Research Article | Vol. 34, No. 13 / 29 Jun 2026 / Optics Express 24098
Optics EXPRESS

Promise and pitfalls of random projection algorithms for optical processors

GRIGORII V. FOMIN, DMITRII S. BATANOV, EGOR O. KIMLICHENKO, NIKITA S. SHMYRIN, GLEB M. SUETIN, ARTEM V. CHETVERTUKHIN, ANDREY A. GRUNIN, AND ANDREY A. FEDYANIN*

Lomonosov Moscow State University, Moscow 119991, Russia
fedyanin@nanolab.phys.msu.ru

Abstract: Optical computing is an emerging field with the potential to enhance the performance and energy efficiency of artificial intelligence-related computing. One prospective application of optics in computation is random projection. Random projection, characterized by the use of a Gaussian independently and identically distributed (i.i.d.) matrix, is notable for its simplicity and ability to preserve distances within high-dimensional space. This property is leveraged by several machine learning algorithms to improve computational efficiency, including randomized singular value decomposition (RSVD), no-prior-knowledge exponentially weighted moving average (NEWMA) for change-point detection, and reservoir computing for time-series forecasting. Moreover, the application of random projection extends to neural network training methodologies such as direct feedback alignment (DFA). Optical implementations of random projection have also been explored, offering potential benefits in terms of time efficiency and energy savings, with initiatives like LightOn leading such efforts. This article provides a comprehensive analysis of these algorithms and identifies a trend in which random projection accounts for a small fraction of the total computational time in most algorithms examined. Additionally, we highlight the limitations of optical random projection, demonstrating that it does not significantly enhance performance. This finding raises questions about the relevance of random projection-based optical processing units.

1. Introduction

Artificial Intelligence (AI) has become a cornerstone in various sectors by enhancing efficiency and introducing new capabilities. However, the complexity and scale of modern neural networks (NNs) require powerful computational resources, making the training process both costly and time-consuming. A widely adopted approach to address this challenge involves the development of more advanced computing architectures, including state-of-the-art GPUs and TPUs, designed to accelerate matrix multiplication (MM) processes, which constitute a significant portion of computational overhead.

An alternative approach shifts the focus away from universal MM optimization and instead focuses on adapting optimizations to specific MM cases or other operations that are uniquely related to the particular algorithm in question. Beyond digital accelerators, alternative approaches explore unconventional physical principles for computation, such as thermodynamic [1] and mechanical [2] systems for computing. Among these physical platforms, optics is particularly promising, as it leverages additional properties of light to accelerate computation, as demonstrated by commercial optical computing hardware such as Lightmatter's Envise photonic accelerator [3] and Optalysys' optical processors [4].

We examine the approach to accelerating AI and machine learning (ML) algorithms through random matrix multiplication (RMM), also known as random projection (RP) techniques [5], which can be significantly accelerated using optics. This speed-up becomes particularly effective

#599206 https://doi.org/10.1364/OE.599206
Journal © 2026 Received 24 Mar 2026; revised 1 Jun 2026; accepted 8 Jun 2026; published 22 Jun 2026

Optics EXPRESS

when the data dimensionality is sufficiently large, allowing optical RP to outperform GPU-based or TPU-based RMM.

The capability of RP to be accelerated optically is leveraged in Optical Processing Units (OPUs) that utilize random projection to perform faster computations and enhance algorithms dependent on RP. LightOn [5] was among the pioneers in this field, introducing a non-von Neumann photonic AI accelerator designed to optimize large-scale matrix computations while overcoming memory and energy limitations inherent to conventional computing architectures. Unlike integrated photonic processors [6] that intrinsically limited on small-dimensional signals and edge computing applications, LightOn's approach targets high-dimensional data processing, enabling computations on vectors with dimensionality up to 1M and delivering 1500 TeraOPS.

Despite its computational efficiency, the OPUs are inherently limited to algorithms that explicitly leverage random projection, restricting its applicability to tasks that can be reformulated within this framework. This raises the question of whether these algorithms incorporate additional computational components beyond RP. If therefore, it becomes critical to assess the relative contribution of these components compared to the RP step. Understanding this balance is essential for determining the overall acceleration potential of the OPUs.

In this paper, we explore several RP algorithms, including Randomized Singular Value Decomposition (RSVD) [7], No-prior-knowledge Exponentially Weighted Moving Average (NEWMA) [8], Reservoir Computing (RC) [9], and Direct Feedback Alignment (DFA) [10], and their actual acceleration using OPUs. The data dimensions are considered to be sufficiently large to treat the OPU ideal, allowing us to assume that RP is performed instantaneously. We then compare the theoretical acceleration with practical examples to evaluate the potential benefits of computing RP with OPUs.

It is also important to note that the evaluation method proposed in this work is intentionally coarse and relies on several strong approximations and assumptions. In particular, it does not account for factors such as I/O latency, data transfer overhead, or the impact of optical noise on algorithmic convergence, as well as other practical aspects that may significantly influence the computational complexity of algorithms in real-world implementations. This simplification is motivated by the fact that our goal is to provide a method for rapidly screening and eliminating algorithms that are unlikely to benefit from optical acceleration.

If an algorithm fails to meet this optimistic estimate, it is highly unlikely to satisfy a more realistic evaluation that includes additional overheads such as I/O latency. The presence of such components—e.g., data loading and storage operations—reduces the fraction of total execution time attributable to the random projection step, thereby lowering the maximum achievable acceleration. Consequently, passing our preliminary assessment should not be interpreted as evidence that an algorithm is well suited for OPU acceleration. Rather, it only indicates that the algorithm warrants a more detailed analysis that incorporates I/O latency and other system-level factors affecting the overall computation time.

2. Optical random projection

Random Projection is a method of dimensionality reduction that involves projecting high-dimensional data $x \in \mathbb{R}^N$ onto a lower-dimensional space $y \in \mathbb{R}^n$ using a random matrix $P \in \mathbb{R}^{N\times n}$. RP is widely recognized for its efficiency and simplicity, particularly when dealing with large datasets where traditional dimensionality reduction methods may be time-consuming. However, in the context of our study, the term 'random projection' is used to refer to all instances of RMM, even when the dimensionality of the projected vector y exceeds that of the input vector x.

The method of implementing random projection optically is illustrated in Fig. 1. Notably, aside from the time required for data transfer between the Computer, Spatial Light Modulator (SLM), and camera, this optical approach ensures a fixed computing time, which is independent on the input vector size and solely determined by the frame rates of the camera and SLM. This results

Optics EXPRESS

in a time complexity of $O(1)$, although with a relatively large constant factor. Thus, OPU may surpass digital devices in performance when handling high-dimensional data. Such scalability with input vector dimensionality sets OPU apart from traditional computing units.

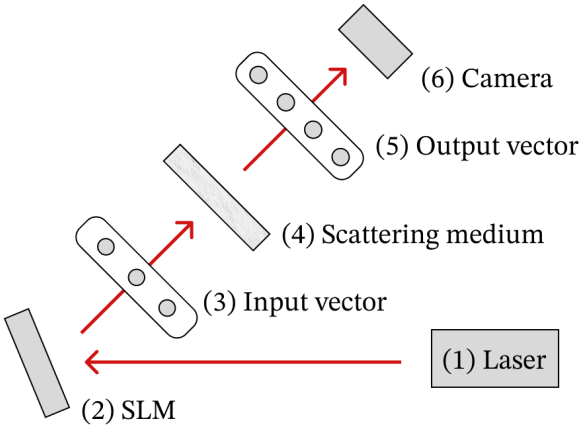


Fig. 1. An experimental set-up for optical Random Projection. The SLM is illuminated by a red laser. The light reflected from the SLM represents an analog input vector $x \in \mathbb{R}^n$, where $n = p_v \cdot p_h$, and p_v, p_h denote the number of pixels along the vertical and horizontal axes of the SLM screen, respectively. The reflected light propagates through a scattering medium, and the resulting scattering pattern is captured by a camera. The image obtained from the camera represents an analog output vector $y \in \mathbb{R}^m$, where $m = q_v \cdot q_h$, with q_v and q_h being the pixel resolution along the vertical and horizontal axes of the camera sensor. In the idealized model adopted throughout this work, the overall optical transformation is represented as a random projection $y = Px$, where P is a random matrix [11]. This model is used for complexity analysis of algorithms relying on linear random projections.

Two distinct optical implementations of random projection have been reported in the literature. The original LightOn architecture [5,12] produces a nonlinear random feature map ($\phi(Hx)$), where (H) is the transmission matrix of the scattering medium and ($\phi(\cdot)$) arises from photodetection. Such systems are mainly intended for random-feature and kernel-based methods. In contrast, recent approaches [13] combine optical measurements with lightweight digital post-processing to obtain a representation statistically equivalent to a linear Gaussian random projection (Px). Since algorithms such as RSVD and DFA require linear random projections, we adopt the latter idealized model throughout this work. The additional post-processing introduces a complexity proportional to the output dimensionality, which is negligible compared to the ($O(n^2)$) complexity of dense matrix-vector multiplication. Furthermore, our analysis already assumes instantaneous random projection; therefore, any practical overhead can only reduce the fraction of execution time attributable to random projection and further limit the achievable acceleration of algorithms such as RSVD and DFA.

The optical random-projection models discussed above rely on the assumption of coherent monochromatic illumination. Under this condition, propagation through the scattering medium is described by a fixed transmission matrix, enabling either the generation of nonlinear random features or, after suitable post-processing, a representation statistically equivalent to a linear random projection. If the illumination is not sufficiently coherent, contributions from different wavelengths and propagation angles add incoherently at the detector. This lowers the speckle contrast and reduces the effective rank of the transformation, causing the realized operation to deviate from the ideal random mappings assumed in this work. The present analysis assumes the ideal coherent case. Any practical implementation of optical random projection should therefore

Optics EXPRESS

ensure sufficient temporal and spatial coherence of the source, so that the optical operation faithfully approximates the random projection or random feature map considered in the analysis.

OPU performance: Here, we present a preliminary assessment of the performance of the OPU specifically for squared matrix-vector multiplication. The computational complexity associated with multiplying a vector of dimensionality n by an $n \times n$ square matrix is $O(n^2)$. In contrast, the complexity for multiplying two square matrices of size $n \times n$ is $O(n^3)$, excluding advanced algorithms such as the Strassen algorithm [14].

As optical RP operates with $O(1)$ complexity, the time required to multiply a vector by a random matrix remains constant regardless of dimensionality. To evaluate the performance in operations per second (OPS) of an OPU, the product of the SLM frame rate and the number of operations required for each random projection can be calculated. In this context, the frame rate of the camera has been excluded, as it is significantly bigger than that of the SLM:

$$OPS_{OPU} = 2 \cdot f \cdot n^2, \tag{1}$$

where f is the frame rate of SLM, approximately 1 kHz. According to the performance metrics presented in [15,16], as detailed in Table 1, the NVIDIA H100 NVL and the NVIDIA GB200 Grace Blackwell Superchip (GBS) exhibit distinct capabilities in terms of FLOPS (Floating Point Operations Per Second).

Table 1. Performance Comparison: H100 NVL vs. GB200 GBS.

Metric	H100 NVL	GB200 GBS
FP64	30 TFLOPS	90 TFLOPS
FP64 Tensor Core	60 TFLOPS	90 TFLOPS
FP32	60 TFLOPS	180 TFLOPS
FP32 Tensor Core	0.8 PFLOPS	5 PFLOPS
FP16 Tensor Core[a]	1.7 PFLOPS	10 PFLOPS
FP8 Tensor Core[a]	3.3 PFLOPS	20 PFLOPS

[a]Indicates performance with sparsity.

When analyzing FP8 performance, the results indicate that the OPU begins to surpass the performance of other processing units at vector dimensionalities of approximately 1.3M for the NVIDIA H100 NVL and 3.2M for the NVIDIA GB200 Grace Blackwell Superchip. Importantly, the analysis does not account for memory access times or potential storage constraints associated with handling vectors and matrices of such substantial dimensions.

A squared matrix $P \in \mathbb{R}^{n \times n}$, corresponding to a vector $x \in \mathbb{R}^n$ with $n = 3.2 \times 10^6$, contains approximately 10^{13} elements. Models using such matrices imply at least 10 trillion parameters, for which current confirmation of practical application is lacking. Therefore, RP-based OPUs are currently less efficient than conventional processing units for existing models.

3. Random matrix multiplication algorithms

Let us consider the use of the RP method to accelerate ML algorithms, particularly in scenarios where RP can reduce computational time by an order of magnitude or more. The analysis assumes large data dimensionality (Section IV), which forms the basis of our assumptions regarding the scalability of the algorithms under consideration. This assumption implies that robust algorithms, capable of scaling effectively with increasing dimensions without significant performance degradation, have already been selected.

However, the process of selecting such algorithms is a critical step that requires careful attention. In this context, it is necessary to analyze the scaling laws [17] that govern deep learning RP

Optics EXPRESS

algorithms. Scaling laws describe the mathematical relationships that define how an algorithm's performance, efficiency, and computational requirements change as key hyperparameters (e.g., model size, the amount of training data) are scaled up. These laws are particularly important when working with high-dimensional data, as they enable predictions of algorithmic behavior as the problem size increases.

Understanding these relationships is crucial for assessing whether an algorithm maintains its efficiency and accuracy under the large-scale computations typical of high-dimensional datasets. The effectiveness of random projection techniques, particularly when combined with optical computing approaches, depends heavily on the scalability of the underlying algorithm. Algorithms that degrade in performance as dimensions grow do not benefit significantly from random projection or optical acceleration.

Therefore, a thorough evaluation of scaling laws, such as those discussed in [17], is vital for determining the feasibility of using RP-based optical methods in specific machine learning applications. This analysis ensures that the algorithms chosen for optical random projection are not only theoretically suitable but also practically efficient at large scales.

In light of this, random projection can be effectively applied to several machine learning algorithms, such as RSVD, NEWMA, RC, and DFA. In the following sections, we examine these algorithms and provide a comprehensive analysis of their computational complexity and scalability, where relevant, to evaluate their suitability for random projection and optical acceleration. It is important to note that all potential accelerations discussed below are evaluated on a per-iteration basis to demonstrate hardware speedup, rather than to compare algorithms; thus, various methods of interacting with the random projection to improve algorithmic results, e.g. [36,37], remain out of the scope of this research.

3.1. RSVD: randomized singular value decomposition

RSVD which is given in Algroithm 1 is designed to efficiently compute an approximate Singular Value Decomposition (SVD) for large matrices, addressing the computational challenges posed by traditional SVD methods. Standard SVD approaches, though powerful, become impractical for large-scale matrices due to their high computational complexity and memory requirements. RSVD overcomes these limitations by employing random projection to reduce the dimensionality of the problem, enabling faster computations while maintaining a high level of accuracy.

The primary objective of RSVD is to compute the SVD for a matrix $X \in \mathbb{R}^{n\times m}$:

$$X = U_X \Sigma V, \tag{2}$$

where $\Sigma \in \mathbb{R}^{n\times m}$ is a diagonal matrix whose elements are singular values, $U_X \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times m}$ are unitary matrices consisting of right and left singular vectors, respectively.

RSVD starts with projecting the original high-dimensional data matrix X into a subspace of smaller dimension using a random matrix $P \in \mathbb{R}^{m\times r}$:

$$Z = XP. \tag{3}$$

Here, Z is a matrix of smaller size. The next step is QR decomposition, where Q is an orthonormal basis for Z:

$$Z = QR. \tag{4}$$

Following this, the original matrix X is projected onto the lower-dimensional basis Q, facilitating an efficient SVD within this subspace:

$$Y = Q^T X. \tag{5}$$

Then, we compute the SVD for Y:

$$Y = U_Y \Sigma V. \tag{6}$$

Optics EXPRESS

However, U_Y must be projected back to the original high-dimensional space:

$$U_X = QU_Y. \tag{7}$$

Furthermore, Σ and V in Eq. (6) match those in Eq. (2). In this way, all components of the SVD are obtained.

Algorithm 1. Randomized Singular Value Decomposition

RSVD *Gray background indicates random projection.*	**Complexity**
Consider $X \in \mathbb{R}^{n\times m}$ is an input matrix.	
Step 1: Random projection	
Generation of random matrix $P \in \mathbb{R}^{m\times r}$	$O(mr)$
Random projection $Z = XP$	$O(nmr)$
QR decomposition $Z = QR$	$O(nr^2)$
Step 2: Singular decomposition	
Projection $Y = Q^T X$	$O(mnr)$
Singular decomposition $Y = U_Y \Sigma V$	$O(mr^2)$
Projection $U_X = QU_Y$	$O(nr^2)$

For RSVD, only the random projection step $Z = XP$ can be accelerated by an OPU, while QR decomposition, computation of $Y = Q^T X$, SVD of Y, and reconstruction of U_X remain digital. Each row x_i of X is encoded on the SLM and propagated through a scattering medium. The resulting optical measurements are recorded by a camera and processed following the approach of [13], yielding a vector statistically equivalent to the linear random projection $z_i = x_i P$. Repeating this procedure for all rows forms the matrix Z, after which the remaining RSVD computations are performed digitally.

The analysis of the algorithm indicates that the complexity of RSVD is primarily influenced by the Random Projection $O(nmr)$ and the subsequent projection $Y = Q^T X$, also $O(nmr)$. Notably, the RP component can account for up to 50% of the total complexity. However, the multiple vector-matrix multiplications increase the complexity of optical processing. As a result, this does not significantly impact the overall performance of the algorithm, implying that opportunities for substantial acceleration may be limited.

3.2. NEWMA: no-prior-knowledge exponential weighted moving average

In this section, we examine the online change-point detection method NEWMA [8], providing an overview of its principles sufficient for complexity analysis and for evaluating the Random Projection component. The NEWMA algorithm is shown in Algorithm 2. It is used for change-point detection, a technique that identifies moments in a time series when underlying statistical properties, such as mean or variance, change significantly.

These change points indicate moments where the data's behavior changes, which is essential for analyzing and reacting to dynamic systems. More detailed discussion on change-point detection can be found in [18]. In change-point detection for time series, methods are classified as offline or online. Offline methods analyze the full series and are more resource-intensive but offer higher accuracy. In contrast, online methods process data in real-time, enabling quick adaptation and efficient change-point detection.

Consider the time sequence $x_t \in \mathbb{R}^N$. We define a mapping function $\Psi : \mathbb{R}^N \to \mathbb{R}^n$ that maps initial space to another multivariate space. We consider the case where the mapping function is RP, although it can generally be arbitrary [8]. Define forgetting factors λ and Λ such as

Optics EXPRESS

Algorithm 2. No-Prior-Knowledge Exponential Weighted Moving Average

NEWMA *Gray background indicates random projection*	**Complexity**
`for` $t = 1, 2, \ldots$ `do`	
$z_t = (1-\Lambda)z_{t-1} + \Lambda\Psi(x_t)$	$O(Nn)$
$z'_t = (1-\lambda)z'_{t-1} + \lambda\Psi(x_t)$	$O(Nn)$
`if` $\|z_t - z'_t\| \geq \tau$ `then`	$O(n)$
Mark t as the change point	

$0<\lambda<\Lambda<1$ that characterize the rate of weight reduction, and threshold value τ. The norm of the difference between two exponentially weighted moving averages is used as a parameter for detecting changes. When this value surpasses the stopping criterion, the point is identified as a change-point:

$$\|z_t - z'_t\| \geq \tau, \tag{8}$$

where z_t and z'_t are calculated as follows:

$$z_t = (1 - \Lambda)z_{t-1} + \Lambda\Psi(x_t), \tag{9}$$

$$z'_t = (1 - \lambda)z'_{t-1} + \lambda\Psi(x_t). \tag{10}$$

For NEWMA, the optical system is used to implement the mapping function $\Psi(x_t)$ applied to each incoming sample. At each time step, the input vector x_t is encoded on the SLM and propagated through the scattering medium, generating a high-dimensional random feature representation that serves as $\Psi(x_t)$. The resulting optical measurements are recorded by a camera and transferred to a conventional processor, where the exponentially weighted moving averages z_t and z'_t are updated and the detection criterion $|z_t - z'_t|$ is evaluated against the threshold τ. Since the mapping $\Psi(x_t)$ constitutes the computationally dominant part of the algorithm, while the remaining operations involve only low-dimensional vector updates and norm evaluation, NEWMA represents one of the most favorable scenarios for OPU acceleration.

The asymptotic analysis suggests that the Random Projection component dominates the computational complexity, with $O(Nn)$ for RP compared to $O(n)$ for computing the norm, and the asymptotic RP fraction $Nn/(Nn + cn)$ approaches unity at large N. However, as we show empirically in Section 4.2, this asymptotic regime is not reached at the dimensions of published NEWMA experiments; constants in the $O(n)$ operations of Algorithm 2, as well as implementation-level effects on digital hardware, keep the realized RP fraction substantially below unity in practice.

3.3. RC: reservoir computing

RC is a framework for recurrent neural network training that utilizes a fixed, high-dimensional dynamical system (the reservoir) to project input signals into another space. It is primarily used for its computational efficiency in modeling complex temporal dynamics, making it particularly well-suited for tasks such as time-series prediction and chaotic system forecasting. The RC architecture is schematically shown in Fig. 2, while the algorithm can be found in Algorithm 3.

The input vector $i(t) \in \mathbb{R}^m$ is introduced into a multi-dimensional dynamic system, the reservoir. The reservoir is described by the vector $r(t) \in \mathbb{R}^p$, whose initial state is determined randomly. Let $W_{\text{res}} \in \mathbb{R}^{p\times p}$ define the internal connections between reservoir nodes, and $W_{\text{in}} \in \mathbb{R}^{p\times m}$ define the connections between the input and the reservoir nodes. Both matrices are initialized randomly

Optics EXPRESS

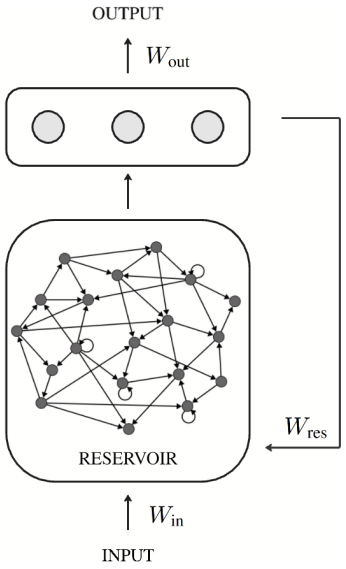


Fig. 2. The Reservoir Computing architecture. It is composed of three primary layers: an input layer, a recurrent hidden layer (the reservoir), and an output layer. This schema depicts how input signals are introduced to the reservoir, where they undergo a high-dimensional transformation. Subsequently, the reservoir's state is mapped to the final output via a linear readout mechanism.

Algorithm 3. Reservoir Computing

Reservoir Computation *Gray background indicates random projection*	**Complexity**
Consider a set of input data $i(t) \in \mathbb{R}^m$. *For simplicity, let us assume* $t \ll p$, *and also let* $T/\Delta t = N - 1$.	
Step 1: Reservoir State Calculation	
Initialization of W_{in}, W_{res}	
`for` $t = -T, -T + \Delta t, \ldots, 0$ `do`	
$r(t + \Delta t) = f\,(W_{in} g[i(t)] + W_{res} g[r(t)])$	$O(N \cdot p^2)$
Step 2: Linear Model Training	
`train linear` $(W_{out}, r(0))$	$O(p^2 + p^3)$

and fixed during the reservoir computing process. The evolution of the reservoir can be written as

$$r(t + \Delta t) = f[W_{\text{in}} i(t) + W_{\text{res}} r(t)]. \tag{11}$$

To obtain the output prediction vector $o(t)$, a simple linear regression over the reservoir state $r(t)$ can be used:

$$o(t) = W_{\text{out}} r(t). \tag{12}$$

The training occurs by adjusting the weights in the output layer $W_{\text{out}} \in \mathbb{R}^{k \times p}$.

Consider optical RC. It leverages a physical scattering medium and a phase-modulating SLM to form its reservoir. This implementation results in complex-valued matrices $W_{\text{res}}, W_{\text{in}}, W_{\text{out}}$. Consequently, the system's state update is governed by a following modified equation:

$$r(t + \Delta t) = f[h(i(t)) + W_{\text{res}} g(r(t))], \tag{13}$$

Optics EXPRESS

where f and g are nonlinear functions characterizing the non-linearity in the camera detection and SLM, respectively. The training stage proceeds as follows. For a given time interval $-T \leq t \leq 0$, the states of the reservoir $r(t)$ are calculated. Then, the output weights W_{out} are adjusted to minimize the prediction error using a simple linear regression model.

For optical Reservoir Computing, the scattering medium itself forms the reservoir. At each time step, the input signal $i(t)$ and the current reservoir state $r(t)$ are encoded on the SLM and propagated through the scattering medium, generating a high-dimensional random feature representation. The resulting optical measurements are recorded by a camera and fed back to the system to form the next reservoir state according to Eq. (13). Repeating this procedure over the interval $-T \leq t \leq 0$ produces the sequence of reservoir states $r(t)$. These states are subsequently transferred to a conventional processor, where the output weights W_{out} are obtained using linear regression.

Since $p \ll N$, the first part of this algorithm, which can be accelerated optically, has greater computational complexity. Thus, the use of random projection looks promising in this algorithm.

3.4. DFA: direct feedback alignment

DFA is an alternative algorithm for training NNs [10]. To simplify the explanation, we draw a parallel with the commonly used Gradient Descent-based algorithm, Backpropagation (BP). The distinctive feature of DFA lies in its approach to error propagation. Unlike BP, where the error is propagated through each layer, DFA directly projects the error from the output layer to the current layer. Despite the seemingly unconventional nature of this method, studies have demonstrated that DFA can learn complex nonlinear features and effectively extract useful information [19]. The DFA schematic architecture and its corresponding algorithm are shown in Fig. 3 and Algorithm 4, respectively.

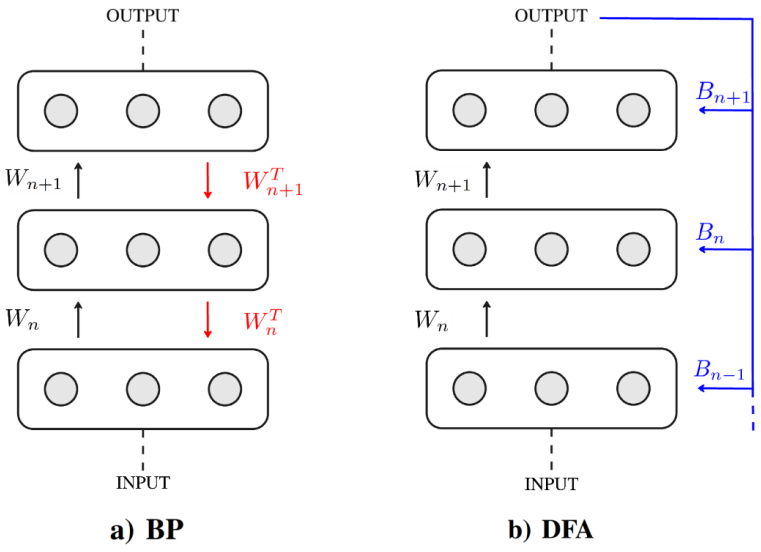


Fig. 3. Comparison of the error propagation mechanisms in two neural network training algorithms. (a): Backpropagation. The output error is propagated backward through the network layers using the transpose of the forward weights W^T. This process calculates the precise gradient for each layer's weights. (b): Direct Feedback Alignment. The output error is sent directly to each layer via a fixed, random feedback matrix B_i. This method avoids the need for a backward pass using the network's own weights, providing a biologically plausible alternative to backpropagation.

Optics EXPRESS

Algorithm 4. Direct Feedback Alignment

DFA *Gray background indicates random projection*	**Complexity**
Let us assume that the trained NN is fully connected with N layers of size $n \times n$, and the input batch of size $b \times n$.	
Step 1. Forward propagation	
N activations $f(W_i h_{i-1})$	$O(Nbn^2)$
Step 2. Backward propagation	
The last layer is computed as BP	$O(bn^2)$
N random projections $B_i \delta a_N$	$O(Nbn^2)$
N Hadamard products $\odot f_i'(a_i)$	$O(Nbn)$
N updated weights $\delta a_i h_i^T$	$O(Nb^2 n)$

Let the NN consists of N layers, then the forward propagation can be written as follows:

$$\forall i \in [1, \ldots, N] : a_i = W_i h_{i-1},\ h_i = f(a_i),$$

where W_i is the weight matrix of the i-th layer, a_i is the output of the i-th layer without activation function (nonlinearity) f and h_i is the output of the i-th layer after the activation function. It is also noteworthy that $h_0 = X$, where X is the input data vector, and $h_N = \hat{y}$, where $\hat{y}$ is the network prediction. The loss function is denoted as $\mathcal{L}(y, \hat{y})$ where y are the correct answers.

Thus, for BP we have:

$$\delta W_i = \frac{\partial \mathcal{L}}{\partial W_i} = \delta a_i h_{i-1}^T, \quad \delta a_i = \frac{\partial \mathcal{L}}{\partial a_i} = (W_{i+1}^T \delta a_{i+1}) \odot f'(a_i),$$

and for DFA we have:

$$\delta W_i = \frac{\partial \mathcal{L}}{\partial W_i} = \delta a_i h_{i-1}^T, \quad \delta a_i = \frac{\partial \mathcal{L}}{\partial a_i} = (B_i \delta a_N) \odot f'(a_i).$$

Clearly, the expressions for BP and DFA are similar, except that the transposed weight W_{i+1}^T is replaced by a fixed random matrix B_i, and the gradient is taken not from the previous layer δa_{i+1} but from the last layer δa_N.

For DFA, the optical system can be used to accelerate the computation of the random feedback projections $B_i \delta a_N$ during the backward pass. The output-layer error δa_N is encoded on the SLM and propagated through the scattering medium. The resulting optical measurements are recorded by a camera and processed using a lightweight post-processing scheme similar to that proposed in Ref. [13], yielding vectors statistically equivalent to the linear random projections $B_i \delta a_N$. These projected error signals are then used to compute the local gradients δa_i and update the network weights digitally.

Since only the random feedback projection stage is offloaded to the optical hardware, while the remaining backward-pass operations and the entire forward pass remain digital, the achievable acceleration is limited.

4. Optical boosting

To conduct a thorough analysis of the proposed algorithms and assess the practical necessity of utilizing OPU, it is essential to establish a set of criteria and conditions. This section outlines these criteria, details the methodological framework employed for evaluating the algorithms, and provides a complexity analysis of them.

Optics EXPRESS

4.1. Conditions and feasibility criterion

The feasibility of employing an OPU in algorithmic implementations relies on a comparative analysis of the execution times of the different components of the algorithm. In particular, the potential performance gain depends on the fraction of the total runtime occupied by the Random Projection (RP) step when implemented with classical computation. This consideration is closely related to the principle formulated in Amdahl's Law [20], which states that the overall speedup of a system is fundamentally limited by the fraction of the computation that cannot be accelerated. In the present context, the OPU effectively accelerates the RP stage; therefore, a substantial improvement in total runtime can only be achieved if this stage constitutes the dominant portion of the computational cost. Consequently, a practical criterion for the effective use of an OPU is that the execution time of the RP step, when performed with a classical algorithm, should significantly exceed that of the remaining stages of the algorithm. In particular, the RP stage should account for the vast majority of the runtime, ideally taking at least an order of magnitude longer than the other processes involved.

The choice of a one-order-of-magnitude threshold is not arbitrary. In practice, the performance of a realized hardware accelerator is expected to fall short of its theoretically estimated speedup due to implementation overheads, hardware constraints, data-movement costs, and other system-level effects that are difficult to account for in analytical models. Furthermore, the adoption of a new computational architecture typically entails substantial engineering and economic costs, including the development and integration of new hardware, adaptation of software pipelines, and the establishment of production and supply chains. For this reason, modest improvements in performance (e.g., on the order of 20–30%) would likely be insufficient to justify such a transition. We therefore adopt a pragmatic criterion requiring a potential speedup of at least one order of magnitude, which we consider a reasonable margin when moving from theoretical analysis to a realizable hardware implementation. It should be emphasized that this threshold does not follow from a strict mathematical argument but rather represents a practical engineering guideline. In addition, historical experience shows that theoretical performance gaps may narrow once mature software and architectural optimizations are applied. A well-known example is the comparison between GPUs and CPUs, where early claims of two-orders-of-magnitude speedups were later shown to depend strongly on workload characteristics and implementation details [21].

This analysis assumes that the input data size is sufficiently large, rendering the time required for RP using an OPU negligible compared to the processing time of the other algorithm's components. This idealized assumption allows the RP time to be excluded from subsequent computational evaluations, enabling a clearer assessment of the performance improvements achieved through OPU-based RP. To estimate computational complexity for each algorithm, specific constraints are defined, focusing on the dominant variable n, and expressed in Big O notation as $O(n)$, while other variables are excluded for simplicity.

4.2. Algorithms analysis

We consider the four algorithms: RSVD, NEWMA, RC, and DFA. Using the pseudocode provided in the Section III we estimate the maximum potential RP component in each algorithm. We then calculate the maximum achievable acceleration if OPU is applied. The results are summarized in Table 2. The "Max RP Component" refers to the theoretical upper limit of the RP's contribution within the computational workload of the algorithm. In contrast, the "Real RP Component" represents the empirically measured proportion of the RP contribution for a single instance of algorithm application.

RSVD: Let's revisit the pseudocode provided in Section III Algorithm 1: RSVD. For simplicity, assume a large compression factor where $1 \ll \frac{m}{r}$ or, equivalently, $r \ll m, n$. In this scenario, we can disregard lower-order terms, including the generation of the random matrix, QR decomposition, singular value decomposition, and projection $U_X = QU_Y$. Consequently, the

Optics EXPRESS

Table 2. Comparative performance metrics of four distinct algorithms: DFA, RSVD, NEWMA, and RC.

Metric	DFA	RSVD	NEWMA	RC
Time complexity	$O(2Nbn^2)$	$O(nmr)$	$O(kNn)$	$O(Np^2 + p^3)$
Max RP component	50%	50%	100%	100%
Real RP component	32%	15%	48%	75%

complexity of RSVD is primarily determined by the complexity of the Random Projection, which is $O(nmr)$, and the complexity of the projection $Y = Q^T X$, also $O(nmr)$. Therefore, the maximum theoretical RP component is $\frac{nmr}{2nmr}$, which approaches 50%. Notably, that the decomposition is performed for matrices, which implies the need for multiple Vector-Matrix Multiplications (VMMs). This causes the complexity of the OPU to be $O(n)$ instead of $O(1)$.

We also present numerical computation results for matrices sized according to standard image dimensions. Table 3 shows that as the resolution increases (from Full HD to 8K), the proportion of time spent on the RP component also increases for both compression factors which is proportional to $\frac{m}{r}$. For instance, the RP component in RSVD, when compressing an 8K image with a resolution of 7680 x 4320 pixels, constitutes only 12-14% of the total computation when the image is compressed by factors of 10 and 40, respectively. Despite higher compression levels leading to a larger percentage of time spent on the RP component, the increase is relatively modest.

Table 3. Numerical Simulation of the Dependence of RP Time Percentage on Total Algorithm Time Across Varying Compression Levels.

Compression / Resolution	Full HD	4K	8K
10 times	7.7%	10.2%	12.5%
40 times	12.5%	13.3%	14.2%

Table 3 shows that the RP component does not dominate the computation time and remains well below the theoretical maximum of 50%, even at higher resolutions and compression levels. This indicates that significantly higher-dimensional matrices are necessary to achieve more substantial acceleration.

NEWMA: This section quantifies the RP component in the NEWMA algorithm empirically. Performance and speed comparisons between NEWMA and other online methods can be found in [22,23]. To measure the realized RP fraction, we benchmarked the wall-clock time of one NEWMA step on a digital backend, separating the projection $\Psi(x_t) = Px_t$ from the remaining operations of Algorithm 2. The benchmark was implemented in PyTorch with fp16 precision; each measurement timed a block of $T = 1000$ consecutive NEWMA steps to suppress timer overhead, with a warm-up phase discarded and the median over repeated runs reported. The benchmark code and raw data are openly available in the project repository [35].

At the dimensions of the synthetic Gaussian-mixture experiment in [8] ($N = 100$, $n = 3000$), the measured RP fraction is $f_{\mathrm{RP}} \approx 0.48$, corresponding to a maximum theoretical speedup of approximately 2× under an idealized OPU. The regime where f_{RP} exceeds 0.9 is reached only for $N \gtrsim 10^4$ at $n = 3000$, two orders of magnitude beyond the dimensions of [8]. This confirms the empirical statement anticipated in Section 3.2: the asymptotic RP fraction is not realized at the dimensions of published NEWMA experiments.

Reservoir Computing: While RC shares similarities with RNNs and LSTMs [24–26] we focus on RC's random projection component. We also note a New Generation RC [27]. Generally, the value of N is chosen to be as large as possible; therefore, we assume that $N>>p$. In this case, the time for random projection constitutes at least $N/(N + p)$ that tends to 100% of the

Optics EXPRESS

total execution time of the algorithm, indicating its dependence on the number of recursive calculations in the reservoir.

To estimate actual RP component we site the following study [9]. It indicates that for parameter values $m = 64$, $N = 9 \cdot 10^5$, and $p = 5 \cdot 10^4$, the OPU demonstrates a x4 advantage over the GPU that corresponds to 75% of actual RP component in the algorithm. Thus, RC appears to be a promising algorithm for implementation on an OPU, as the theoretical acceleration could exceed an order of magnitude. However, as demonstrated above, achieving such acceleration may be challenging for certain parameters, such as input data sizes and the number of iterations in RC.

DFA: Despite initial perceptions, DFA has shown the capability to learn complex features and train deep neural networks (DNNs), including Convolutional Neural Networks (CNNs) [28]. However, most benchmark results indicate that it consistently yields lower performance metrics compared to BP [29,30]. Various modifications of DFA, such as those discussed in [31,32], have been proposed to enhance the algorithm, yet they still fall short in performance metrics. Recent studies have shown that DFA is effective for pre-training decoder models with more than 1B parameters. Nevertheless, its loss function diverges from that of BP after the same number of iterations, indicating that a precise comparison requires additional data [33]. Further investigations into applying DFA for pre-training large language models (LLMs) [34] reveal that the performance degradation worsens as the model scales. These findings raise concerns about the algorithm's relevance and scalability.

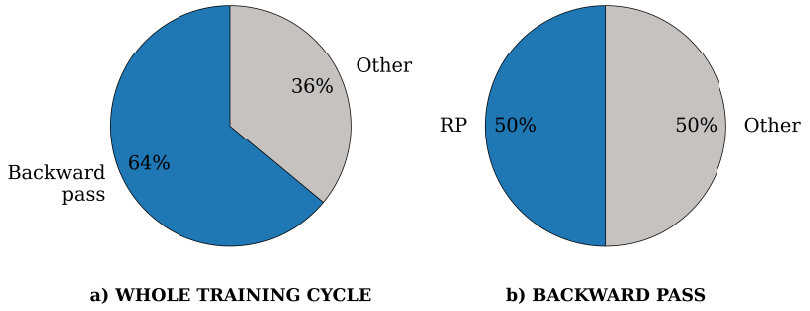
The acceleration is computed for DFA with implemented optical RP where it is possible over DFA calculated using classical VMM. Parts, that may be realized optically highlighted with grey color in Algorithm 4: DFA. The code is available at GitHub [35]. We assume, for simplicity, to have N fully connected layers with size $n \times n$. Typically, batch size b and the number of layers N is much lower than the number of features $b \ll n, \quad N \ll n$, therefore stages with computing Hadamard products and calculating updated weights may be neglected. This relationship is particularly true for deep learning models designed for high-dimensional data, such as CNNs used in computer vision tasks and transformer models employed in natural language processing. In this case, the time for random projection takes $\frac{N-1}{2N}$ that tends to 50%.

However, NN architectures can be more complex. Also the actual number of random projections is Nb, as the OPU performs vector-matrix multiplications thus it has to perform b random projections for each of N layers. These factors result in actual values being significantly lower than 50%, and for some architectures, an increase in training time may be observed. Additionally, Fig. 4 presents numerical estimates obtained using an optical system emulator. The diagram shows that the actual RP part constitutes 32% (50% of 64%) which is lower that estimated 50%. The complete results are provided in the Table 2.

In summary, the maximum theoretical RP component in the algorithm is 50%, which corresponds to x2 acceleration on an ideal OPU. However, the conditions required to achieve this maximum may not always be met, potentially resulting in a lower actual RP component. The example above illustrates a scenario where the actual RP component is 32%, leading to a x1.5 speedup that is still far from the desired x10 or greater acceleration.

Evaluating DFA solely on a per-iteration hardware speedup presents an incomplete picture of training efficiency. Recent analysis [34] demonstrates that DFA scales significantly worse than BP in terms of end-to-end quality: DFA requires a much larger number of iterations to reach the same accuracy as BP. Crucially, [34] noted that even under the most optimistic hardware assumptions—where a specialized coprocessor drastically reduces the computational overhead of DFA—the algorithmic inefficiency remains too profound. Therefore, even if our OPU were to achieve its maximum theoretical speedup, the exponential increase in required total compute resources ultimately prevents DFA from outperforming BP in large-scale training.

Optics EXPRESS



a) WHOLE TRAINING CYCLE **b) BACKWARD PASS**

Fig. 4. Time consumption analysis of the DFA training algorithm. The analysis was conducted on a neural network with two fully connected layers: FC1 (784x128) and FC2 (128x10), incorporating a ReLU activation function. a) WHOLE TRAINING CYCLE, divides the total training time into two main components. The backward pass consumes the majority of the time at 64%, while all other operations, account for the remaining 36%. This highlights that the process of error propagation and subsequent weight updates is the most computationally intensive part of the training cycle. b) BACKWARD PASS provides a breakdown of the time spent within the backward pass itself. This process is split evenly between RP and all other operations, with each consuming 50% of the time.

4.3. Discussion

We evaluated the feasibility and efficiency of OPU for algorithms involving RP. A key criterion was established: the time consumed by RP in a conventional algorithm must exceed the execution time of other parts by an order of magnitude. Among the four algorithms analyzed—RSVD, DFA, RC, and NEWMA—only the latter two met this criterion under ideal conditions. However, practical limitations, such as the dimensionality of data, often prevent achieving the expected performance gains.

It was determined that, assuming an ideal Optical Processing Unit, the random projection operation consumes approximately 50% of the computational resources for the DFA and RSVD algorithms. This corresponds to a maximum performance gain of two times. For the NEWMA and Reservoir Computing algorithms, the proportion of computation attributed to random projection asymptotically approaches 100%. This exceeds the threshold for an order-of-magnitude improvement, which is achieved at 90%, and potentially surpasses the second-order improvement threshold of 99% and more.

However, achieving such acceleration in practical tasks appears challenging. For DFA, the contribution of RP was approximately 30%, for RSVD it was no more than 20%, for NEWMA approximately 50% at the dimensions of the original validation experiments [8], and in Reservoir Computing, the proportion was 75%. While the asymptotic RP fraction in NEWMA approaches unity for sufficiently large N, this regime exceeds the dimensions used in published NEWMA experiments by orders of magnitude. All four algorithms therefore lie within a factor of two to four of one another in terms of the realized OPU speedup at practical problem sizes, despite their differing asymptotic limits.

Another key limitation is that OPU operations require extremely high-dimensional vectors to achieve significant computational advantages. Specifically, for an OPU to exceed the computational speed of conventional devices in a single squared matrix multiplication, vector sizes must be approximately 3×10^6 or larger. This condition is rarely met in practical applications, significantly constraining the applicability of OPUs in real-world scenarios.

Furthermore, while theoretical acceleration exceeding tenfold was observed in two of the four analyzed algorithms under ideal conditions, the actual performance benefits remain limited by architectural constraints. In particular, DFA performance improvements were restricted by the

Optics EXPRESS

structure of NNs and inefficiencies in current OPU-based random projection implementations. These limitations suggest that while OPU can offer computational advantages in specific high-dimensional settings, their broader adoption remains uncertain without significant advancements in hardware and algorithmic integration.

5. Conclusions

In summary, while OPUs demonstrate potential for selected high-dimensional applications, their general applicability is constrained by fundamental limitations. The requirement for extremely high-dimensional vectors, the restricted scope of algorithms benefiting from random projection method, and the inefficiencies in practical implementations diminish their overall utility. The current state of OPU technology does not justify widespread adoption across the evaluated algorithms. Future advancements in both hardware design and algorithmic frameworks may expand their practical relevance, but at present, the advantages of OPU do not outweigh their limitations in the analyzed scenarios. This study provides a foundation for further research into optimizing hardware and algorithm designs to fully harness the capabilities of optical processing.

Funding. Ministry of Economic Development of the Russian Federation (agreement identifier 000000C313925P4H0002; grant No. 139-15-2025-012).

Disclosures. The authors declare no conflicts of interest.

Data availability. The data and code that support the findings of this study are openly available in the "Random-Projections" repository at [35].

References

1. D. Melanson, M. A. Khater, M. Aifer, *et al.*, "Thermodynamic computing system for AI applications," Nat. Commun. **16**(1), 3757 (2025).
2. M. Mousa and M. Nouh, "Parallel mechanical computing: Metamaterials that can multitask," Proc. Natl. Acad. Sci. U. S. A. **121**(52), e2407431121 (2024).
3. Lightmatter, "Envise," lightmatter.co (2025), https://lightmatter.co/products/envise.
4. Optalysys, optalysys.com (2025), https://optalysys.com/.
5. C. Brossollet, A. Cappelli, I. Carron, *et al.*, "LightOn Optical Processing Unit: Scaling-up AI and HPC with a Non von Neumann co-processor," arXiv (2021).
6. T. W. Hughes, M. Minkov, Y. Shi, *et al.*, "Training of photonic neural networks through in situ backpropagation and gradient measurement," Optica **5**(7), 864–871 (2018).
7. N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions," SIAM Rev. **53**(2), 217–288 (2011).
8. N. Keriven, D. Garreau, and I. Poli, "NEWMA: A New Method for Scalable Model-Free Online Change-Point Detection," IEEE Trans. Signal Process. **68**, 3515–3528 (2020).
9. M. Rafayelyan, J. Dong, Y. Tan, *et al.*, "Large-Scale Optical Reservoir Computing for Spatiotemporal Chaotic Systems Prediction," Phys. Rev. X **10**, 041037 (2020).
10. A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems* (2016), pp. 1045–1053.
11. S. Dasgupta and A. Gupta, "An elementary proof of a theorem of Johnson and Lindenstrauss," Random Struct. Algorithms **22**(1), 60–65 (2003).
12. A. Saade, F. Caltagirone, I. Carron, *et al.*, "Random projections through multiple optical scattering: Approximating kernels at the speed of light," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2016), pp. 6215–6219.
13. R. Ohana, D. Hesslow, D. Brunner, *et al.*, "Linear optical random projections without holography," Opt. Express **31**(16), 25881–25888 (2023).
14. V. Strassen, "Gaussian elimination is not optimal," Numer. Math. **13**(4), 354–356 (1969).
15. Nvidia, "NVIDIA H100," nvidia.com (2025), https://www.nvidia.com/en-us/data-center/h100.
16. Nvidia, "NVIDIA GB200-NVL72," nvidia.com (2026), https://www.nvidia.com/en-us/data-center/gb200-nvl72/.
17. J. Kaplan, S. McCandlish, T. Henighan, *et al.*, "Scaling Laws for Neural Language Models," arXiv (2020).
18. Y. Xie, M. Wang, and A. Thompson, "Sketching for Sequential Change-Point Detection," arXiv (2015).
19. J. Launay, I. Poli, F. Boniface, *et al.*, "Direct feedback alignment scales to modern deep learning tasks and architectures," in *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS)* (2020), paper 784.
20. G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the Spring Joint Computer Conference*, AFIPS '67 (1967), pp. 483–485.

Optics EXPRESS

21. V. W. Lee, C. Kim, J. Chhugani, *et al.*, "Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU," in *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA)* (2010), pp. 451–460.
22. M. Mozaffari, K. Doshi, and Y. Yilmaz, "Self-Supervised Learning for Online Anomaly Detection in High-Dimensional Data Streams," Electronics **12**(9), 1971 (2023).
23. M. Mozaffari, K. Doshi, and Y. Yilmaz, "Online Multivariate Anomaly Detection and Localization for High-Dimensional Settings," Sensors **22**(21), 8264 (2022).
24. M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," Comput. Sci. Rev. **3**(3), 127–149 (2009).
25. K. Greff, R. K. Srivastava, J. Koutník, *et al.*, "LSTM: A Search Space Odyssey," IEEE Trans. Neural Netw. Learn. Syst. **28**(10), 2222–2232 (2017).
26. P. R. Vlachas, J. Pathak, B. R. Hunt, *et al.*, "Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics," Neural Netw. **126**, 191–217 (2020).
27. D. J. Gauthier, E. Bollt, A. Griffith, *et al.*, "Next generation reservoir computing," Nat. Commun. **12**(1), 5564 (2021).
28. D. Han and H.-J. Yoo, "Efficient Convolutional Neural Network Training with Direct Feedback Alignment," arXiv (2019).
29. M. Refinetti, S. d'Ascoli, R. Ohana, *et al.*, "Align, then memorise: the dynamics of learning with feedback alignment," in *International Conference on Machine Learning* vol. 139 (2021) pp. 8925–8935.
30. J. Launay, I. Poli, and F. Krzakala, "Principled Training of Neural Networks with Direct Feedback Alignment," arXiv (2019).
31. B. Crafton, A. S. Parihar, E. Gebhardt, *et al.*, "Direct Feedback Alignment With Sparse Connections for Local Learning," Front. Neurosci. **13**, 525 (2019).
32. F. Bacho and D. Chu, "Low-variance Forward Gradients using Direct Feedback Alignment and momentum," Neural Netw. **169**, 572–583 (2024).
33. Z. Wang, K. Müller, M. Filipovich, *et al.*, "Streamlined optical training of large-scale modern deep learning architectures with direct feedback alignment," arXiv (2025).
34. M. J. Filipovich, A. Cappelli, D. Hesslow, *et al.*, "Scaling Laws Beyond Backpropagation," arXiv (2022).
35. "Random-Projections repository," (2025), https://github.com/tiny-engineery/Random-Projections.
36. B. Çarpınlıoğlu and B. Teğin, "U. Genetically programmable optical random neural networks," Commun. Phys. **8**(1), 349 (2025).
37. F. N. Kilinç and U. Teğin, "Self-optimizing multichannel optical computing," arXiv (2026).