaussian function's value after each inner loop.

The TDCA framework can also optimize two distinct Gaussian functions. We adapt the framework to optimize multiple Gaussian functions simultaneously (see Fig. S2). We treat two Gaussian functions with different means and variances as separate tasks, each with a unique one-hot ID. We assign Gaussian 1 with task ID vector $[1, 0]$ and Gaussian 2 with task ID vector $[0, 1]$. When TDCA-network optimizes one of the Gaussian functions, we input the task ID along with its current coordinate values. The TDCA-network should generate different gradients based on the task ID.

In the inner loop, we use the SGD optimizer with $lr_b = 0.1$. For each iteration, 10 initial points are randomly sampled from the parameter space, and the final average of these 10 points is used as the fitness for the evolutionary algorithm. The TDCA-network is a multi-layer perceptron (MLP) with a structure of 2–100-2. The PGPE uses the SGD optimizer with $lr_t = 0.05$, a sampling population size of $N = 501$, $\sigma = 5$ and evolves $G = 100$ generations.

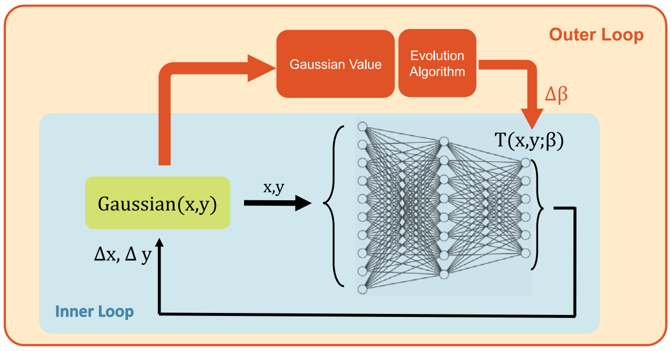


Fig. S1. We adapted the top-down learning framework for non-convex Gaussian optimization. Coordinate parameters to be optimized, x and y, are fed into the TDCA-network, which then produces a gradient used to update these parameters. Simultaneously, the TDCA-network is optimized to generate improved gradients for each coordinate within the Gaussian parameter space.

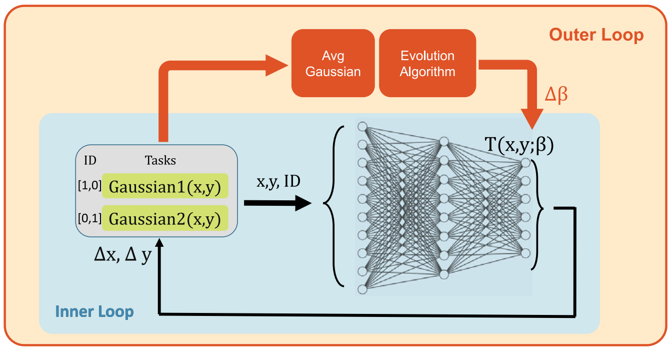


Fig. S2. Multiple Gaussian function-optimization tasks, featuring a single TDCA-network and two distinct Gaussian functions. Depending on the task number ID, the TDCA-network can independently optimize the parameters of Gaussian1 and Gaussian2. Fitness is the average value of the two Gaussian functions.

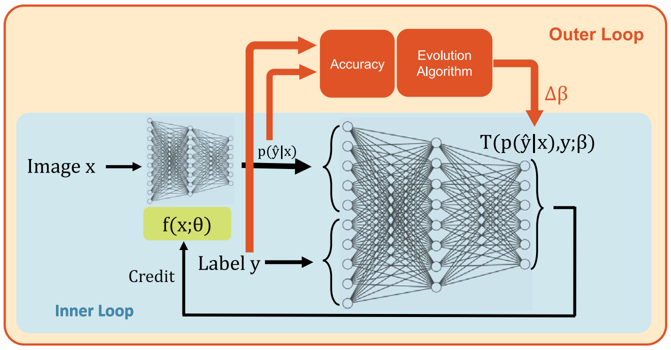


Fig. S3. The top-down credit assignment learning framework for classification tasks in supervised learning. The framework uses an image x as the input for the feed-forward network, which in turn generates probabilities attributing the image to each category. The TDCA-network generates a credit based on the output probabilities $p(\hat{y}|x)$ and the image category label y, which is used for parameter updates. In the outer loop, the β hyperparameter optimizer evaluates and updates the TDCA-network's optimization strategy based on the classification accuracy or cross-entropy after each inner loop run.

B.2. Supervised learning

We use the TDCA framework to optimize feed-forward networks for image classification tasks, as shown in Fig. S3. The feed-forward (bottom-up) network $f(x; \theta)$ predicts the class label probability distribution $p(\hat{y}|x)$ for an input image x. The TDCA-network calculates the parameter gradient from the feed-forward network's state $S(f)$ and label y. The traditional back-propagation can be as

$$\Delta\theta = Backpropagation(Loss(f(x; \theta), y), x, \theta). \tag{B.9}$$

The variables are input x, label y and parameters θ. Thus, we define the feed-forward network state, $S(f) = [x, \theta]$. Given the current state $S(f)$ and label y, the TDCA-network's credit assignment strategy is determined by parameter β.

To optimize large-scale feed-forward networks, it is crucial to minimize computational complexity and enhance efficiency. We can achieve this by employing the learning rule of single neuron depicted in Fig. 4 and the credit diffusion mechanism illustrated in Fig. 5, which helps decrease the output dimension of the TDCA-network.

Moreover, we can further reduce the dimension of inputs to the TDCA-network. The input to the TDCA-network is the state of the feed-forward network $S(f)$. If we define this state using the full dimension of x and θ, the TDCA-network becomes excessively large. Therefore, it is important to define the state effectively to capture as much information about the feed-forward network as possible while maintaining a low dimensionality. This involves reducing the dimensions of x and θ. Conveniently, the feed-forward network output $f(x; \theta)$ inherently reduces dimensionality by projecting these two parameters into the probability distribution of each class. Consequently, we can use the feed-forward network output as the input to the TDCA-network, significantly reducing its input dimension.

B.2.1. Experimental details

We build a fully connected bottom-up network for classification tasks and a fully connected TDCA-network for credit signal generation. The TDCA-network's ability to train a three-layer bottom-up network with 100 hidden units is assessed using MNIST, and Fashion-MNIST [76,77]. The default bottom-up network structure remains consistent 784–100-10

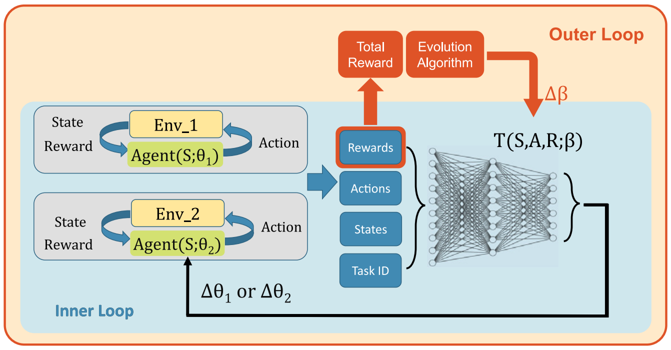


Fig. S4. The top-down credit assignment learning framework for reinforcement learning. The framework comprises two neural network agents (each corresponding to different environments) and a shared TDCA-network. The agents interact with their respective environments, accumulating rewards until the tasks conclude. The TDCA-network generates the gradients to update parameters θ for both agents, distinguishing the gradients for each agent by the environment ID. In the outer loop, the hyperparameter optimizer evaluates the performance of the TDCA-network by averaging the rewards of the two agents after each inner loop run, then adjusts the hyperparameter β. The optimization of a single agent is simply a special case of optimizing multiple agents.

for MNIST and Fashion-MNIST. The TDCA-network uses a 20–30-30–110 neuron configuration. The TDCA-network input, generating 110 credits (10 for outputs and 100 for hidden neurons in the bottom-up network), combines the bottom-up network's output with a one-hot label from the dataset. To decrease the necessary sample number of PGPE algorithm, we use cross-entropy as the fitness function for the outer loop.

In the inner loop, we use the SGD optimizer with a learning rate of 0.1 or Adam optimizer with learning rate of 0.01. The PGPE uses the SGD optimizer with a learning rate of $lr_t = 0.1$, a sampling population size of $N = 501$, and $\sigma = 0.1$. We evolve 5000 generations of the TDCA-network parameters using the PGPE algorithm. Our exploratory experiment in Appendix C demonstrates that the batch size and the number of update steps in the inner loop significantly impact the final performance. To minimize the introduction of data noise, we set the batch size to 60,000. Meanwhile, to achieve satisfactory performance without excessive computational cost, we limit the number of inner loop steps to $M = 150$. In the comparative experiments, we perform five independent runs with different random initializations and report the mean accuracy and standard deviation for each optimization method.

B.3. Reinforcement learning

To further investigate the TDCA optimization framework's applicability, we apply it to reinforcement learning for better learning strategies. In traditional reinforcement learning, algorithms are typically more rigid than those in supervised learning, placing a greater emphasis on reward shaping, which can be considered a form of designing the loss function. Similarly, top-down networks can utilize parameter search to take the place of manual design process.

Fig. S4 illustrates a bottom-up network $Agent(S;\theta)$, functioning as an agent, interacting directly with the reinforcement learning environment. The agent produces actions from the current state of the environment S, leading to state updates and corresponding rewards. The environment resets at the start and reaches a terminal state after a predetermined number of interactions, which we define as an episode. After each episode, the agent's parameters θ are updated by TDCA-network to maximize future rewards. This network generates the parameter update gradients $\Delta\theta$, using information from the current episode including environment states, actions, and rewards. We expect the agent to adapt to the environment through iterative inner loop processes, achieving maximum reward. When one TDCA-network is used to optimize multiple agents performing different reinforcement learning tasks, we assign unique IDs for each task, allowing the TDCA-network to generate update gradients for each task based on its ID.

In the outer loop, we collect the episode reward of the agent at the end of the inner loop to evaluate the TDCA-network. Then evolutionary algorithm estimates $\Delta\beta$ and updates the TDCA-network. With the bottom-up network's small size, additional methods to reduce the TDCA-network's parameter count are not needed.

B.3.1. Experiment details

We conducted experiments in the Gym reinforcement learning simulation environment [78], testing the TDCA-framework on CartPole, Pendulum, and BipedalWalker games in ascending difficulty order (Fig. S5). The TDCA-network optimizes agent parameters, aiming to successfully complete the Gym games. An "agent" in deep learning is a neural network that produces actions for the simulation environment based on the current state, receiving rewards and a new state in return. This interaction generates sequential information, informing our decision to design an RNN structure for the TDCA-network. We define the game's duration from start to finish as an "episode". The TDCA-network receives the state, action, and reward vectors from each episode and generates the credit for each parameter, which updates the agent.

For inner loop on Gym tasks, we set the agent's update iterations to $M = 200$ steps, BipedalWalker uses agent network:24–40-4 (Zero initialization), Adam with lr 0.01; CartPole uses agent network:4–100-1, Adam with lr 0.001; Pendulum uses agent network:3–100-1, Adam with lr 0.1. For out loop, the TDCA-network is RNN(29-50–50)-1164 for BipedalWalker, RNN(6-50–50)-601 for CartPole, and RNN(5-50–50)-501 for Pendulum. The population $N = 301$ for CartPole, $N = 501$ for Pendulum, $N = 3001$ for BipedalWalker, $\sigma = 0.1$, SGD optimizer with $lr_t = 0.01$, evolves over $G = 3000$ generations.

To optimize multiple reinforcement learning tasks, we choose the meta-world environment (Fig. S6 [79]), controlling a robotic arm performing various tasks. With constant degrees of freedom in the robotic arm tasks, we train agents with identical network structures of 39–16-16–16-4 on different tasks, using Adam optimizer with $lr_b = 0.001$,

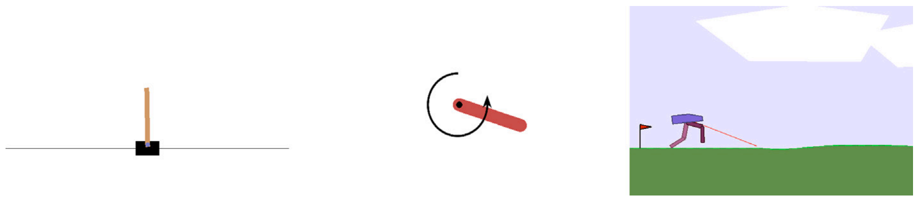
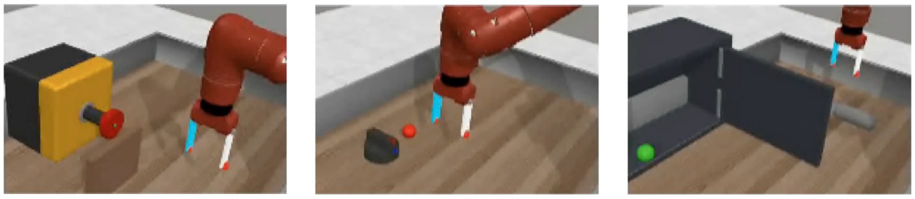


Fig. S5. The reinforcement learning environment of the Gym platform. We evaluated three Gym reinforcement environments of varying difficulty levels (from easy to difficult): cart pole, pendulum, and bipedal walker. The cart pole game employs a discrete action space, while the pendulum and bipedal walker utilize continuous action spaces.



(a) Button-press-wall (b) Dial-turn (c) Door-close

Fig. S6. The reinforcement learning environment of the Meta-World platform. We assess three types of reinforcement tasks related to robotic arm control: a) Button-press-wall, b) Dial-turn, c) Door-close.

update step $M = 200$. We optimize agents on different tasks with a single TDCA-network with structure of RNN(44-30–30)-980. The PGPE algorithm uses $\sigma = 0.01$, SGD optimizer with $lr_t = 0.1$, population $N = 899$, and updates over 1000 generations. We chose three task types for testing: Button-press-wall, Dial-turn, and Door-close. Each task has different initial states, including varying arm initializations and target positions. Tasks with different target positions are considered distinct but share the same paradigm. The TDCA-network updates various agents according to the task type ID in its inner loop. We used the average reward score as the fitness metric to estimate the magnitude of the TDCA-network updates' magnitude. To explain high reward cases without success ("Botton-press-wall-v2" column), We consulted the Meta-world environment's author, who confirmed this is common.

Appendix C. Exploratory experiment on supervised learning

We tested the TDCA-network's training efficiency for the bottom-up network using the MNIST dataset under various conditions, such as differing update steps and dataset sizes. We also transferred the TDCA-network, trained on the MNIST dataset, to the Fashion-MNIST dataset to evaluate its generalization ability. Table S1 shows the results. The first three rows represent MNIST dataset findings, and the rest outline the Fashion-MNIST dataset results. Each row corresponds to a unique initialization and optimization method for the bottom-up network. We used Kaiming Initialization ("Kaiming BP") and the BP method for network initialization and optimization, respectively. "Zero BP" represents all bottom-up network parameters initialized to 0 (origin initialization), optimized with the BP method. Owing to the lack of effective gradients at a parameter value of 0, initialization is actually restricted to within a 1e-16 range around the origin. "TDCA-net" signifies initializing all network parameters at 0, optimized by the TDCA-network. "TrFC"indicates the transfer of a fully-connected network trained on the MNIST dataset to the Fashion-MNIST dataset. "TrTDCA-net" denotes the transfer of the TDCA-network from the MNIST dataset for training a new bottom-up network on the Fashion-MNIST dataset. The three wider columns represent different dataset sizes and iteration numbers for updating bottom-up networks.

Table S2
Classification performance on CIFAR-10.

Method	MLP	Deep-FC
CE + Kaiming BP	52.21 ± 0.66 %/38.99 ± 0.51 %	58.17 ± 0.99 %/38.07 ± 0.63 %
CE + Kaiming FA	46.54 ± 0.26 %/39.10 ± 0.23 %	44.75 ± 0.80 %/39.08 ± 0.43 %
CE + Kaiming DFA	46.52 ± 0.62 %/39.32 ± 0.45 %	49.56 ± 0.59 %/41.47 ± 0.27 %
CE + Kaiming DTP	50.49 ± 0.29 %/40.69 ± 0.23 %	52.67 ± 0.47 %/42.94 ± 0.71 %
CE + Zero BP	60.83 ± 0.83 %/40.42 ± 0.41 %	48.34 ± 0.87/38.19 ± 0.57
ML^3	42.78 ± 0.88 %/41.24 ± 0.76 %	31.39 ± 6.39 %/31.73 ± 4.46 %
TDCA-net	59.81 ± 0.30 %/41.92 ± 0.19 %	58.76 ± 0.23 %/41.70 ± 0.14 %

We used the Kaiming initialization method as a positive control due to its widespread use and satisfactory performance. The MNIST dataset results suggest that Zero BP's performance matches the Kaiming Initialization method, indicating that advanced initialization methods may not always excel. However, Zero BP stagnates at a local optimal solution with a training set of 1000 samples. The TDCA-network method surpasses local optima, outperforming Zero BP and even the Kaiming Initialization method on smaller training sets.

We tested the TDCA-network's generalization capability by transferring it to the Fashion-MNIST dataset. Its performance matches the Zero BP algorithm on training sets of 10,000 and 60,000 samples and surpasses Zero BP on a training set of 1000 samples. Hence, without Fashion-MNIST fine-tuning, the transferred TDCA-network from the MNIST dataset (TrTDCA-net) optimizes the bottom-up network (TrTDCA-net vs. Zero BP) effectively. We confirmed this performance by using TrFC as a control. The results reveal that the TrTDCA-net's effect stems from the TDCA-network's generalization ability (TrTDCA-net vs. TrFC). In conclusion, the TDCA's update strategy, evolved on the MNIST dataset, can generalize across different datasets.

Table S1 reveals that larger training datasets and more iterations improve performance. Therefore, we set the iteration steps to 150 and utilized the full training dataset in follow-up experiments.

We also conducted experiments on CIFAR-10, where the bottom-up network was an MLP with the structure 3072–100-10. The Adam

Table S1
Classification performance under different optimization methods and settings.

Dataset	Method	20 step		50 step		150 step	
		Train (1000)	Test (10,000)	Train (10,000)	Test (10,000)	Train (60,000)	Test (10,000)
MNIST	CE + Kaiming BP	96.00 ± 0.88 %	88.14 ± 0.23 %	96.02 ± 0.095 %	93.12 ± 0.16 %	98.20 ± 0.15 %	96.16 ± 0.14 %
	CE + Zero BP	90.96 ± 0.33 %	85.31 ± 0.28 %	97.23 ± 0.093 %	93.43 ± 0.083 %	99.04 ± 0.07 %	96.35 ± 0.15 %
	TDCA-net	99.8 %	88.61 %	96.54 %	93.56 %	99.08 %	96.82 %
Fashion-MNIST	CE + Kaiming BP	84.28 ± 2.34 %	76.06 ± 1.25 %	87.592 ± 0.50 %	83.63 ± 0.26 %	91.26 ± 0.37 %	87.19 ± 0.41 %
	CE + Zero BP	72.5 ± 0.59 %	67.43 ± 0.81 %	87.37 ± 0.14 %	83.47 ± 0.25 %	92.33 ± 0.11 %	87.48 ± 0.12 %
	TrFC	8.2 %	10.00 %	8.73 %	9.31 %	5.38 %	5.18 %
	TrTDCA-net	76.1 %	73.29 %	78.28 %	76.99 %	85.94 %	83.91 %

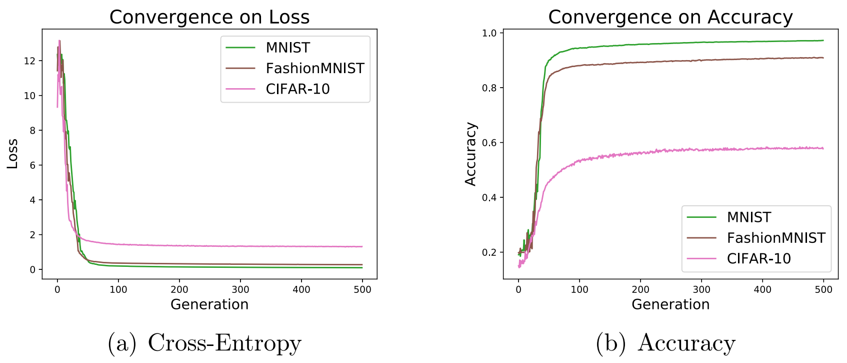


Fig. S7. Convergence of the PGPE algorithm during TDCA-network training. The first 500 iterations of the TDCA-network training's outer loop are selected for analysis, after which there is almost no significant change. Convergence curves of these two metrics across various data sets are depicted in the figures. Panels (a) and (b) illustrate the convergence curve of cross-entropy and accuracy, respectively.

optimizer was used with a learning rate of 0.01, and the training was performed for 150 iterations. In Table S2, it can be observed that TDCA-net under Zero initialization outperforms Zero BP and most biologically plausible algorithms that use Kaiming initialization.

Appendix D. Analysis of on supervised learning

To understand the TDCA framework's dynamics, we studied the behavior of the TDCA-network and bottom-up networks during classification tasks. We focused on how the credit, generated by a well-trained TDCA-network, updates the bottom-up network.

D.1. Convergence of TDCA-network

We tested the efficacy of the PGPE algorithm in optimizing the TDCA-network in outer loop optimization. By training a three-layer fully connected feed-forward network, we present the TDCA-network's convergence process under the PGPE algorithm in Fig. S7. The x-axis of the figure represents the generation of the TDCA-network updates, while the y-axis indicates the cross-entropy and classification accuracy of the bottom-up network after being trained by the TDCA-network.

The figure displays the first 500 generations. Fig. S7(a) presents the final cross-entropy loss of the bottom-up network at the end of the inner loop, while Fig. S7(b) illustrates the classification accuracy. The TDCA-network rapidly converges to optimal performance across various datasets, demonstrating its efficient optimization using the PGPE algorithm.

D.2. Landscape analysis of bottom-up network

To understand the behavior of the bottom-up network under different update strategies, we plotted the loss landscapes [80] of the bottom-up

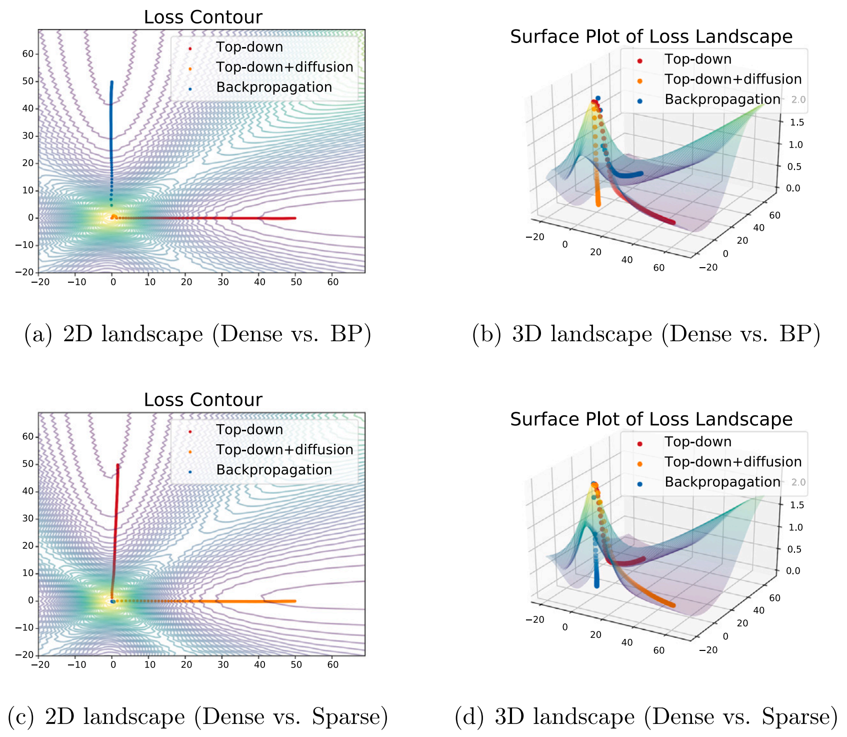


Fig. S8. Parameter loss landscape of a three-layer bottom-up network on the MNIST dataset. Each landscape is defined by the update trajectory of two distinct algorithms. The update trajectories of Top-down (TD), Top-down + diffuse (Diff), and Backpropagation (BP) are almost orthogonal to each other in the TD-BP and TD-Diff spaces.

networks on MNIST and showed the trajectories of parameter updates under the supervision of the top-down method, top-down + diffusion method, and backpropagation (Fig. S8). When plotting the loss landscapes, one of the most important choices is which directions to use to determine the projection plane. We used the PCA direction (used by [80]) of the top-down method and backpropagation in Fig. S8(a) and (b); and the PCA direction of the top-down method and the top-down + diffusion method in Fig. S8(c) and Fig. S8(d). We observed that the top-down trajectory and back-propagation trajectory are almost orthogonal (Fig. S8(a), as is the top-down + diffusion trajectory to the top-down trajectory (Fig. S8(c). Based on Fig. S8(b) and (d), we concluded that the three methods generate distinct update trajectories for the bottom-up model, unrelated to each other.

The convergence analysis shows that the TDCA-network, within the top-down learning framework, can devise a more effective update strategy for the bottom-up network than BP. The orthogonal update trajectories of different methods highlight their unique optimization problem-solving strategies.

Data availability

Data will be made available on request.

References

[1] B.A. Richards, T.P. Lillicrap, P. Beaudoin, Y. Bengio, R. Bogacz, A. Christensen, C. Clopath, R.P. Costa, A. de Berker, S. Ganguli, et al., A deep learning framework for neuroscience, Nat. Neurosci. 22 (11) (2019) 1761–1770.

[2] A. Stolyarova, Solving the credit assignment problem with the prefrontal cortex, Front. Neurosci. 12 (2018) 182.

[3] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536, https://doi.org/10.1038/323533a0, http://www.nature.com/articles/323533a0.

[4] C.D. Gilbert, W. Li, Top-down influences on visual processing, Nat. Rev. Neurosci. 14 (5) (2013) 350–363.

[5] F. Tong, Primary visual cortex and visual awareness, Nat. Rev. Neurosci. 4 (3) (2003) 219–229.

[6] T.P. Lillicrap, D. Cownden, D.B. Tweed, C.J. Akerman, Random synaptic feedback weights support error backpropagation for deep learning, Nat. Commun. 7 (1) (2016) 1–10.

[7] T.P. Lillicrap, A. Santoro, L. Marris, C.J. Akerman, G. Hinton, Backpropagation and the brain, Nat. Rev. Neurosci. 21 (6) (2020) 335–346.

[8] J.C.R. Whittington, R. Bogacz, Theories of error back-propagation in the brain, Trends Cogn. Sci. 23 (3) (2019) 235–250.

[9] Y. Bengio, D.-H. Lee, J. Bornschein, T. Mesnard, Z. Lin, Towards biologically plausible deep learning, arXiv preprint arXiv:1502.04156, 2015.

[10] S. Grossberg, Competitive learning: From interactive activation to adaptive resonance, Cogn. Sci. 11 (1) (1987) 23–63.

[11] F. Crick, The recent excitement about neural networks, Nature 337 (6203) (1989) 129–132.

[12] P. Baldi, P. Sadowski, Z. Lu, Learning in the machine: the symmetries of the deep learning channel, Neural Netw. 95 (2017) 110–133.

[13] S. Löwe, P. O'Connor, B. Veeling, Putting an end to end-to-end: Gradient-isolated learning of representations, Adv. Neural Inf. Process. Syst. 32 (2019).

[14] W.M. Czarnecki, G. Świrszcz, M. Jaderberg, S. Osindero, O. Vinyals, K. Kavukcuoglu, Understanding synthetic gradients and decoupled neural interfaces, in: International Conference on Machine Learning, PMLR, 2017, pp. 904–912.

[15] M. Jaderberg, W.M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, K. Kavukcuoglu, Decoupled neural interfaces using synthetic gradients, in: International Conference on Machine Learning, PMLR, 2017, pp. 1627–1635.

[16] A.H. Marblestone, G. Wayne, K.P. Kording, Toward an integration of deep learning and neuroscience, Front. Comput. Neurosci. 10 (2016) 94.

[17] A. Journé, H.G. Rodriguez, Q. Guo, T. Moraitis, Hebbian deep learning without feedback, arXiv preprint arXiv:2209.11883, 2022.

[18] A. Zador, S. Escola, B. Richards, B. Ölveczky, Y. Bengio, K. Boahen, M. Botvinick, D. Chklovskii, A. Churchland, C. Clopath, et al., Catalyzing next-generation artificial intelligence through neuroai, Nat. Commun. 14 (1) (2023) 1597.

[19] S. Gonzalez, R. Miikkulainen, Improved training speed, accuracy, and data utilization through loss function optimization, in: 2020 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2020, pp. 1–8.

[20] S. Bechtle, A. Molchanov, Y. Chebotar, E. Grefenstette, L. Righetti, G. Sukhatme, F. Meier, Meta learning via learned loss, in: 2020 25th International Conference on Pattern Recognition (ICPR), IEEE, 2021, pp. 4161–4168.

[21] P. Liu, G. Zhang, B. Wang, H. Xu, X. Liang, Y. Jiang, Z. Li, Loss function discovery for object detection via convergence-simulation driven search, arXiv preprint arXiv:2102.04700, 2021.

[22] H. Li, T. Fu, J. Dai, H. Li, G. Huang, X. Zhu, Autoloss-zero: Searching loss functions from scratch for generic tasks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 1009–1018.

[23] X. Guo, K. Wu, X. Zhang, J. Liu, Automated loss function search for class-imbalanced node classification, arXiv preprint arXiv:2405.14133, 2024.

[24] B. Morgan, D. Hougen, Neural loss function evolution for large-scale image classifier convolutional neural networks, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2024, pp. 603–606.

[25] S. Baik, J. Choi, H. Kim, D. Cho, J. Min, K.M. Lee, Meta-learning with task-adaptive loss function for few-shot learning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9465–9474.

[26] W. Adams, J.N. Graham, X. Han, H. Riecke, Top-down inputs drive neuronal network rewiring and context-enhanced sensory processing in olfaction, PLoS Comput. Biol. 15 (1) (2019) e1006611.

[27] M.F.S. Rushworth, M.P. Noonan, E.D. Boorman, M.E. Walton, T.E. Behrens, Frontal cortex and reward-guided learning and decision-making, Neuron 70 (6) (2011) 1054–1069.

[28] E.R. Kandel, J.H. Schwartz, T.M. Jessell, Principles of neural science, vol. 4, McGraw-hill New York, 2000.

[29] Y. Chen, S. Martinez-Conde, S.L. Macknik, Y. Bereshpolova, H.A. Swadlow, J.-M. Alonso, Task difficulty modulates the activity of specific neuronal populations in primary visual cortex, Nat. Neurosci. 11 (8) (2008) 974–982.

[30] Y. Liu, Y. Xin, N.-L. Xu, A cortical circuit mechanism for structural knowledge-based flexible sensorimotor decision-making, Neuron 109 (12) (2021) 2009–2024.

[31] R. Malik, Y. Li, S. Schamiloglu, V.S. Sohal, Top-down control of hippocampal signal-to-noise by prefrontal long-range inhibition, Cell 185 (9) (2022) 1602–1617.

[32] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.

[33] T. Hospedales, A. Antoniou, P. Micaelli, A. Storkey, Meta-learning in neural networks: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 44 (9) (2021) 5149–5169.

[34] S. Gonzalez, R. Miikkulainen, Optimizing loss functions through multi-variate taylor polynomial parameterization, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2021, pp. 305–313.

[35] B. Gao, H. Gouk, T.M. Hospedales, Searching for robustness: Loss learning for noisy classification tasks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 6670–6679.

[36] B. Gao, H. Gouk, Y. Yang, T. Hospedales, Loss function learning for domain generalization by implicit gradient, in: International Conference on Machine Learning, PMLR, 2022, pp. 7002–7016.

[37] Q. Liao, J. Leibo, T. Poggio, How important is weight symmetry in backpropagation? in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, 2016.

[38] A. Nøkland, Direct feedback alignment provides learning in deep neural networks, Adv. Neural Inf. Process. Syst. 29 (2016).

[39] C. Frenkel, M. Lefebvre, D. Bol, Learning without feedback: Direct random target projection as a feedback-alignment algorithm with layerwise feedforward training, arXiv preprint arXiv:1909.01311 10, 2019.

[40] A. Meulemans, M. Tristany Farinha, J. Garcia Ordonez, P. Vilimelis Aceituno, J. Sacramento, B.F. Grewe, Credit assignment in neural networks through deep feedback control, Adv. Neural Inf. Process. Syst. 34 (2021) 4674–4687.

[41] B. Scellier, Y. Bengio, Equilibrium propagation: Bridging the gap between energy-based models and backpropagation, Front. Comput. Neurosci. 11 (2017) 24.

[42] D.-H. Lee, S. Zhang, A. Fischer, Y. Bengio, Difference target propagation, in: Joint European Conference on Machine Learning and Knowledge Discovery in databases, Springer, 2015, pp. 498–515.

[43] G. Hinton, The forward-forward algorithm: Some preliminary investigations, arXiv preprint arXiv:2212.13345, 2022.

[44] G. Dellaferrera, G. Kreiman, Error-driven input modulation: solving the credit assignment problem without a backward pass, in: International Conference on Machine Learning, PMLR, 2022, pp. 4937–4955.

[45] R.F. Srinivasan, F. Mignacco, M. Sorbaro, M. Refinetti, A. Cooper, G. Kreiman, G. Dellaferrera, Forward learning with top-down feedback: Empirical and analytical characterization, arXiv preprint arXiv:2302.05440, 2023.

[46] Y. Song, B. Millidge, T. Salvatori, T. Lukasiewicz, Z. Xu, R. Bogacz, Inferring neural activity before plasticity as a foundation for learning beyond backpropagation, Nat. Neurosci. (2024) 1–11.

[47] J. Zhang, S.A. Bargal, Z. Lin, J. Brandt, X. Shen, S. Sclaroff, Top-down neural attention by excitation backprop, Int. J. Comput. Vis. 126 (10) (2018) 1084–1102.

[48] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu, et al., Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2956–2964.

[49] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, A.C. Courville, Modulating early visual processing by language, Adv. Neural Inf. Process. Syst. 30 (2017).

[50] G. Paraskevopoulos, E. Georgiou, A. Potamianos, Mmlatch: Bottom-up top-down fusion for multimodal sentiment analysis, in: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2022, pp. 4573–4577.

[51] R. Salakhutdinov, G. Hinton, Deep boltzmann machines, in: Artificial Intelligence and Statistics, PMLR, 2009, pp. 448–455.

[52] P. Dayan, G.E. Hinton, R.M. Neal, R.S. Zemel, The helmholtz machine, Neural Comput. 7 (5) (1995) 889–904.

[53] A. Vaswani, Attention is all you need, Adv. Neural Inf. Process. Syst. (2017).

[54] D. Dai, Y. Sun, L. Dong, Y. Hao, S. Ma, Z. Sui, F. Wei, Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers, arXiv preprint arXiv:2212.10559, 2022.

[55] B.A. Richards, T.P. Lillicrap, Dendritic solutions to the credit assignment problem, Curr. Opin. Neurobiol. 54 (2019) 28–36.
[56] A. Lamba,M. R. Nassar, O. FeldmanHall, Prefrontal cortex state representations shape human credit assignment, eLife 12 (2023) e84888. doi:https://doi.org/10.7554/eLife.84888.
[57] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, J. Schmidhuber, Policy gradients with parameter-based exploration for control, in: V. Kůrková, R. Neruda, J. Koutník (Eds.), Artificial Neural Networks - ICANN 2008, Berlin, Heidelberg, Springer Berlin Heidelberg, 2008, pp. 387–396.
[58] T. Salimans, J. Ho, X. Chen, S. Sidor, I. Sutskever, Evolution strategies as a scalable alternative to reinforcement learning (2017).
[59] L. Weng, Evolution strategies, lilianweng.github.io (2019) https://lilianweng.github.io/posts/2019-09-05-evolution-strategies/.
[60] A.N. Sloss, S. Gustafson, 2019 evolutionary algorithms review, Genet. Program. Theory Pract. XVII (2020) 307–344.
[61] X. Zhang, J. Clune, K.O. Stanley, On the relationship between the openai evolution strategy and stochastic gradient descent (2017).
[62] M. Chistiakova, N.M. Bannon, M. Bazhenov, M. Volgushev, Heterosynaptic plasticity: multiple mechanisms and multiple roles, The Neuroscientist 20 (5) (2014) 483–498.
[63] T.E. Chater, Y. Goda, My neighbour hetero—deconstructing the mechanisms underlying heterosynaptic plasticity, Curr. Opin. Neurobiol. 67 (2021) 106–114.
[64] T.E. Chater, M.F. Eggl, Y. Goda, T. Tchumatchenko, Competitive processes shape multi-synapse plasticity along dendritic segments, Nat. Commun. 15 (1) (2024) 7572.
[65] M. Kourosh-Arami, A. Komaki, M. Gholami, S.H. Marashi, S. Hejazi, Heterosynaptic plasticity-induced modulation of synapses, J. Physiol. Sci. 73 (1) (2023) 33.
[66] K.R. Jenks, K. Tsimring, J.P.K. Ip, J.C. Zepeda, M. Sur, Heterosynaptic plasticity and the experience-dependent refinement of developing neuronal circuits, Front. Neural Circuits 15 (2021) 803401.
[67] M. Chistiakova, N.M. Bannon, J.-Y. Chen, M. Bazhenov, M. Volgushev, Homeostatic role of heterosynaptic plasticity: models and experiments, Front. Comput. Neurosci. 9 (2015) 89.
[68] B. Barbour, M. Häusser, Intersynaptic diffusion of neurotransmitter, Trends Neurosci. 20 (9) (1997) 377–384.
[69] A.V. Sem'yanov, Diffusional extrasynaptic neurotransmission via glutamate and gaba, Neurosci. Behav. Physiol. 35 (3) (2005) 253–266.
[70] C. Liu, P.S. Kaeser, Mechanisms and regulation of dopamine release, Curr. Opin. Neurobiol. 57 (2019) 46–53.
[71] S.O. Ögren, T.M. Eriksson, E. Elvander-Tottie, C. D'Addario, J.C. Ekström, P. Svenningsson, B. Meister, J. Kehr, O. Stiedl, The role of 5-ht1a receptors in learning and memory, Behav. Brain Res. 195 (1) (2008) 54–77.
[72] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, JMLR.org, 2014, ICML'14, pp. I–387–I–395.
[73] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, PMLR, 2018, pp. 1587–1596.
[74] B.J. Kim, H. Choi, H. Jang, S.W. Kim, Smooth momentum: improving lipschitzness in gradient descent, Applied Intell. 53 (11) (2023) 14233–14248.
[75] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, J. Schmidhuber, Parameter-exploring policy gradients, Neural Netw. 23 (4) (2010) 551–559.
[76] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[77] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms 2017. arXiv:cs.LG/1708.07747.
[78] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym 2016. arXiv:arXiv:1606.01540.
[79] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, S. Levine, Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. 2019, in: Conference on Robot Learning (CoRL). https://arxiv.org/abs/1910.10897.
[80] H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein, Visualizing the loss landscape of neural nets, Adv. Neural Inf. Process. Syst. 31 (2018).

Author biography

Jianhui Chen is a Special Research Assistant in Institute of Neuroscience, State Key Laboratory of Neuroscience, Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences in Shanghai, China. He has earned his Ph.D. in Neuroscience from University of Chinese Academy of Sciences. Previously, he obtained his B.Eng. in Computer Science and Technology from Northeastern University in Shenyang, China. His research interests focus on brain-inspired neural networks, credit assignment, and structured regularization.

Tianming Yang is currently a Principal Investigator of Center for Excellence in Brain Science and Intelligence Technology (CEB SIT), Chinese Academy of Sciences and Head of the 'Laboratory of Neural Mechanisms of Decision Making and Cognition'. He received his BS degree in Fudan University, pursued his Ph.D. degree at Baylor College of Medicine and conducted his post-doctoral research in at University of Washington. His current research focuses on the neural and computational mechanisms that underlie complex cognition.

Cheng-Lin Liu is a professor at the Institute of Automation of Chinese Academy of Sciences, Beijing, China. He received the BS degree in electronic engineering from Wuhan University, Wuhan, China, the ME degree in electronic engineering from Beijing Polytechnic University, Beijing, China, the PhD degree in pattern recognition and intelligent control from the Chinese Academy of Sciences, Beijing, China, in 1989, 1992 and 1995, respectively. He was a postdoctoral fellow at Korea Advanced Institute of Science and Technology (KAIST) and later at Tokyo University of Agriculture and Technology from March 1996 to March 1999. From 1999 to 2004, he was a researcher at the Central Research Laboratory, Hitachi, Ltd., Tokyo, Japan. His research interests include pattern recognition, machine learning, document analysis and understanding. He has published over 400 technical papers in journals and conferences. He is an Associate Editor-in-Chief of Pattern Recognition Journal and Acta Automatica Sinica, an Associate Editor of several domestic and international journals. He is a Fellow of the CAA, CAAI, the IAPR and the IEEE.

Zuoren Wang is currently a Principal Investigator of Center for Excellence in Brain Science and Intelligence Technology (CEB SIT), Chinese Academy of Sciences and Head of the 'Laboratory of Neural Circuits and Animal Behavior'. He received his BS degree in East China University of Science and Technology, pursued his Ph.D. degree at Rutgers University and conducted his post-doctoral research in at University of California, Berkeley. The research of his lab focuses on neural circuit mechanisms in social behaviors and brain-inspired artificial neural networks.