

# Training of Physical Neural Networks

Ali Momeni<sup>1</sup>, Babak Rahmani<sup>2</sup>, Benjamin Scellier<sup>3</sup>, Logan G. Wright<sup>4</sup>, Peter L. McMahon<sup>5</sup>, Clara C. Wanjura<sup>6</sup>, Yuhang Li<sup>7</sup>, Anas Skalli<sup>8</sup>, Natalia G. Berloff<sup>9</sup>, Tatsuhiro Onodera<sup>5,10</sup>, Ilker Oguz<sup>11</sup>, Francesco Morichetti<sup>12</sup>, Philipp del Hougne<sup>13</sup>, Manuel Le Gallo<sup>14</sup>, Abu Sebastian<sup>14</sup>, Azalia Mirhoseini<sup>15,16</sup>, Cheng Zhang<sup>2</sup>, Danijela Marković<sup>17</sup>, Daniel Brunner<sup>8</sup>, Christophe Moser<sup>11</sup>, Sylvain Gigan<sup>18</sup>, Florian Marquardt<sup>6</sup>, Aydogan Ozcan<sup>7</sup>, Julie Grollier<sup>19</sup>, Andrea J. Liu<sup>20</sup>, Demetri Psaltis<sup>21</sup>, Andrea Alù<sup>22,23</sup>, and Romain Fleury<sup>1,\*</sup>

<sup>1</sup>Laboratory of Wave Engineering, School of Electrical Engineering, Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland

<sup>2</sup>Microsoft Research, 198 Cambridge Science Park, CB4 0AB Cambridge, UK

<sup>3</sup>Rain AI, San Francisco, USA

<sup>4</sup>Department of Applied Physics, Yale University, CT, USA

<sup>5</sup>School of Applied and Engineering Physics, Cornell University, Ithaca, New York 14853, USA

<sup>6</sup>Max Planck Institute for the Science of Light, Staudtstraße 2, 91058 Erlangen, Germany

<sup>7</sup>Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095, USA

<sup>8</sup>FEMTO-ST Institute/Optics Department, CNRS & University Bourgogne Franche-Comté, Besançon Cedex 25030, France

<sup>9</sup>Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK

<sup>10</sup>NTT Physics and Informatics Laboratories, NTT Research, Inc., Sunnyvale, USA

<sup>11</sup>Applied Photonics Devices (LAPD), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

<sup>12</sup>Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy

<sup>13</sup>Univ Rennes, CNRS, IETR – UMR 6164, F-35000 Rennes, France

<sup>14</sup>IBM Research Europe– Zurich, 8803 Rüschlikon, Switzerland

<sup>15</sup>Department of Computer Science, Stanford University, USA

<sup>16</sup>Google DeepMind, 1600 Amphitheatre Parkway Mountain View, CA 94043.

<sup>17</sup>Unité Mixte de Physique CNRS/Thales, CNRS, Thales, Université Paris-Saclay, Palaiseau, France

<sup>18</sup>Laboratoire Kastler Brossel, Sorbonne Université, École Normale Supérieure, Collège de France, CNRS UMR 8552, Paris, France

<sup>19</sup>Laboratoire Albert Fert, CNRS, Thales, Université Paris-Saclay, Palaiseau 91767, France

<sup>20</sup>Department of Physics and Astronomy, University of Pennsylvania, Philadelphia, PA, 19104, USA

<sup>21</sup>Optics Laboratory (LO), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

<sup>22</sup>Photonics Initiative, Advanced Science Research Center, City University of New York, New York, NY, 10031, USA

<sup>23</sup>Physics Program, Graduate Center, City University of New York, New York, NY, 10016, USA

\*E-mail: romain.fleury@epfl.ch

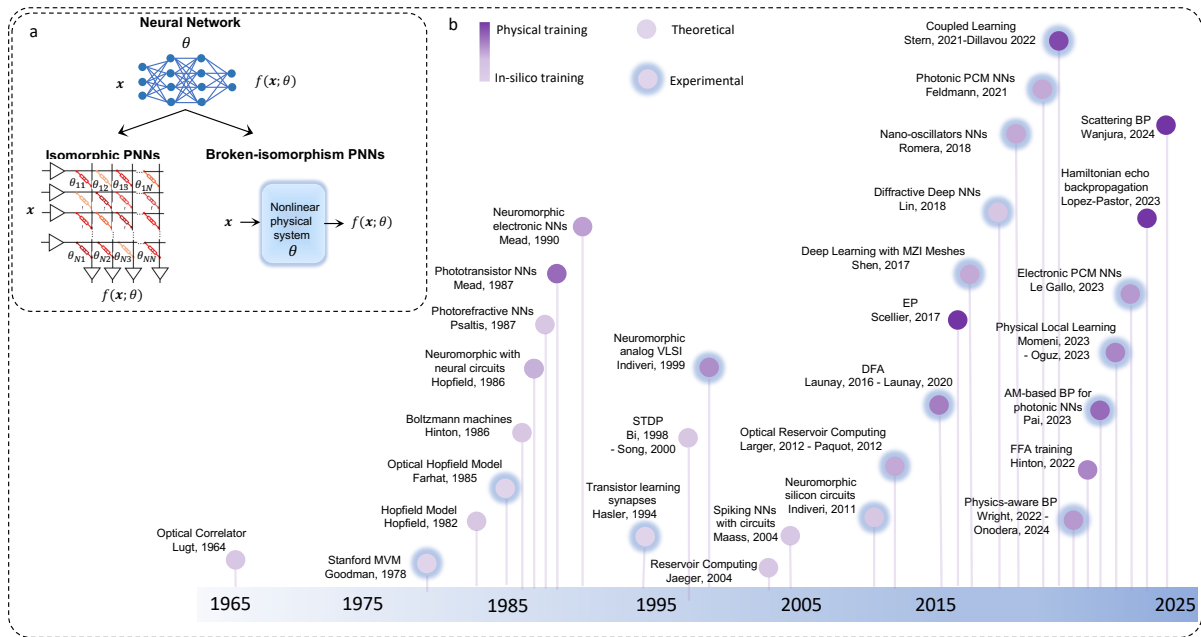
## ABSTRACT

Physical neural networks (PNNs) are a class of neural-like networks that leverage the properties of physical systems to perform computation. While PNNs are so far a niche research area with small-scale laboratory demonstrations, they are arguably one of the most underappreciated important opportunities in modern artificial intelligence (AI). Could we train AI models 1000x larger than current ones? Could we do this and also have them perform inference locally and privately on edge devices, such as smartphones or sensors?

Research over the past few years has shown that the answer to all these questions is likely “textit{yes}, with enough research”: PNNs could one day radically change what is possible and practical for AI systems. To do this will however require rethinking both how AI models work, and how they are trained – primarily by considering the problems through the constraints of the underlying hardware physics. To train PNNs at large scale, many methods including backpropagation-based and backpropagation-free approaches are now being explored. These methods have various trade-offs, and so far no method has been shown to scale to the same scale and performance as the backpropagation algorithm widely used in deep learning today. However, this is rapidly changing, and a diverse ecosystem of training techniques provides clues for how PNNs may one day be utilized to create both more efficient realizations of current-scale AI models, and to enable unprecedented-scale models.

## Introduction

In recent years, artificial intelligence (AI) has profoundly influenced our daily lives through tools such as personal assistant chatbots, and has been utilized in various scientific fields such as healthcare, weather prediction, and material design to tackle some of the world’s most challenging questions. Recent advancements in AI systems have been powered by the digital, silicon-based computing power of Graphics Processing Units (GPUs) as well as the unprecedented abundance of data. AI systems continue to evolve at an accelerated pace. With a clear trend toward increasingly larger models, the reliance on



**Figure 1. a Physical Neural Networks (PNNs)**, processing input data  $\vec{x}$  using trainable parameters  $\vec{\theta}$ . PNNs can be constructed to realize computations isomorphic to those commonly found in artificial neural networks, such as matrix-vector multiplications, or can sacrifice isomorphism for potential speed/energy advantages, where the physical system is left to perform the computation it most naturally performs. **b Timeline of training methods for PNNs**. The corresponding references of selected key milestones and publications from left to right:<sup>3–38</sup>.

traditional digital GPUs is becoming untenable. The primary concerns are the high energy consumption (number of operations per second per Watt), low throughput (number of samples processed per second), and high latency caused by the separation of the memory and processing unit during the training and inference phases of AI systems. Given the widening gap between, on the one hand, the rapid increase in floating-point computation required for AI training and the slower improvements in computing hardware traditionally predicted by Moore’s Law, and on the other hand the low data transfer rates between the memory modules and the computational cores, there has been renewed interest in alternative computing platforms, such as optical, photonics, and analog electronics. We collectively refer to these unconventional computing platforms as analog physical neural networks (PNNs). Previous reviews have concentrated on the domain-specific technological advancement, applications and inference capabilities of PNNs majorly in optics<sup>1</sup> and electronics<sup>2</sup>, among other media. In this article, we aim to explore the development of these PNNs from a training perspective, as broadly as possible, from the ground up and agnostic to the domain. We review methods that employ backpropagation—the learning mechanism most commonly used for digital-electronic neural networks. Additionally, we investigate approaches that minimise digital-electronic computation and leverage the inherent dynamics of the system to learn parameters of the analog system. Finally, we examine local training algorithms that minimize some local objectives which might be easier to implement without digital-electronic processing. Finally, we discuss the application of analog computers in handling larger models and the strategies for efficient training.

## Historical Overview of Analog Computing and PNNs

Artificial neural networks (ANNs) were originally used to model biological neural networks starting in the 1930s<sup>39</sup>. The field gained momentum with the invention of the perceptron by Warren McCulloch and Walter Pitts in 1943<sup>39</sup>, followed by Frank Rosenblatt’s hardware implementation in 1957<sup>40</sup>. These

developments marked a shift towards using ANNs in machine learning, gradually deviating from their biological origins.

A natural evolution from the perceptron is the adaptive linear neuron classifier, or ADALINE. A key difference from the perceptron is that the linear activation function is used to train the weights, while the step function is only used during the inference phase<sup>41</sup>. Hebbian Learning Rule, also known as the Hebb Rule, was proposed by Donald O. Hebb<sup>42</sup>. Hebb's rule can be described as a method of determining how to adjust the weights of a model. According to this principle, the weight between two neurons increases if the two neurons activate simultaneously and decreases if they activate separately. Spike Timing-Dependent Plasticity (STDP)<sup>13,14</sup> is a temporally asymmetric form of the Hebb Rule that is triggered by tight temporal correlations between the pre- and postsynaptic neurons' spikes. This type of local and event-based learning requires no extra energy for non-local transmission, as is needed during the training of ANNs. Based on this principle, spiking neural networks (SNNs), which have been proposed and implemented on neuromorphic platforms, may be more energy-efficient than ANNs<sup>12,15,17,18</sup>. Another milestone was the associative model proposed by Hopfield<sup>5</sup>. Hopfield's approach illustrates the method of collecting and retrieving memories based on the system's attributed energy. The memories correspond to the minima of the energy attributed to the system. In the early 1980s, two other promising learning procedures for deep neural networks were proposed. One was Boltzmann Machines<sup>7</sup> which performed unsupervised contrastive learning. Another was backpropagation<sup>43</sup> which is now a widely used learning algorithm for training deep neural networks.

Despite the rapid advancement of artificial neural networks (ANNs) in digital processors, there has now been a renewed interest in implementing these networks in analog systems (see Figure 1). This idea was initially pursued in optical systems, such as holographic gratings<sup>44</sup>, and in electronic systems with bipolar transistors<sup>10</sup> as well as memristive-crossbar neural networks<sup>45</sup>. Reservoir computing (RC) and Extreme Learning Machines (ELM) are two other historically important computational frameworks widely implemented in various fields<sup>46</sup>. These frameworks treat PNNs as a non-trainable black-box that non-linearly transforms input data. The PNN may have rich recurrent dynamics and thus can have "memory" for time-dependent tasks (RC) or it may simply be a pure feed-forward system suitable for static input data (ELM). RC was first introduced under the names "echo-state network"<sup>47</sup> and "liquid state machine"<sup>48</sup>. In RC, input data is transformed through a high-dimensional network of interconnected nonlinear nodes—the reservoir. Usually, the RC output consists of a linear combination of node responses, and importantly, only the weights of this final output layer are trained, making it resource-efficient and reducing complexity by maintaining input and internal coupling weights constant. More importantly, because only a simple output layer is trained, RC and ELMs can be thought of as the simplest trainable framework for PNNs. Indeed, the reservoir can be any non-linear, sufficiently high-dimensional physical system. Neurons are embedded in the physical system's degrees of freedom, giving RC unprecedented flexibility regarding physical implementation. Internal coupling is usually realized through the physical system's inherent interactions, and input weights correspond to coupling the system to an external signal that drives the system's various dimensions. Implementing the output layer is crucial and can be realized either in hardware i.e. online with physical devices or in software i.e. offline via digital weights that combine previously measured neuron states. RC and ELMs have been implemented on a wide variety of physical substrates ranging from electronics<sup>49</sup> and spintronics<sup>50</sup>, optics for using a time-multiplexed approach<sup>51</sup>, frequency<sup>52,53</sup> and spatial<sup>54,55</sup> multiplexing, to exciton-polariton condensates<sup>56</sup> and mechanical substrates<sup>57</sup>. Recently, there have also been efforts for quantum RC proposals and implementations to leverage the exponential scaling of the Hilbert space as a high-dimensional mapping for information processing<sup>58–65</sup>. Most physical implementations of RC leverage software weights for their high flexibility. Indeed, they provide single-shot learning via a simple matrix inversion, the ability to leverage existing sensory data, particularly in the field of robotics and edge computing<sup>57,66</sup> and easier

integration with follow-up models that would leverage RC as an efficient physical pre-processor<sup>67</sup>. In contrast, the implementation of hardware weights to fully exploit capabilities of the physical system such as inference speed and latency<sup>54</sup>, hardware weights also offer potentially substantial energy savings and scaling as an in-memory alternative to their software counterpart and require hardware-compatible training strategies<sup>68,69</sup>. Moreover, depending on the physical substrate, additional challenges may arise, particularly when it comes to memory-dependent tasks. These tasks imply that input information should drive the physical system at a rate in accordance with its own intrinsic timescales, which can be challenging for ultra-fast systems and entails fast modulation bandwidths.

### Box1: PNNs

Physical Neural Networks (PNNs) are defined as physical systems involving weights ( $\theta$ ) that can be adjusted in order to learn and perform a desired computing task. PNNs resemble neural networks, however at least part of the system is analog rather than digital, meaning that part or all the input/output data is encoded continuously in a physical parameter, and the weights can also be physical, with the ultimate goal of surpassing digital hardware in performance or efficiency. PNNs are divided into two categories, depending on whether or not they mimic digital neural networks: isomorphic PNNs and broken-isomorphism PNNs, respectively. Isomorphic PNNs perform mathematical transformations by designing hardware for strict, operation-by-operation mathematical isomorphism, such as memristor crossbars for performing matrix-vector multiplications (see Fig. 1a). In contrast, broken-isomorphism PNNs break mathematical isomorphism to directly train the hardware's physical transformations<sup>30</sup>. One complication with broken-isomorphism PNNs is that it is often unknown what features are required for universal computation or universal function approximation. The notion of trainable broken-isomorphism PNNs emerged, in part, from untrained physical systems being used for machine learning: physical reservoir computing<sup>46</sup>. There were also several theoretical proposals<sup>70-72</sup> of broken-isomorphism PNNs prior to the general framework and experimental demonstrations presented in Ref.<sup>30</sup>. Broken-isomorphism PNNs could potentially perform certain computations much more efficiently than digital methods, leading to a path for more scalable, energy-efficient, and faster machine learning. Minimizing the power consumption of a PNN will generally result in a reduction of the signal-to-noise ratio in the PNN, even more strongly motivating the need to deal with noise. One strategy is to treat the stochasticity of the physical system as a resource that can be harnessed in machine-learning applications that are fundamentally probabilistic, such as generative models<sup>73,74</sup> or Bayesian models<sup>75</sup>. Alternatively, one can train the PNN to perform deterministic inference tasks in a way that is resilient to noise<sup>76</sup>, even in the limit where the signal-to-noise ratio is low ( $\sim 1$ ) and quantum noise dominates<sup>77</sup>.

## Training Techniques of PNNs

### In-Silico Training

In-silico methods for training PNNs involve digitally emulating and optimizing the physical degrees of freedom ( $\theta$ ) of the hardware, followed by deploying this optimized physical architecture<sup>25,70,78-86</sup> during the inference phase, as summarized in box2. These methods employ physics-based forward models and/or digital neural networks to create digital twins of PNNs within a computer environment, which are optimized for a specific task. After this optimization process, the resulting PNN hardware is deployed for the analog processing of new data.

In-silico training methods enable rapid exploration, validation and testing of various PNN architectures, helping to improve the accuracy and functionality of PNNs before they are physically constructed. This approach is notably faster and cost-effective, eliminating the need to set up and optimize expensive and time-consuming physical systems for each iteration of the design; this also allows scalability, where the PNN architecture can be adjusted and expanded as needed. When the hardware to be deployed does not have significant nonlinearities or device defects, which would require more complex and specific information to be included during training, an approximate digital model with Gaussian noise injected during the forward pass can be sufficient in many cases to obtain accurate inference results on hardware<sup>83,85</sup>. In-silico training also ensures scientific reproducibility and transparency, which is an important advantage compared to some of the delicate and expensive in-situ learning systems that are harder to replicate. Lastly, in-silico methods enable the exploration of theoretical/Gedanken models and PNN structures that

are beyond the capabilities of current technological constraints (e.g., fabrication resolution/precision, material properties, optical nonlinearities, losses, etc.) and help us analyze the effects of various factors in a controlled environment.

However, in-silico training comes with its own set of limitations, as emphasized in Box 2. One of the challenges is related to the digital forward model of the physical architecture. The physical/hardware complexity of the PNN architecture might pose a challenge in identifying appropriate analytical and/or numerical models to digitally represent it accurately, which can limit the effectiveness of in-silico learning. Digital forward models might fail to encompass all physical phenomena in the actual PNN hardware, such as detection noise, misalignments, fabrication and material imperfections<sup>30,34</sup>, among other experimental factors; this forms a challenge for the accurate deployment of these trained PNNs at large scale, covering many devices. The computational demands of these forward model simulations form another potential hurdle. The process of discretizing the continuous physical world requires finer grids for improved accuracy, which can lead to exponential increases in computational requirements with the scaling of the physical size of the PNN and its input/output channels<sup>87,88</sup>. Another limitation is that this method can only be as fast and efficient as digital computers. It will typically be much less efficient than training conventional digital neural networks since modeling PNN hardware will come with computational overhead.

### **Physics-aware BP Training**

Transitioning from gradient-free in-situ optimization to the favorable scaling of hybrid in-situ–in-silico backpropagation algorithms<sup>89–91</sup> has been a critical step for the emerging field of PNN training. Physics-aware Training<sup>30</sup> (PAT) crystallizes the notion that this can be reliably done for any physical system with an approximate predictive model. In this algorithm, the physical system performs the forward pass, and the backward pass is performed by differentiating the digital model. Its key feature, a mismatched forward and backward pass, which is shared with many training algorithms<sup>89–94</sup>, requires only that the digital model produce an estimated gradient that is approximately aligned with the true gradient<sup>92</sup>. This condition, which is much less stringent than requiring a perfect digital model, allows PAT in many cases to be a drop-in replacement for in-silico training, with many of the benefits of in-situ training algorithms. Demonstrating its versatility, PAT has successfully trained PNNs across various domains including optical, mechanical, and electronic systems<sup>30</sup>.

Physics-aware training is a hybrid of in-situ and in-silico methods, inheriting strengths and weaknesses from both. Owing to its in-situ component, PAT mitigates the effect of experimental noise and mismatches between the experiment and the digital model. Meanwhile, its in-silico nature enables accurate training with time scaling similar to backpropagation. However, training may be slow if the physical system's parameters can only be updated slowly. Although constructing a digital model for PAT is less demanding than for pure in-silico methods, it still poses challenges for complex, large-scale PNNs. Given the difficulties in constructing accurate physics-based models (see In-Silico Training Section), the emerging field of physics-informed machine learning, which integrates data-driven methods with physical principles, offers a promising solution<sup>95,96</sup>. In this vein, an optical wave simulation model with data-driven fine-tuning was used to perform PAT on a complex PNN with 10,000 parameters in Ref.<sup>31</sup>.

## Box2: Inference and Training Processes in PNNs

PNNs embody the cutting-edge convergence of physical hardware, material science and artificial intelligence, operating in a two-step process: training and inference. In the training phase, a PNN learns from data related to a specific inference task, adjusting its physical degrees of freedom ( $\theta$ ) based on the corresponding feedback to minimize the difference between its outputs and ground truth. Central to the training of PNNs is the use of error backpropagation, a method used to calculate the gradients of a desired loss function with respect to network weights. This process can be encapsulated by modeling the PNN as a parameterized function  $f_{\text{physical}}$  that relates the input ( $\mathbf{x}$ ) and the physical parameters ( $\theta$ ) of the system with the output ( $\mathbf{y}$ ). Given that, for many physical models, there currently exist no straightforward method to extract the weight gradients, a digital twin ( $f_{\text{digital}}$ ) is often used to approximate  $f_{\text{physical}}$ . Stochastic gradient descent, commonly used in deep learning, can be applied to optimize PNNs through the following steps:

- Forward Model Execution:  $\mathbf{y} = f_*(\mathbf{x}; \theta)$

- Loss Computation:  $L = l(\mathbf{y}, \mathbf{y}_{\text{target}})$

- Error Backpropagation and Parameter Update:  $\mathbf{g}_\theta = \frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \theta}; \theta \rightarrow \theta - \mu \mathbf{g}_\theta$

where  $f_*$  denotes either  $f_{\text{digital}}$  or  $f_{\text{physical}}$ , corresponding to in-silico or in-situ training methods, respectively.  $l$  is the loss function,  $\mu$  is the learning rate and  $\mathbf{y}_{\text{target}}$  is the target output (i.e., the ground truth). For forward model execution ( $f_*$ ), both  $f_{\text{digital}}$  and  $f_{\text{physical}}$  can be used<sup>25, 30, 70, 97</sup>; however, for error backpropagation,  $f_{\text{digital}}$  is more commonly employed due to the inherent challenges of applying traditional backpropagation to  $f_{\text{physical}}$ <sup>98, 99</sup>. Despite being challenging, recent advancements have introduced in-situ learning methods that utilize error backpropagation-free training<sup>24, 34, 35</sup> or backpropagation adaptations using the physical forward model of the PNN<sup>33, 100, 101</sup>. Once the training is complete, unlike conventional neural networks that only operate digitally, an optimized PNN physically performs its inference on new data through  $f_{\text{physical}}$  operating in the analog domain. The implementation of PNNs introduces some unique challenges and trade-offs across the training and deployment (blind inference) phases. A key obstacle is the reliance on a digital twin—the mathematical representation of the PNN—which may not fully capture the complexities of the actual physical model, potentially overlooking various factors like fabrication imperfections, misalignments, and detection noise, among others. This might impact the inference accuracy of an in-silico trained PNN when it transitions from  $f_{\text{digital}}$  to  $f_{\text{physical}}$  in the deployment phase<sup>30, 102</sup>. In-situ training of a PNN through  $f_{\text{physical}}$  can circumvent some of these limitations. In either case, a PNN's forward operation necessitates robust stability; for example, temporal variations in  $f_{\text{physical}}$  due to mechanical/physical drifts or temperature fluctuations etc., would hurt both the training and inference phases regardless of which forward model ( $f_*$ ) is used. Arguably, this requirement for stability is one of the most significant challenges of PNN-based information processing and inference for real-world deployment, and it requires the marriage of advanced micro-/nano-fabrication methods along with material engineering and packaging for building resilience against external conditions.

### Feedback Alignment

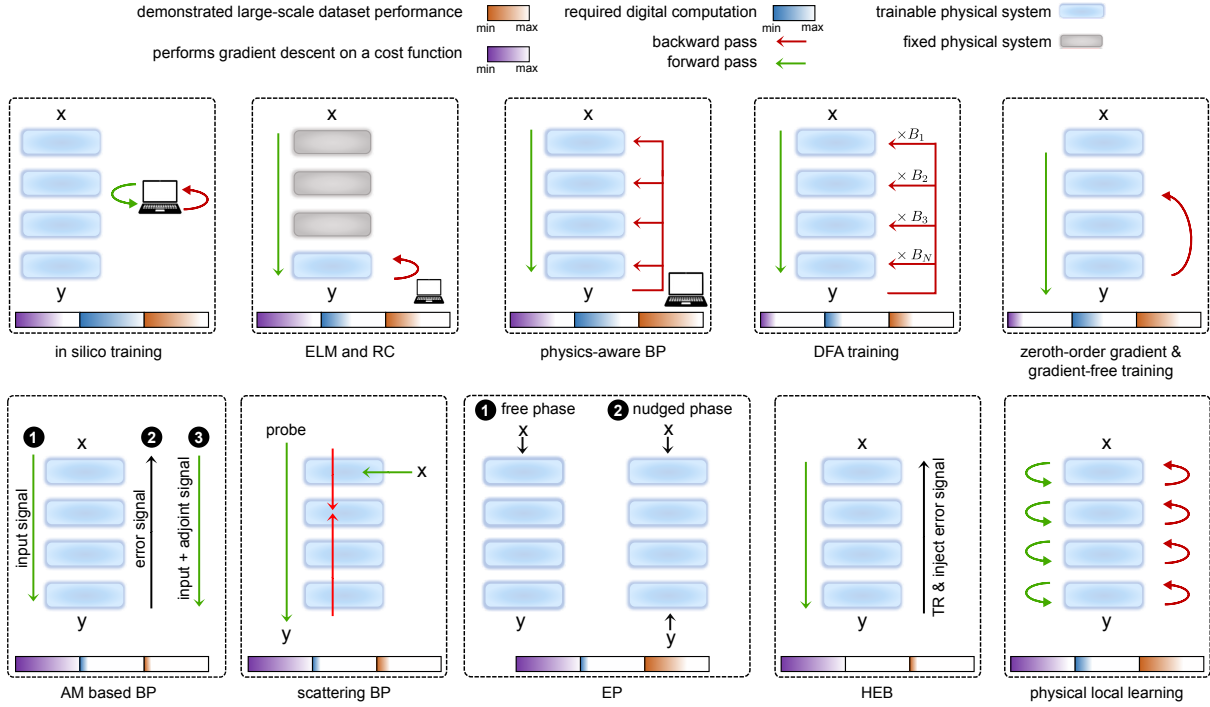
The end-to-end BP faces notable challenges for in-hardware implementations due to gradient communication in the backward pass across all layers<sup>32, 34, 103</sup>. BP uses the transpose of the weight matrix at each layer to back-propagate the error from the output to the input layer, which requires extensive knowledge of the parameters of the NN, and requires that the same weights used in the forward pass are transposed and used in the backward pass, giving rise to "the weight transport problem". In contrast, both Feedback Alignment (FA) and Direct Feedback Alignment (DFA) were introduced as methods that allow training a NN without transferring weights from the forward pass to the backward pass for increased efficiency, usually at the cost of performance. Nevertheless, both methods still require the derivative of activation functions and the states of activation at each layer. FA was first introduced in<sup>104</sup> and presents itself as a simple alternative for training NNs as it uses fixed, random feedback weights at each layer to propagate error signals from the output to earlier layers. This can significantly reduce the computational cost of the backward pass. In<sup>21</sup>, an improved FA algorithm called direct feedback alignment (DFA) was introduced to address the main shortcomings of FA. DFA improves on FA by using fixed random feedback weight matrices to broadcast error signals directly to all layers simultaneously, thereby enabling the successful training of deeper networks. FA methods use fixed random projections to train NNs they are directly more suited to hardware implementations than traditional BP as fixed random projections are easier to implement in physical hardware<sup>22</sup>. Yet, despite its low complexity and apparent compatibility with physical implementation at first glance, physical realizations of DFA are scarce, as it still requires partial knowledge of the NN parameters and, in particular, the activation function of the NN. In<sup>105</sup>, under the name augmented DFA, DFA was extended to no longer require precise knowledge of the network's

activation function and instead replaces it with an arbitrary function that still presents a correlation with said activation function. Augmented DFA was also used to train a neural network architecture utilizing a physical nonlinearity using a Mach-Zehnder modulator. Wang et al.<sup>106</sup> also proposed an asymmetrical estimator based on alignments for in-situ training of photonic neural networks. Additionally, in the context of hardware acceleration for training NNs, a physically implemented DFA algorithm was implemented using high-dimensional random mappings in the optical domain to train digital NNs, using both fully connected and graph convolutional networks in<sup>22</sup>, paving the way for the future use of DFA in training PNN. Despite these advantages, the DFA suffers from accuracy degradation problem. This becomes more serious when the DFA is applied to convolutional and recurrent neural networks<sup>107–109</sup>. Also, this method is only compatible with certain PNNs, where it is possible to separate the nonlinear part and linear layer<sup>105</sup>.

### Physical Local Learning

Another solution to the stagnant gradient problem of end-to-end BP is local learning by eliminating gradient communication entirely. Each layer (or block) independently calculates and applies parameter updates using its own training signal. This setup ensures that no block remains idle, waiting for gradients from others, making it an optimal setting for distributed model optimization. Local objectives were initially used in early unsupervised methods for pre-training deep neural networks, such as the wake-sleep algorithm<sup>110</sup>, Restricted Boltzmann Machine (RBM)<sup>111</sup>, and autoencoders (AE)<sup>112</sup>. The idea behind local training is twofold: one, to minimize a local loss that compresses the information; and two, to extract sufficient information from the input for the next layer/block. These compression and preparation for the next layer could be explained from an information bottleneck perspective<sup>113</sup>. More recently, several variations on the local learning paradigm have been introduced, focusing on parallel training. Löwe et al.<sup>114</sup> use a contrastive predictive loss to perform excellently in an unsupervised setting. Nøkland and Eidnes<sup>115</sup> and Ren et al. and Siddiqui et al.<sup>116,117</sup> consider supervised and self-supervised local learning, respectively; and succeed in matching the accuracy of global learning on classification tasks with up to 100 classes. Some proposals introduced a coupling between subsequent blocks, allowing the gradient signal to flow between pairs of adjacent blocks<sup>118</sup>. This strategy preserves many of the compute efficiency advantages of local optimization while recovering much of the task performance achieved by global optimization<sup>119</sup>. Local parallelism allows for fully asynchronous layer-wise parallelism with a minimal memory footprint. For instance, in reference<sup>114</sup>, the architecture is divided into three independently trainable blocks, resulting in a 2.8 times reduction in GPU memory usage. Generally, the GPU memory will decrease approximately by a factor of  $k$ , where  $k$  represents the number of blocks.

Expanding from digital electronic computing, Oguz et al.<sup>35</sup> utilized the recently proposed contrastive loss-based local learning scheme, forward-forward algorithm (FFA)<sup>32</sup>, to train optical neural networks. This study demonstrates experimentally that multimodal nonlinear optics can significantly improve the performance of multilayer NN architectures without creating additional computational overhead or extensive characterization experiments. Furthermore, Momeni et al.<sup>34</sup> proposed physical local learning (PhyLL). Unlike FFA, PhyLL leverages cosine similarity between two forward passes – one for positive data and one for negative data – eliminating the need for layernorm operation, which can be challenging to implement physically. This approach was evaluated experimentally across three PNNs (Acoustics, Microwave, and Optics), allowing for supervised and unsupervised training without detailed knowledge of the nonlinear physical layer's properties. Another promising avenue is to adapt methods from self-supervised learning<sup>113,120</sup>, which may also be well-suited to training PNNs in a way that avoids gradient communication between layers<sup>103</sup>. A challenge for training arbitrary physical systems with PhyLL is that it uses knowledge of the behavior of each individual layer so that an estimation of the gradient for each layer can be computed and used when updating parameters<sup>121</sup>. More recently, Zhao et al.<sup>122</sup> used a



**Figure 2. Training methods for PNNs.** ELM: Extreme Learning Machine, RC: Reservoir Computing, DFA: Direct Feedback Alignment, EP: Equilibrium Propagation, HEB: Hamiltonian Echo Backpropagation. For a more detailed comparison, refer to Table 1.

Monte-Carlo gradient-estimation algorithm to compute the required gradient for in-situ updating the parameters of PNNs.

While local learning has great potential to scale up in terms of hardware, it remains far from clear whether these methods can, at any scale above small laboratory demonstrations, reproduce the performance of backpropagation. While exactly matching backpropagation is not necessary (especially given the potential for radically improved efficiency), going forward, such guaranteed high-dimensional scaling is an essential requirement for physical local learning techniques.

### Zeroth-Order Gradient and Gradient-free Training

To eliminate the need for detailed knowledge of the physical system whatsoever, model-free, "black-box," or gradient-free training algorithms have been proposed. However, fully-fledged implementations of them in hardware remain scarce. These methods are typically slow because the number of gradient updates scales linearly with the number of learnable parameters in the network, posing a significant challenge for scaling up. These algorithms can be split into two broad categories. On the one hand, perturbative methods estimate the gradient by sampling a target function to optimise (i.e. loss function) at different coordinates (i.e. weights), and after estimating a gradient, weights are optimised via traditional gradient descent. The finite difference method is the simplest way to estimate a gradient by sequentially perturbing each weight and computing its corresponding gradient. More advanced zeroth-order methods have been developed, such as the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm<sup>123</sup>, which perturbs all weights simultaneously. Variants of this algorithm have been used in electronics<sup>124</sup> and optics<sup>125,126</sup>. Although simple, these algorithms have achieved good performance in the context of on-chip in-situ training<sup>127,128</sup>. On the other hand, the second class of gradient-free methods consists of population-based sampling. These include popular classes of algorithms such as genetic algorithms (GA), surrogate optimization (SO), evolutionary strategies (ES), swarm optimization and reinforcement learning (RL) algorithms. These are not directly concerned with achieving an approximation of the



gradient but rather iteratively generating better candidate solutions to an optimization problem either according to heuristic criteria in the case of GA, ES, and swarm-type algorithms or according to an iteratively improved candidate generation policy in the case of RL. Evolutionary strategies such as the CMA-ES<sup>129</sup>, metamodel-based optimization<sup>68</sup>, as well as RL have been used to train optical neural networks in<sup>52,54,130,131</sup>.

### Gradient-Descent Training via Physical Dynamics

Gradient descent optimization is the workhorse of state-of-the-art machine learning systems. We present four physical training methods that achieve gradient descent without needing a digital twin. Such methods can potentially lead to energy gains of 4 orders of magnitude compared to GPU-based neural network training<sup>132</sup>.

The first approach aims at mapping traditional neural networks and BP onto analog hardware. The central insight is that the matrix-vector multiplications required in the forward pass (inference) and backward pass (training) can be implemented using linear reciprocal physical systems<sup>133</sup>, e.g. as a wave propagation through a linear medium in photonics systems<sup>133</sup>, or using memristor crossbar arrays in electrical circuits<sup>80</sup>. In photonic systems, gradients can be efficiently calculated in situ by backpropagating the adjoint electromagnetic field and by interfering its (forward-propagating) time-reversed copy with the original forward field<sup>33,101</sup>. This implementation requires bidirectional propagation of complex amplitude waves (complex field generation modules are required<sup>100,134</sup>) and the (ideally lossless) measurement of the wave intensity in the network (to this aim in photonics almost transparent detectors could be used<sup>135</sup>). This approach requires the network to be lossless, or at least with uniform loss across the network to preserve unitarity. Most often, these photonics and electrical implementations of BP are mixed-signal, executing the nonlinear activation function and its derivative in the digital domain<sup>33,101</sup>. In order to avoid analog-digital and digital-analog converters, and thus further improve energy efficiency, other methods have been proposed to extend physical backpropagation to lossy networks<sup>100</sup> and to nonlinear activation layers<sup>136,137</sup>. However, non-idealities of physical non-linearities will cause deviations between calculated and true gradients.

The second approach is based on nonlinear computation via linear wave scattering<sup>38,138,139</sup> (also used in<sup>34</sup>). Here, input data are encoded in tuneable, physical parameters, such as the frequencies of optical resonators, while other parameters are optimized during training; and the scattering response serves as output of the neuromorphic system. As a significant advantage, all the gradients can be obtained through a minimal number of single-shot scattering experiments<sup>38</sup> without the need for complete knowledge or control of the system.

Equilibrium Propagation (EP) is the third approach<sup>23</sup>. EP applies in energy-based systems, i.e. systems in which physics tends to minimize an “energy” or Lyapunov function. The input is supplied as a boundary condition while physics drives the system to an energy minimum (equilibrium) to produce the response (output). In EP’s original formulation, the weights are updated by a local contrastive rule based on comparing two equilibrium states corresponding to two different boundary conditions. As a major advantage over other contrastive learning algorithms<sup>28,140,141</sup>, EP calculates the weight gradients of arbitrary cost functions<sup>23,142,143</sup>. Examples of energy-based systems trainable by EP include continuous Hopfield networks<sup>23</sup>, nonlinear resistor networks<sup>144</sup>, Ising machines<sup>145</sup> and coupled phase oscillators<sup>146</sup>. Experimental realizations of the contrastive rule pose a challenge since it requires comparing network states under two different sets of boundary conditions. An EP-like contrastive scheme has been successfully implemented in memristor crossbar arrays<sup>132</sup>, and a binary version of EP implemented on D-Wave solved MNIST with software-equivalent accuracy<sup>145</sup>. Both implementations resorted to external memory to store the states between the two phases. Another contrastive scheme called Coupled Learning (CL)<sup>28</sup> has been experimentally demonstrated in elastic networks<sup>147</sup>.

To realize energy gains, however, EP must be implemented in the lab without digital processing or external memory. Potentially scalable laboratory prototypes have been developed of such electrical linear<sup>29</sup> as well as nonlinear<sup>148</sup> resistor networks; these implement CL by coupling two copies of the network<sup>29,148</sup>. Further energy gains can be realized by including the energy as an additional term in the cost function<sup>149</sup>. Other proposed solutions to implementing the contrastive rule resort to encoding the two states in different physical domains<sup>150</sup>, using integral feedback<sup>151</sup>, working in the complex domain<sup>142</sup> or using dynamics of spiking networks<sup>152</sup>. Another conceptual advance is a non-contrastive version of EP where the weights are physically updated through physical equilibration<sup>153</sup>. The potential of EP has further been highlighted in simulations by training energy-based convolutional networks on a down-sampled version of the ImageNet dataset<sup>142</sup>.

Hamiltonian Echo Backpropagation (HEB) is the fourth approach<sup>37</sup>. On top of extracting the weight gradients, HEB directly produces the correct weight updates using physical dynamics, without any feedback. HEB applies in time-reversal invariant Hamiltonian systems where dissipation is (ideally) absent. Another crucial ingredient is a "time-reversal operation", e.g. phase conjugation in nonlinear optics experiments. During training, in the forward pass, a signal wave and a trainable-parameter wave travel jointly through a nonlinear medium where they interact. An error signal is superimposed on the signal wave, and a time-reversal operation sends both waves back through the medium. At the end of this backward pass, the trainable wave has been automatically updated in the direction of the cost-function gradient.

## Continual Learning

Neural networks are usually trained "off-line" on a fixed dataset, and then deployed for inference, without further training. Continual learning aims to enable neural networks to learn from non-stationary streams of data incrementally. It is not a trivial problem: when trained on a new dataset, neural networks tend to lose their previously acquired capabilities by overwriting weights involved in representing the old learning. This problem is known as "catastrophic forgetting"<sup>154</sup>. To address this, research efforts have focused on freezing parts of the network weights while simultaneously growing other parts of the network to extend the learning ability. For example, in class-incremental learning (CIL), the network must incrementally learn to distinguish novel classes without forgetting the previously observed classes<sup>154</sup>. Very recently, few-shot CIL algorithms have been proposed, in which the continual learning of novel classes is done with only a few (e.g. 5) data samples, which is even more challenging<sup>155,156</sup>.

PNNs are excellent choices for implementing few-shot CIL algorithms due to their capacity for continual expansion, allowing them to accommodate new classes effectively. For example, PNNs implemented with arrays of memristive devices can be incrementally expanded by programming previously unused devices by applying suitable electrical pulses, which will retain information about the novel classes in a non-volatile way. Such an implementation was realized based on an enhanced memory-augmented neural network comprising a dynamically growing explicit memory implemented with a phase-change memory (PCM) array<sup>157</sup>. The novel classes were learned and stored incrementally in the explicit memory by exploiting the in-situ progressive crystallization of PCM devices, and an in-memory similarity search was performed during inference in the PCM array to classify unseen examples<sup>157</sup>.

Another opportunity to implement continual learning in PNNs is offered by Ising or XY machines, which are specialized hardware engineered to solve optimization problems. They exploit their inherent ability to discover low-energy states in a spin system naturally<sup>158</sup>. The architecture of such machines, often realized through optical, light-matter, or quantum systems, inherently supports parallel computations, making them suitable for continual dynamic learning. To enable continual learning, the XY or Ising machine's architecture can be modified to dynamically adjust the interactions between spins<sup>159,160</sup>. Such adaptability can be facilitated by developing algorithms that incrementally update the Hamiltonian—the

system's underlying energy function—reflecting the continual integration of new information and the retention of existing patterns while avoiding catastrophic forgetting.

Further work is nonetheless required to implement an entire CIL architecture in PNN hardware to demonstrate fully end-to-end continual learning more efficiently than in a traditional digital architecture.

## Towards Implementation of Analog Efficient Large Models

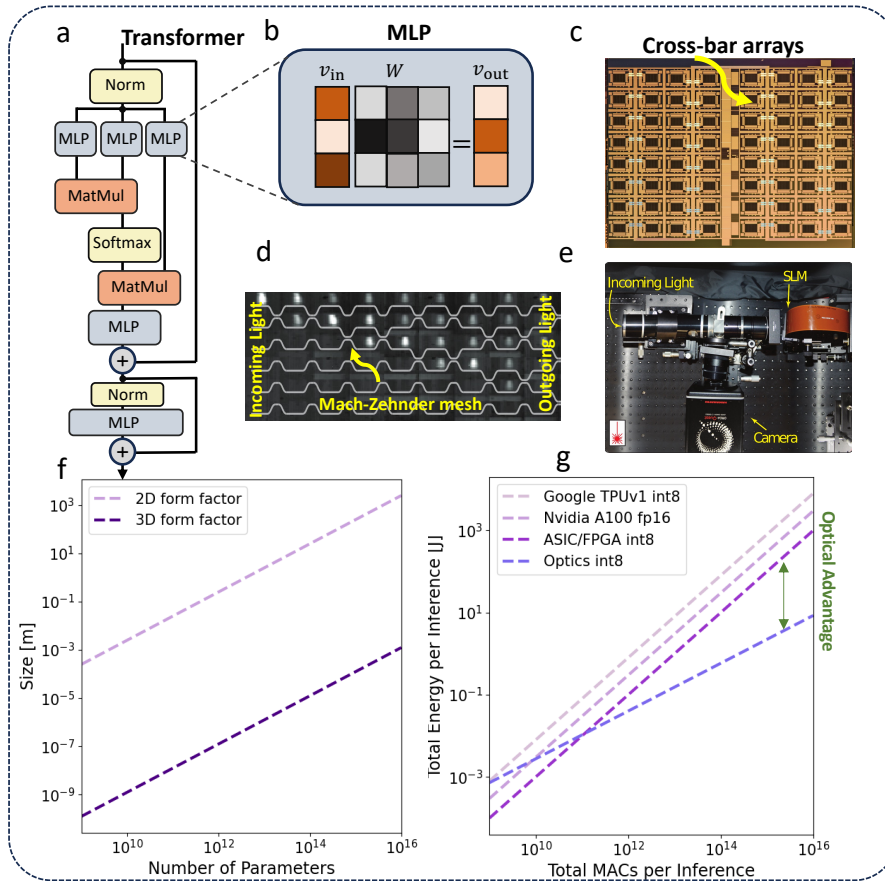
In the next two sections, we first explore large AI models and briefly review strategies to improve their efficiency from a digital standpoint, both in terms of hardware and software innovations. Following that, we consider the potential of "analog" large models, i.e., large AI models implemented by analog PNN hardware. While PNNs are currently far from competitive with digital approaches (which are themselves progressing rapidly), PNNs possess unique physical properties that may make them an important route for scaling AI models beyond the practical limits of conventional digital implementations.

### Efficient Training and Fine-tuning of Large models

In recent years, we have witnessed remarkable improvements in the capabilities of neural networks, especially those designed in rich data setting including language understanding such as GPT-2 (1.2B)<sup>163</sup>, GPT-3 (175B)<sup>164</sup>, LLaMA (65B)<sup>165</sup>, PaLM (540B)<sup>166</sup>, GPT-4<sup>167</sup>, Gemini<sup>168</sup>, vision-language understanding such as CLIP<sup>169</sup> and LLaVA<sup>170</sup>, scientific reasoning including MatterGen<sup>171</sup> and AlphaFold<sup>172</sup>, and climate prediction<sup>173</sup>. Demonstration of emerging abilities in these large models, mainly based on the attention architecture in Fig. 3 a, is driven by "scaling laws"—a trend indicating that a model's ability to generalize improves in a predictable, log-linear manner as a function of number of parameters, data examples, or the amount of compute used to model and train large models<sup>163</sup>. Training frontier LLMs with hundreds of billions of parameters is a prohibitively expensive task that requires weeks (or even months) of optimization on thousands of accelerators (e.g. Nvidia GPUs) and trillions of tokens of data. The inference cost of LLMs is also high due to their large memory footprint, which requires model partitioning across many accelerators (causing I/O overhead), as well as the large number of floating point operations across the model.

To address these challenges, the machine learning research community has been actively exploring strategies to enhance model efficiency without unduly compromising model accuracy. These methods include architectural changes such as moving away from attention-based architectures, quantization of model parameters, fine-tuning methods to avoid costly retraining, and efficient implementation on digital hardware.

One such effort has been on designing sublinear attention mechanisms. Sub-quadratic attention models replace full attention by taking advantage of attention properties such as low-rankness<sup>174</sup> and sparsity<sup>175</sup>. In these models, the softmax operation is replaced by linear<sup>176</sup>, low-degree polynomials<sup>177</sup> or random feature maps<sup>178</sup>. An orthogonal approach is replacing the attention with recurrent models<sup>179</sup>. Another effective approach for performance efficiency is quantization, in which the model weights are transformed to a lower precision (e.g. from 16 bits to 4 bits)<sup>180,181</sup>. Quantization is commonly performed as a post-training step, however, recent methods on quantization-aware training of LLMs show that models with as low as 1.58-bit (ternary parameters) can match the performance of full-precision transformers, saving the energy consumption by tenfold<sup>182,183</sup>. In addition to (pre-)training, fine-tuning is an important phase in LLM optimization process. During fine-tuning, a pre-trained model is adapted to a specific task, using a relatively small dataset (often in the order of thousands of examples or less). Fine-tuning models are commonly done through supervised or reinforcement learning<sup>184,185</sup> and would require updating the entire model parameters. All techniques applied to efficient training are also applicable to fine-tuning. Moreover, there are techniques specifically developed for fine-tuning, such as LoRA<sup>186</sup> and Adapters<sup>187,188</sup>, in which updating only a small subset of the parameters (often those that are newly added) is sufficient to



**Figure 3. Analog large models** (a) The building block of the mainstream large language models is the transformer architecture<sup>161</sup>, whose main building blocks are the attention, multilayer perceptron (MLP) layers, softmax operation and dynamic matrix-vector multiplication (MatMul). The attention layer requires a causal pairwise computations between the elements in the sequence, resulting in a quadratic increase in computational complexity with respect to sequence length, affecting both time and energy overhead, especially as models process longer context lengths. The MLP layer includes very large weight matrices that also impose a large computational overhead; (b) MLP is the architecture of vector-matrix multiplication also known as a fully connected layer. The MLP can be experimentally realized on a number of technologies such as (c) crossbar arrays<sup>36</sup>; (d) Mach-Zehnder Interferometer meshes<sup>33</sup>; (e) free-space multipliers<sup>162</sup>; (f) size scaling of two- and three-dimensional analog models with increasing model parameters computed at wavelength=500 nm with scalings  $\lambda^{2/3}$ ; (g) Energy scaling advantage of analog optical matrix-vector multiplication compared to digital electronics, for Transformer models. Data were obtained from<sup>162</sup>.

adapt the model for downstream tasks. Digital hardware-aware implementations of the exact attention mechanism have also been shown to be very effective. For example, FlashAttention<sup>189</sup> uses tiling and recomputation to reduce the memory bandwidth overhead, Hydragen<sup>190</sup> splits the sequence across the prompt and suffix and batches attention queries over the shared prompt sequences to increase throughput. vLLM<sup>191</sup> avoids redundant storage of the prefix keys and reduces the cost of redundant reads.

### Analog Large Models

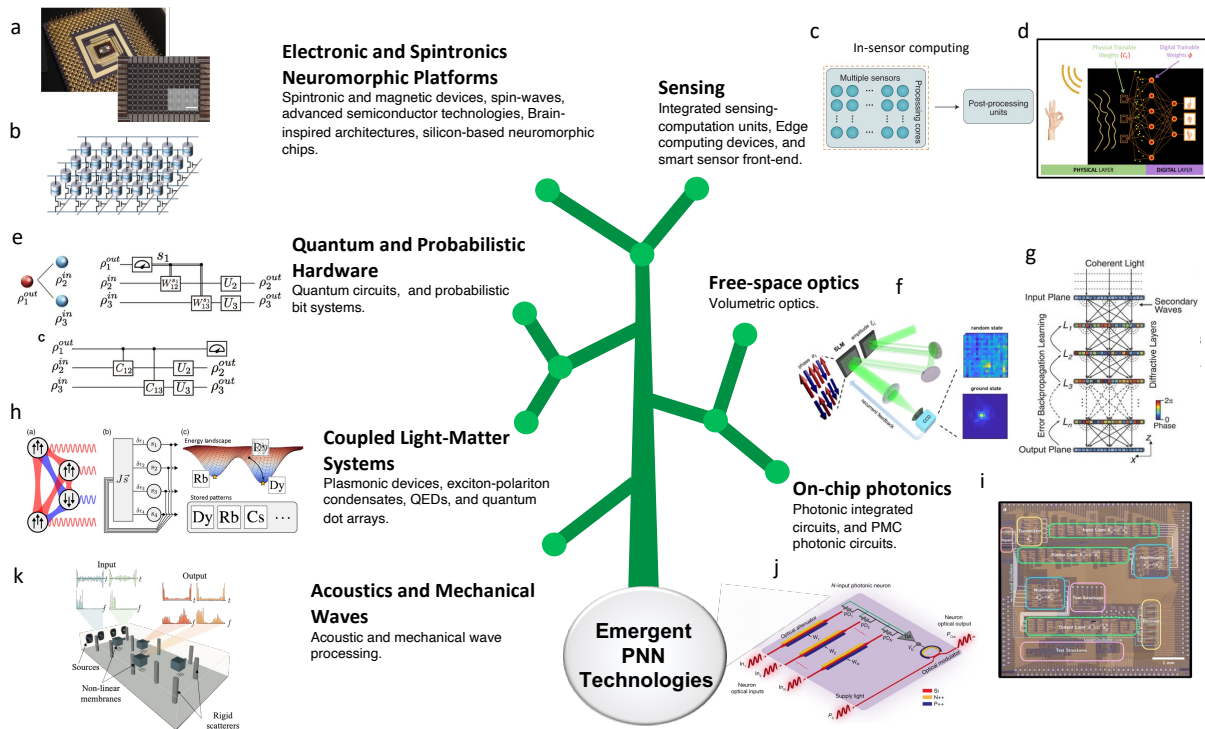
AI models are now large, and they are getting larger. As the previous section summarized, many aspects of these models are likely to change, and digital software and hardware innovations will help reduce their energy costs in both training and inference. Could PNNs make this scaling easier? Could they allow scaling beyond what is economically feasible for digital computers?

As a first consideration, large AI models are literally physically large. In optics, for example, parameters are often encoded in the pixels of spatial light modulators (SLMs). A 1-quadrillion ( $10^{15}$ ) model would require a total area of SLMs of roughly  $16 \text{ m}^2$ , comparable to the cross-section of an elephant. Does this mean that large optical PNNs have no hope? No – in fact for computations of this scale, a large footprint is inevitable with any hardware. However, it does suggest that PNN researchers (across all hardware) will need to propose architectures designed not just for laboratory demos, but for extreme scalability. One initial step are the projections of Anderson *et al.*, which examine the costs of implementing Transformer models using optical hardware<sup>162</sup>. This work confirms that, even when splitting a model across many reasonably-sized optical processing units and accounting for the costs of loading parameters and inputs from memory, plausible optoelectronic hardware could be roughly 100x more efficient than 2023's state-of-the-art digital electronics at implementing current LLMs (about  $10^{15}$  MACs/inference). More importantly, this advantage could grow to  $10^4$  or even  $10^5$  for larger models with  $10^{19}$  MACs/inference.

This highlights perhaps the most important scaling consideration for potential future large-scale PNN AI systems: If PNN hardware is designed properly, its different underlying physics may allow it to exhibit different energy scaling behavior than digital electronics. This means that, given sufficiently large model scale, PNN implementations may offer better efficiency compared to digital systems, even despite the many overhead costs of analog hardware, such as digital-to-analog conversion costs. In optics for example, assuming fixed output precision, the optical-energy cost per operation (scalar MAC) when performing a dot product scales as  $1/N$ , where  $N$  is the length of the vector<sup>24, 192–194</sup>. In digital electronics, the cost per operation is normally fixed ( $\sim 1$ ). This optical dot product energy scaling advantage ( $\sim 1/N$  versus  $\sim 1$ ) may transfer to a similar scaling advantage for AI model inference, since most models consist primarily of dot products. However, this transfer is not guaranteed: the algorithm and data, design of the electronic modulation and memory access, and the optical hardware itself can limit the scaling advantage<sup>162, 195</sup>.

Finally, scaling is not only about hardware. While Transformers are a breakthrough, they are also merely the latest algorithm to rise to prominence because of synergy with scalable hardware<sup>196, 197</sup>. As we look to ultra-scale PNNs, it may be shortsighted to focus on current algorithms – rather, we need new synergistic combinations of hardware and software. Given the inertia of infrastructure and the rapid development of efficient digital large model implementations, commercially viable PNNs will need to provide energy efficiency thousands if not millions of times greater than digital electronic. To do this will require designing physical computers that consider the challenge of scale holistically, hardware and software together, and that make efficiently exploiting physical compute their leading objective<sup>196</sup>.

All the above considers primarily inference of large models, which is the most urgent and accessible opportunity for PNNs in the large-model space. Using PNNs to accelerate training of such models is similarly promising. As this article's earlier sections suggest, the training phase may allow additional physical phenomena to be gainfully exploited, e.g., to realize scalable local learning based on analog physical processes. This means that physics-driven learning of PNN models may exhibit additional scaling



**Figure 4. Emergent technologies.** (a) Optical microscopic image of a memristor crossbar array integrated on the memristor/CMOS chip, from <sup>199</sup>; (b) crossbar array of magnetic tunnel junctions for high-density storage and memory retrieval, from <sup>200</sup>; (c) the schematics of in-sensor computing architecture, from <sup>201</sup>; (d) an operational principle of learned-sensing intelligent meta-imagers, from <sup>202</sup>; (e) an illustration of soft quantum neurons in the quantum circuit model, from <sup>203</sup>; (f) the schematics of spatial photonic Ising machine, from <sup>204</sup>; (g) diffraction optical NNs consisting of multiple transmissive or reflective layers, where each point on a given layer acts as a neuron, with a complex-valued transmission or reflection coefficient, from <sup>25</sup>; (h) superradiance in confocal cavity QED for high-density storage and memory retrieval, from <sup>205</sup>; (i) an image of a PIC with shown signal paths (white) and the local oscillator paths (blue), from <sup>125</sup>; (j) the structure of an N-input photonic neuron with weights of input signals changed using optical PIN attenuators and summed up using photodetectors, from <sup>98</sup>; (k) acoustic data transformer where input data are encoded into the intensity of sound waves at different frequencies that propagate through a random set of membranes, from <sup>34,206</sup>.

advantages (in energy, size, speed, etc.) that extend even beyond the inference energy scaling advantages noted above.

## Emerging PNN Technologies

In the PNN context, quantum, probabilistic, photonic, light-matter and hybrid computing represent promising developments <sup>158</sup>. Quantum computers can exploit features of quantum mechanics such as superposition in a way that might allow them to address optimization problems critical to NN training <sup>198</sup>.

However, the practicality of applying these quantum advantages is tempered by the limitations of current Noisy Intermediate-Scale Quantum systems, which have limited qubits and large computational error rates <sup>207</sup>. Specific quantum algorithms and quantum neural network frameworks are being devised to operate within these constraints, for instance, using ‘soft quantum neurons’ <sup>203</sup>, quantum circuit Born machines, quantum generative adversarial networks <sup>208</sup>, and variational quantum algorithms <sup>209</sup> can potentially outperform classical models in generating new samples and learning data distributions.

Probabilistic hardware systems, which rely on stochastic elements, sometimes referred to as probabilistic bits (p-bits) occupy an intermediate step between quantum and classical deterministic PNNs. They are naturally suited to training deep generative models, specifically deep Boltzmann Machines (DBMs) <sup>210</sup>,

but can also be trained to act as stochastic neural networks for deterministic classification tasks<sup>77</sup>.

Photonic-based optimisers such as Spatial Photonic Ising Machines (SPIMs) that use spatial light modulation to emulate Ising problems introduce another opportunity for PNNs<sup>204</sup>. The properties of light, such as spatial parallelism achievable in optics and the dissipationless dynamics of light propagation that enables computation, offer significant advantages over electronic systems<sup>211</sup>.

Light-matter systems that couple photons to matter particles efficiently<sup>212,213</sup> form the basis of gain-based computing that could lead to new methodologies in PNNs by encoding optimization problems within driven-dissipative systems' gain and loss rates<sup>205,214</sup>. For instance, polaritonic systems with their strong noise-controlling nonlinearities were proposed to lead to all-optical platforms for implementing diffusion models<sup>215,216</sup>.

A limitation of the stand-alone PNNs is the overhead from encoding inputs into and reading outputs from the physical domain. Integrating wave-based computing and PNNs at a sensor front-end naturally avoids such overhead. Intelligent sensors combine sensing and over-the-air computing capabilities to pre-select task-relevant information during data acquisition, yielding substantial improvements in latency and other metrics<sup>202</sup>. Intelligent sensors are to date conceived by training via error backpropagation the entire sensing pipeline end-to-end concerning a specific task, including both analog data acquisition and digital post-processing<sup>217,218</sup>. In other words, training intelligent sensors to date requires a differentiable digital model; alternative training paradigms remain unexplored. The hallmark feature of an intelligent sensor is that its data acquisition is task-aware thanks to the end-to-end training<sup>202</sup>. Endowing the physical layer with programmability and/or non-linearity enables task-reconfigurable smart sensors and/or increases the complexity of mathematical operations that can be performed during data acquisition<sup>99,219–221</sup>. These paradigms are emerging across scales and wave phenomena, and context-aware next-generation wireless networks may already leverage dynamic metasurface antennas for integrated sensing, computing and communications<sup>219,222–225</sup>. Other applications include privacy-preserving cameras<sup>226</sup>, non-destructive testing<sup>227</sup> and noise-adaptive smart computational imagers<sup>228</sup>. A trend toward further integration of wave-based computing with sensing, communications and data storage is clearly emerging<sup>229–231</sup>.

The programming of all-photonic routers in networks also bears significant similarities to the training of PNNs, and these fields can benefit from one another. Irrespective of the detailed implementation<sup>232–236</sup>, programmable all-photonic routers are (usually linear) input-output systems with a multitude of tunable degrees of freedom. The latter must be reconfigured during runtime to realize different routing functionalities (i.e., implement different input-output relations). A further important constraint is that reflections back into the input channels should be suppressed to avoid reflected-power echoes in the network<sup>233</sup>. Besides various well-established global optimization techniques, ideas for progressively configuring specific hardware architectures purely based on local feedback loops are emerging<sup>236,237</sup>, even for only partially coherent light<sup>238</sup>.

Integrating these advanced computational paradigms into PNNs requires addressing several challenges, including adapting learning algorithms to leverage quantum and photonic, wave-based or gain-based processes, managing noise and error rates in quantum systems, and the scalability of architectures. The development of hybrid systems, which combine quantum or photonic processing units with classical computational elements, might offer practical pathways to use the advantages of these technologies while mitigating their current limitations. Aligning the unique properties of these physical systems with the goals of PNNs can pave the way for the next generation of intelligent systems characterized by unprecedented levels of speed, efficiency, and scalability.

Finally, beyond human-made devices, the dynamics of biological systems may offer enticing opportunities to implement new generations of PNNs, e.g., using octopus-inspired soft robotic arms for reservoir computing<sup>57,239</sup>. Biological contexts may require rethinking the training process of PNNs to enable their compatibility with the vulnerability of underlying biological systems.

## Outlook

PNNs may ultimately be found from the data center to the edge, from powering large generative models, through to aiding in classification in smart sensors. In all cases, they will need to be trained, but depending on the application, the constraints on training may be different. For example, large models on servers might only require updates every few months and be able to use a lot of energy to be trained, whereas some models at the edge might need to be adaptive on a timescale of hours or even minutes, and be severely limited in their power budget for retraining. The diversity of PNNs and use cases suggests that the major open challenge for the field is not to find what the single best training method is, but rather what the best training method for each situation is, and what the tradeoffs between different methods are.

An ideal training method would:

- 1) be model-free, not requiring the training procedure to have access to a mathematical description of the behavior of the PNN hardware, and would rely on as few assumptions about the behavior and structure of the PNN hardware as possible, allowing PNN designers to optimize the hardware for speed and energy benefits in inference;

- 2) give speed and energy advantages in the training time and energy cost versus training a conventional artificial neural network to perform the same task with the same accuracy, using the same training data – to achieve such benefits in training, the method should leverage the benefits of the PNN hardware itself in training rather than relying heavily on digital-electronic hardware during training, and the training method should be able to exploit the full expressivity of the PNN hardware;

- 3) be robust to hardware copy-to-copy variations, drift, and noise – and if not fully robust, then at least able to cheaply compensate for these imperfections.

None of the known training methods for PNNs simultaneously satisfy all of these properties, or even perfectly satisfy any one of them. However, the past few years have seen many different training methods be developed that push the boundaries of tradeoffs within this space of properties, and we anticipate further advances that will lead to methods that are simultaneously more general, more efficient, and more robust, enabling practical and widespread use of PNNs.

**Acknowledgements** We thank Jérémie Laydevant, Martin Stein, and Mandar Sohoni for helpful feedback on a draft of this manuscript. R.F. and A.M. acknowledge funding from the Swiss National Science Foundation (SNSF) under the Eccellenza award 181232. R.F. and P.d.H. acknowledge funding from the ANR-SNSF MINT project 212577 entitled “Ultra-compact non-linear metamaterial wave processors for analog deep learning”. P.L.M. acknowledges funding from the National Science Foundation (award CCF-1918549) and a David and Lucile Packard Foundation Fellowship. N.G.B. acknowledges the support from the HORIZON EIC-2022-PATHFINDERCHALLENGES-01 HEISINGBERG project 101114978 and Weizmann-UK Make Connection grant 142568. S.G. is a member of the institut Universitaire de France. T.O., L.G.W, and P.L.M. thank NTT Research for their financial and technical support.

**Competing interests** T.O., L.G.W. and P.L.M. are listed as inventors on a US provisional patent application (number 63/178,318) on physical neural networks and physics-aware training.



**Table 1.** Table for comparing different algorithms:  $N$  is the number of parameters,  $M$  is the number of neurons, and  $T_0$  is time to convergence for backpropagation (BP).

Algorithm	Comments	Memory (Measurements)	Expected wall clock time to convergence	Updatable physical or digital parameters	Digital model (or simulation) required
<b>ELMs and RC</b>	Solution is found after one matrix inversion	$O(M)$	Time to perform matrix inversion	Last digital linear layer	No
<b>Backpropagation-based methods</b>					
<b>In-silico BP</b>		$O(M)$	$O(T_0)$	All digital parameters + physical parameters simulated in digital model	Yes
<b>Physics-aware BP</b>	Reduces constraints on model faithfulness	$O(M)$	$O(T_0)$	All digital parameters + physical parameters simulated in digital model	Yes
<b>Direct Feedback Alignment (DFA) training</b>		$O(M)$	$> O(T_0)$	Matrix elements of a digital or physical matrix-vector multiplier	Only for nonlinear activation function and its derivative
<b>Physical local learning</b>	Need the knowledge of each individual layer for estimating the gradients	$O(M)$	$> O(T_0)$ (depends on the number of layers)	All controllable parameters	No
<b>Zeroth-order/gradient-free methods</b>					
<b>Finite difference stochastic approximation</b>		$O(N)$	$O(N \cdot T_0)$	All controllable parameters	No
<b>Simultaneous perturbation stochastic approximation</b>		$O(N)$	$O(N \cdot T_0)$	All controllable parameters	No
<b>Gradient-free training (GA/etc.)</b>		For Population-Based Methods (e.g., Genetic Algorithms, Evolution Strategies): $O(PN)$ , where $P$ is population size.	$\gg O(T_0)$	All controllable parameters	No
<b>Physical gradient computation/physical backpropagation</b>					
<b>Adjoint Method (AM) based BP</b>	Reduces memory load	$O(1)$	$O(T_0)$	All controllable parameters	Only for nonlinear activation function and its derivative
<b>Scattering BP</b>	Nonlinear computation in physically linear systems; requires knowledge of form of Hamiltonian terms depending on tunable parameters; allows for batch processing using frequency multiplexing	$O(N_{\text{out}}M)$ (with $N_{\text{out}}$ the output dimension)	$O(T_0)$	All controllable parameters	No

**Table 1.** Table for comparing different algorithms (continued).

Algorithm	Comments	Memory (Measurements)	Expected wall clock time to convergence	Updatable physical or digital parameters	Digital model required (Digital model accuracy)
<b>Equilibrium Propagation (EP)</b>	Applies to system converging to the minimum of an energy function. Requires knowledge of energy derivatives wrt trainable parameters.	Lazy implementation: $O(N)$ , Advanced implementation: $O(M)$	$O(1)$	All controllable parameters	No
<b>Hamiltonian Echo Backpropagation (HEB)</b>	Applies to lossless systems with time-reversal operation	$O(0)$	$O(1)$	All controllable parameters	No

## References

1. Wetzstein, G. *et al.* Inference in artificial intelligence with deep optics and photonics. *Nat.* **588**, 39–47 (2020).
2. Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R. & Eleftheriou, E. Memory devices and applications for in-memory computing. *Nat. nanotechnology* **15**, 529–544 (2020).
3. Lugt, A. V. Signal detection by complex spatial filtering. *IEEE Transactions on information theory* **10**, 139–145 (1964).
4. Goodman, J. W., Leonberger, F. J., Kung, S.-Y. & Athale, R. A. Optical interconnections for vlsi systems. *Proc. IEEE* **72**, 850–866 (1984).
5. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. national academy sciences* **79**, 2554–2558 (1982).
6. Farhat, N. H., Psaltis, D., Prata, A. & Paek, E. Optical implementation of the hopfield model. *Appl. optics* **24**, 1469–1475 (1985).
7. Hinton, G. E., Sejnowski, T. J. *et al.* Learning and relearning in boltzmann machines. *Parallel distributed processing: Explor. microstructure cognition* **1**, 2 (1986).
8. Hopfield, J. J. & Tank, D. W. Computing with neural circuits: A model. *Sci.* **233**, 625–633 (1986).
9. Psaltis, D., Brady, D. & Wagner, K. Adaptive optical networks using photorefractive crystals. *Appl. Opt.* **27**, 1752–1759 (1988).
10. Mead, C. A. Neural hardware for vision. *Eng. Sci.* **50**, 2–7 (1987).
11. Mead, C. Neuromorphic electronic systems. *Proc. IEEE* **78**, 1629–1636 (1990).
12. Hasler, P., Diorio, C., Minch, B. & Mead, C. Single transistor learning synapses. *Adv. neural information processing systems* **7** (1994).
13. Bi, G.-q. & Poo, M.-m. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. neuroscience* **18**, 10464–10472 (1998).
14. Song, S., Miller, K. D. & Abbott, L. F. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. neuroscience* **3**, 919–926 (2000).
15. Indiveri, G. Neuromorphic analog vlsi sensor for visual tracking: Circuits and application examples. *IEEE Transactions on Circuits Syst. II: Analog. Digit. Signal Process.* **46**, 1337–1347 (1999).
16. Jaeger, H. & Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science* **304**, 78–80 (2004).
17. Maass, W. & Markram, H. On the computational power of circuits of spiking neurons. *J. computer system sciences* **69**, 593–616 (2004).
18. Indiveri, G. *et al.* Neuromorphic silicon neuron circuits. *Front. neuroscience* **5**, 73 (2011).
19. Larger, L. *et al.* Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. *Opt. express* **20**, 3241–3249 (2012).
20. Paquot, Y. *et al.* Optoelectronic reservoir computing. *Sci. reports* **2**, 287 (2012).
21. Nøkland, A. Direct feedback alignment provides learning in deep neural networks. *Adv. neural information processing systems* **29** (2016).
22. Launay, J. *et al.* Hardware beyond backpropagation: a photonic co-processor for direct feedback alignment. *arXiv preprint arXiv:2012.06373* (2020).
23. Scellier, B. & Bengio, Y. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Front. computational neuroscience* **11**, 24 (2017).
24. Shen, Y. *et al.* Deep learning with coherent nanophotonic circuits. *Nat. photonics* **11**, 441–446 (2017).

25. Lin, X. *et al.* All-optical machine learning using diffractive deep neural networks. *Sci.* **361**, 1004–1008 (2018).
26. Romera, M. *et al.* Vowel recognition with four coupled spin-torque nano-oscillators. *Nat.* **563**, 230–234 (2018).
27. Feldmann, J. *et al.* Parallel convolutional processing using an integrated photonic tensor core. *Nat.* **589**, 52–58 (2021).
28. Stern, M., Hexner, D., Rocks, J. W. & Liu, A. J. Supervised learning in physical networks: From machine learning to learning machines. *Phys. Rev. X* **11**, 021045 (2021).
29. Dillavou, S., Stern, M., Liu, A. J. & Durian, D. J. Demonstration of decentralized physics-driven learning. *Phys. Rev. Appl.* **18**, 014040 (2022).
30. Wright, L. G. *et al.* Deep physical neural networks trained with backpropagation. *Nat.* **601**, 549–555 (2022).
31. Onodera, T. *et al.* Scaling on-chip photonic neural processors using arbitrarily programmable wave propagation. *arXiv:2402.17750* (2024).
32. Hinton, G. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345* (2022).
33. Pai, S. *et al.* Experimentally realized in situ backpropagation for deep learning in photonic neural networks. *Sci.* **380**, 398–404 (2023).
34. Momeni, A., Rahmani, B., Malléjac, M., del Hougne, P. & Fleury, R. Backpropagation-free training of deep physical neural networks. *Sci.* **382**, 1297–1303 (2023).
35. Oguz, I. *et al.* Forward–forward training of an optical neural network. *Opt. Lett.* **48**, 5249–5252 (2023).
36. Le Gallo, M. *et al.* A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *Nat. Electron.* **6**, 680–693 (2023).
37. Lopez-Pastor, V. & Marquardt, F. Self-learning machines based on hamiltonian echo backpropagation. *Phys. Rev. X* **13**, 031020 (2023).
38. Wanjura, C. C. & Marquardt, F. Fully non-linear neuromorphic computing with linear wave scattering. *arXiv:2308.16181* (2023).
39. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin mathematical biophysics* **5**, 115–133 (1943).
40. Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol. review* **65**, 386 (1958).
41. Denker, J. S. Neural network models of learning and adaptation. *Phys. D: Nonlinear Phenom.* **22**, 216–232 (1986).
42. Hebb, D. O. *The organization of behavior: A neuropsychological theory* (Psychology press, 2005).
43. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *nature* **323**, 533–536 (1986).
44. Psaltis, D., Brady, D., Gu, X.-G. & Lin, S. Holography in artificial neural networks. *Nat.* **343**, 325–330 (1990).
45. Hubbard, W. *et al.* Electronic neural networks. In *AIP Conference Proceedings*, vol. 151, 227–234 (American Institute of Physics, 1986).
46. Tanaka, G. *et al.* Recent advances in physical reservoir computing: A review. *Neural Networks* **115**, 100–123 (2019).
47. Jaeger, H. The “echo state” approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Ger. Ger. Natl. Res. Cent. for Inf. Technol. GMD Tech. Rep.* **148**, 13 (2001).
48. Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation* **14**, 2531–2560 (2002).

49. Appeltant, L. *et al.* Information processing using a single dynamical node as complex system. *Nat. communications* **2**, 468 (2011).
50. Marković, D. *et al.* Reservoir computing with the frequency, phase, and amplitude of spin-torque nano-oscillators. *Appl. Phys. Lett.* **114** (2019).
51. Brunner, D., Soriano, M. C., Mirasso, C. R. & Fischer, I. Parallel photonic information processing at gigabyte per second data rates using transient states. *Nat. communications* **4**, 1364 (2013).
52. Lupo, A., Picco, E., Zajnulina, M. & Massar, S. Deep photonic reservoir computer based on frequency multiplexing with fully analog connection between layers. *Opt.* **10**, 1478–1485 (2023).
53. Momeni, A. & Fleury, R. Electromagnetic wave-based extreme deep learning with nonlinear time-floquet entanglement. *Nat. Commun.* **13**, 2651 (2022).
54. Skalli, A. *et al.* Computational metrics and parameters of an injection-locked large area semiconductor laser for neural network computing. *Opt. Mater. Express* **12**, 2793–2804 (2022).
55. Rafayelyan, M., Dong, J., Tan, Y., Krzakala, F. & Gigan, S. Large-scale optical reservoir computing for spatiotemporal chaotic systems prediction. *Phys. Rev. X* **10**, 041037 (2020).
56. Ballarini, D. *et al.* Polaritonic neuromorphic computing outperforms linear classifiers. *Nano Lett.* **20**, 3506–3512 (2020).
57. Nakajima, K. *et al.* A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm. *Front. Comput. Neurosci.* **7**, 91 (2013).
58. Fujii, K. & Nakajima, K. Harnessing disordered-ensemble quantum dynamics for machine learning. *Phys. Rev. Appl.* **8**, 024030 (2017).
59. Marković, D. & Grollier, J. Quantum neuromorphic computing. *Appl. physics letters* **117** (2020).
60. Ghosh, S., Nakajima, K., Krisnanda, T., Fujii, K. & Liew, T. C. Quantum neuromorphic computing with reservoir computing networks. *Adv. Quantum Technol.* **4**, 2100053 (2021).
61. Govia, L., Ribeill, G., Rowlands, G., Krovi, H. & Ohki, T. Quantum reservoir computing with a single nonlinear oscillator. *Phys. Rev. Res.* **3**, 013077 (2021).
62. Mujal, P. *et al.* Opportunities in quantum reservoir computing and extreme learning machines. *Adv. Quantum Technol.* **4**, 2100027 (2021).
63. Yasuda, T. *et al.* Quantum reservoir computing with repeated measurements on superconducting devices. *arXiv:2310.06706* (2023).
64. Hu, F. *et al.* Overcoming the coherence time barrier in quantum machine learning on temporal data. *arXiv:2312.16165* (2023).
65. Senanian, A. *et al.* Microwave signal processing using an analog quantum reservoir computer. *arXiv:2312.16166* (2023).
66. Nakajima, K. Physical reservoir computing—an introductory perspective. *Jpn. J. Appl. Phys.* **59**, 060501 (2020).
67. Teğın, U., Yıldırım, M., Oğuz, İ., Moser, C. & Psaltis, D. Scalable optical learning operator. *Nat. Comput. Sci.* **1**, 542–549 (2021).
68. Oğuz, I. *et al.* Programming nonlinear propagation for efficient optical learning machines. *Adv. Photonics* **6**, 016002, [10.1117/1.AP.6.1.016002](https://doi.org/10.1117/1.AP.6.1.016002) (2024). Publisher: SPIE.
69. Syed, M., Kalinin, K. & Berloff, N. Beyond digital: Harnessing analog hardware for machine learning. In *Machine Learning with New Compute Paradigms* (2023).
70. Hughes, T. W., Williamson, I. A., Minkov, M. & Fan, S. Wave physics as an analog recurrent neural network. *Sci. advances* **5**, eaay6946 (2019).
71. Khoram, E. *et al.* Nanophotonic media for artificial neural inference. *Photonics Res.* **7**, 823–827 (2019).

72. Nakajima, M., Tanaka, K. & Hashimoto, T. Neural schrödinger equation: Physical law as deep neural network. *IEEE transactions on neural networks learning systems* **33**, 2686–2700 (2021).
73. Wu, C. *et al.* Harnessing optoelectronic noises in a photonic generative network. *Sci. advances* **8**, eabm2956 (2022).
74. Coles, P. J. *et al.* Thermodynamic ai and the fluctuation frontier. In *2023 IEEE International Conference on Rebooting Computing (ICRC)*, 1–10 (IEEE, 2023).
75. Bonnet, D. *et al.* Bringing uncertainty quantification to the extreme-edge with memristor-based bayesian neural networks. *Nat. Commun.* **14**, 7530 (2023).
76. Olin-Ammentorp, W., Beckmann, K., Schuman, C. D., Plank, J. S. & Cady, N. C. Stochasticity and robustness in spiking neural networks. *Neurocomputing* **419**, 23–36 (2021).
77. Ma, S.-Y., Wang, T., Laydevant, J., Wright, L. G. & McMahon, P. L. Quantum-noise-limited optical neural networks operating at a few quanta per activation. *arXiv:2307.15712* (2023).
78. Wu, Z., Zhou, M., Khoram, E., Liu, B. & Yu, Z. Neuromorphic metasurface. *Photonics Res.* **8**, 46–50 (2020).
79. Furuhata, G., Niiyama, T. & Sunada, S. Physical deep learning based on optimal control of dynamical systems. *Phys. Rev. Appl.* **15**, 034092 (2021).
80. Burr, G. W. *et al.* Neuromorphic computing using non-volatile memory. *Adv. Physics: X* **2**, 034092 (2017).
81. Chen, Z. *et al.* Deep learning with coherent vcsel neural networks. *Nat. Photonics* **17**, 723–730 (2023).
82. Chang, J., Sitzmann, V., Dun, X., Heidrich, W. & Wetzstein, G. Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification. *Sci. Reports* **8**, 12324 (2018).
83. Joshi, V. *et al.* Accurate deep neural network inference using computational phase-change memory. *Nat. Commun.* **11**, 2473 (2020).
84. Rasch, M. J. *et al.* Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators. *Nat. Commun.* **14**, 5282 (2023).
85. Gallo, M. L. *et al.* A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *Nat. Electron.* **6**, 680–693 (2023).
86. Rahmani, B. *et al.* Actor neural networks for the robust control of partially measured nonlinear systems showcased for image propagation through diffuse media. *Nat. Mach. Intell.* **2**, 403–410 (2020).
87. Matsushima, K. & Shimobaba, T. Band-limited angular spectrum method for numerical simulation of free-space propagation in far and near fields. *Opt. Express* **17**, 19662–19673 (2009).
88. Shi, L., Li, B., Kim, C., Kellnhofer, P. & Matusik, W. Towards real-time photorealistic 3d holography with deep neural networks. *Nat.* **591**, 234–239 (2021).
89. Adhikari, S. P., Yang, C., Kim, H. & Chua, L. O. Memristor bridge synapse-based neural network and its learning. *IEEE Transactions on Neural Networks Learn. Syst.* **23**, 1426–1435 (2012).
90. Cramer, B. *et al.* Surrogate gradients for analog neuromorphic computing. *Proc. Natl. Acad. Sci.* **119** (2022).
91. Spall, J., Guo, X. & Lvovsky, A. I. Hybrid training of optical neural networks. *Opt.* **9**, 803 (2022).
92. Lillicrap, T. P., Cownden, D., Tweed, D. B. & Akerman, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nat. Commun.* **7** (2016).
93. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. & Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.* **18**, 1–30 (2018).
94. Launay, J., Poli, I., Boniface, F. & Krzakala, F. Direct feedback alignment scales to modern deep learning tasks and architectures. *Adv. neural information processing systems* **33**, 9346–9360 (2020).
95. Brunton, S. L. & Kutz, J. N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press, 2022).

96. Karniadakis, G. E. *et al.* Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021).
97. Spall, J., Guo, X. & Lvovsky, A. I. Hybrid training of optical neural networks. *Opt.* **9**, 803–811 (2022).
98. Ashtiani, F., Geers, A. J. & Aflatouni, F. An on-chip photonic deep neural network for image classification. *Nat.* **606**, 501–506 (2022).
99. Liu, C. *et al.* A programmable diffractive deep neural network based on a digital-coding metasurface array. *Nat. Electron.* **5**, 113–122 (2022).
100. Zhou, T. *et al.* In situ optical backpropagation training of diffractive optical neural networks. *Photonics Res.* **8**, 940–953 (2020).
101. Hughes, T. W., Minkov, M., Shi, Y. & Fan, S. Training of photonic neural networks through in situ backpropagation and gradient measurement. *Opt.* **5**, 864–871 (2018).
102. Mengü, D. *et al.* Misalignment resilient diffractive optical networks. *Nanophotonics* **9**, 4207–4219 (2020).
103. Laydevant, J., Lott, A., Venturelli, D. & McMahon, P. L. The benefits of self-supervised learning for training physical neural networks. In *Machine Learning with New Compute Paradigms* (2023).
104. Lillicrap, T. P., Cownden, D., Tweed, D. B. & Akerman, C. J. Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247* (2014).
105. Nakajima, M. *et al.* Physical deep learning with biologically inspired training method: gradient-free approach for physical hardware. *Nat. communications* **13**, 7847 (2022).
106. Wang, Y. *et al.* Asymmetrical estimator for training grey-box deep photonic neural networks. *arXiv preprint arXiv:2405.18458* (2024).
107. Han, D. & Yoo, H.-j. Efficient convolutional neural network training with direct feedback alignment. *arXiv preprint arXiv:1901.01986* (2019).
108. Refinetti, M., d’Ascoli, S., Ohana, R. & Goldt, S. Align, then memorise: the dynamics of learning with feedback alignment. In *International Conference on Machine Learning*, 8925–8935 (PMLR, 2021).
109. Han, D., Park, G., Ryu, J. & Yoo, H.-j. Extension of direct feedback alignment to convolutional and recurrent neural network for bio-plausible deep learning. *arXiv preprint arXiv:2006.12830* (2020).
110. Hinton, G. E., Dayan, P., Frey, B. J. & Neal, R. M. The "wake-sleep" algorithm for unsupervised neural networks. *Sci.* **268**, 1158–1161 (1995).
111. Salakhutdinov, R., Mnih, A. & Hinton, G. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, 791–798 (2007).
112. Vincent, P. *et al.* Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. machine learning research* **11** (2010).
113. Federici, M., Dutta, A., Forré, P., Kushman, N. & Akata, Z. Learning robust representations via multi-view information bottleneck. *arXiv preprint arXiv:2002.07017* (2020).
114. Löwe, S., O’Connor, P. & Veeling, B. Putting an end to end-to-end: Gradient-isolated learning of representations. *Adv. neural information processing systems* **32** (2019).
115. Nøklund, A. & Eidnes, L. H. Training neural networks with local error signals. In *International conference on machine learning*, 4839–4850 (PMLR, 2019).
116. Ren, M., Kornblith, S., Liao, R. & Hinton, G. Scaling forward gradient with local losses. In *The Eleventh International Conference on Learning Representations* (2023).
117. Siddiqui, S. A., Krueger, D., LeCun, Y. & Deny, S. Blockwise self-supervised learning at scale. *arXiv preprint arXiv:2302.01647* (2023).
118. Xiong, Y., Ren, M. & Urtasun, R. Loco: Local contrastive representation learning. *Adv. neural information processing systems* **33**, 11142–11153 (2020).
119. Gomez, A. N. *et al.* Interlocking backpropagation: Improving depthwise model-parallelism. *J. Mach. Learn. Res.* **23**, 1–28 (2022).

120. Balestriero, R. *et al.* A cookbook of self-supervised learning. *arXiv:2304.12210* (2023).
121. Momeni, A., Rahmani, B., Malléjac, M., del Hougne, P. & Fleury, R. Phyff: Physical forward forward algorithm for in-hardware training and inference. In *Machine Learning with New Compute Paradigms* (2023).
122. Zhao, G., Shu, X. & Zhou, R. High-performance real-world optical computing trained by in situ model-free optimization. *arXiv preprint arXiv:2307.11957* (2023).
123. Spall, J. C. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control* **37**, 332–341 (1992).
124. Wang, C., Xiong, L., Sun, J. & Yao, W. Memristor-based neural networks with weight simultaneous perturbation training. *Nonlinear Dyn.* **95**, 2893–2906 (2019).
125. Bandyopadhyay, S. *et al.* Single chip photonic deep neural network with accelerated training. *arXiv preprint arXiv:2208.01623* (2022).
126. Gu, J. *et al.* Flops: Efficient on-chip learning for optical neural networks through stochastic zeroth-order optimization. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 1–6 (IEEE, 2020).
127. Gu, J. *et al.* L2ight: Enabling on-chip learning for optical neural networks via efficient in-situ subspace optimization. *Adv. Neural Inf. Process. Syst.* **34**, 8649–8661 (2021).
128. McCaughan, A. N. *et al.* Multiplexed gradient descent: Fast online training of modern datasets on hardware neural networks without backpropagation. *APL Mach. Learn.* **1** (2023).
129. Hansen, N. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772* (2016).
130. Bueno, J. *et al.* Reinforcement learning in a large-scale photonic recurrent neural network. *Opt.* **5**, 756–760 (2018).
131. Kanno, K., Naruse, M. & Uchida, A. Adaptive model selection in photonic reservoir computing by reinforcement learning. *Sci. reports* **10**, 10062 (2020).
132. Yi, S.-i., Kendall, J. D., Williams, R. S. & Kumar, S. Activity-difference training of deep neural networks using memristor crossbars. *Nat. Electron.* **6**, 45–51 (2023).
133. Hermans, M., Burm, M., Van Vaerenbergh, T., Dambre, J. & Bienstman, P. Trainable hardware for dynamical computing using error backpropagation through physical media. *Nat. Commun.* **6**, 6729 (2015).
134. Miller, D. A. B. Setting up meshes of interferometers - reversed local light interference method. *Opt. Express* **25**, 29233–29248, [10.1364/OE.25.029233](https://doi.org/10.1364/OE.25.029233) (2017).
135. Morichetti, F. *et al.* Non-invasive on-chip light observation by contactless waveguide conductivity monitoring. *IEEE J. Sel. Top. Quantum Electron.* **20**, 292–301, [10.1109/JSTQE.2014.2300046](https://doi.org/10.1109/JSTQE.2014.2300046) (2014).
136. Guo, X., Barrett, T. D., Wang, Z. M. & Lvovsky, A. Backpropagation through nonlinear units for the all-optical training of neural networks. *Photonics Res.* **9**, B71–B80 (2021).
137. Spall, J., Guo, X. & Lvovsky, A. Training neural networks with end-to-end optical backpropagation. *arXiv preprint arXiv:2308.05226* (2023).
138. Yildirim, M., Dinc, N. U., Oguz, I., Psaltis, D. & Moser, C. Nonlinear processing with linear optics. *arXiv:2307.08533* (2023).
139. Xia, F. *et al.* Deep learning with passive optical nonlinear mapping. *arXiv:2307.08558* (2023).
140. Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. A learning algorithm for boltzmann machines. *Cogn. science* **9**, 147–169 (1985).
141. Movellan, J. R. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist Models*, 10–17 (Elsevier, 1991).
142. Laborieux, A. & Zenke, F. Holomorphic equilibrium propagation computes exact gradients through finite size oscillations. *Adv. Neural Inf. Process. Syst.* **35**, 12950–12963 (2022).



143. Scellier, B., Ernoult, M., Kendall, J. & Kumar, S. Energy-based learning algorithms for analog computing: a comparative study. *Adv. Neural Inf. Process. Syst.* **36** (2023).
144. Kendall, J., Pantone, R., Manickavasagam, K., Bengio, Y. & Scellier, B. Training end-to-end analog neural networks with equilibrium propagation. *arXiv preprint arXiv:2006.01981* (2020).
145. Laydevant, J., Marković, D. & Grollier, J. Training an ising machine with equilibrium propagation. *Nat. Commun.* **15**, [10.1038/s41467-024-46879-4](https://doi.org/10.1038/s41467-024-46879-4) (2024).
146. Wang, Q., Wanjura, C. C. & Marquardt, F. Training coupled phase oscillators as a neuromorphic platform using equilibrium propagation. *arXiv preprint arXiv:2402.08579* (2024).
147. Altman, L. E., Stern, M., Liu, A. J. & Durian, D. J. Experimental demonstration of coupled learning in elastic networks. *arXiv preprint arXiv:2311.00170* (2023).
148. Dillavou, S. *et al.* Machine learning without a processor: Emergent learning in a nonlinear electronic metamaterial. *arXiv preprint arXiv:2311.00537* (2023).
149. Stern, M., Dillavou, S., Jayaraman, D., Durian, D. J. & Liu, A. J. Training self-learning circuits for power-efficient solutions. *APL Mach. Learn.* **2** (2024).
150. Anisetti, V. R., Kandala, A., Scellier, B. & Schwarz, J. Frequency propagation: Multimechanism learning in nonlinear physical networks. *Neural Comput.* 1–25 (2024).
151. Falk, M., Strupp, A., Scellier, B. & Murugan, A. Contrastive learning through non-equilibrium memory. *arXiv preprint arXiv:2312.17723* (2023).
152. Martin, E. *et al.* Eqspike: spike-driven equilibrium propagation for neuromorphic implementations. *IScience* **24** (2021).
153. Scellier, B., Mishra, S., Bengio, Y. & Ollivier, Y. Agnostic physics-driven deep learning. *arXiv preprint arXiv:2205.15021* (2022).
154. Van de Ven, G. M., Tuytelaars, T. & Tolias, A. S. Three types of incremental learning. *Nat. Mach. Intell.* **4**, 1185–1197 (2022).
155. Zhang, C. *et al.* Few-shot incremental learning with continually evolved classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 12455–12464 (2021).
156. Hersche, M. *et al.* Constrained few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9057–9067 (2022).
157. Karunaratne, G. *et al.* In-memory realization of in-situ few-shot continual learning with a dynamically evolving explicit memory. In *ESSCIRC 2022-IEEE 48th European Solid State Circuits Conference (ESSCIRC)*, 105–108 (IEEE, 2022).
158. Stroeve, N. & Berloff, N. G. Analog photonics computing for information processing, inference, and optimization. *Adv. Quantum Technol.* **6**, 2300055, <https://doi.org/10.1002/qute.202300055> (2023).
159. Mohseni, N., McMahon, P. L. & Byrnes, T. Ising machines as hardware solvers of combinatorial optimization problems. *Nat. Rev. Phys.* **4**, 363–379 (2022).
160. Stroeve, N. & Berloff, N. G. Neural network architectures based on the classical xy model. *Phys. Rev. B* **104**, 205435 (2021).
161. Vaswani, A. *et al.* Attention is all you need. *Adv. neural information processing systems* **30** (2017).
162. Anderson, M. G., Ma, S.-Y., Wang, T., Wright, L. G. & McMahon, P. L. Optical Transformers. *Transactions on Mach. Learn. Res. arXiv:2302.10360* (2024).
163. Radford, A. *et al.* Language models are unsupervised multitask learners. *OpenAI blog* **1**, 9 (2019).
164. Mann, B. *et al.* Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).
165. Touvron, H. *et al.* Llama: Open and efficient foundation language models. corr, abs/2302.13971, 2023. doi: 10.48550. *arXiv preprint arXiv.2302.13971* (2023).

166. Chowdhery, A. *et al.* Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.* **24**, 1–113 (2023).
167. Achiam, J. *et al.* Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
168. Team, G. Gemini: A family of highly capable multimodal models (2024). [2312.11805](#).
169. Radford, A. *et al.* Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763 (PMLR, 2021).
170. Liu, H., Li, C., Wu, Q. & Lee, Y. J. Visual instruction tuning. *Adv. neural information processing systems* **36** (2024).
171. Zeni, C. *et al.* Mattergen: a generative model for inorganic materials design. *arXiv preprint arXiv:2312.03687* (2023).
172. Jumper, J. *et al.* Highly accurate protein structure prediction with alphafold. *Nat.* **596**, 583–589 (2021).
173. Andrychowicz, M. *et al.* Deep learning for day forecasts from sparse observations. arxiv 2023. *arXiv preprint arXiv:2306.06079* .
174. Xiong, Y. *et al.* Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 14138–14148 (2021).
175. Child, R., Gray, S., Radford, A. & Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509* (2019).
176. Katharopoulos, A., Vyas, A., Pappas, N. & Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, 5156–5165 (PMLR, 2020).
177. Zhang, M., Bhatia, K., Kumbong, H. & Ré, C. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *arXiv preprint arXiv:2402.04347* (2024).
178. Peng, H. *et al.* Random feature attention. *arXiv preprint arXiv:2103.02143* (2021).
179. Gu, A. & Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752* (2023).
180. Frantar, E., Ashkboos, S., Hoefler, T. & Alistarh, D. Optq: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations* (2022).
181. Lin, J. *et al.* Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978* (2023).
182. Ma, S. *et al.* The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764* (2024).
183. Wang, H. *et al.* Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453* (2023).
184. Christiano, P. F. *et al.* Deep reinforcement learning from human preferences. *Adv. neural information processing systems* **30** (2017).
185. Rafailov, R. *et al.* Direct preference optimization: Your language model is secretly a reward model. *Adv. Neural Inf. Process. Syst.* **36** (2024).
186. Hu, E. J. *et al.* Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).
187. Houlsby, N. *et al.* Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, 2790–2799 (PMLR, 2019).
188. Lin, Z., Madotto, A. & Fung, P. Exploring versatile generative language model via parameter-efficient transfer learning. *arXiv preprint arXiv:2004.03829* (2020).
189. Dao, T., Fu, D. Y., Ermon, S., Rudra, A. & Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness (2022). [2205.14135](#).

190. Juravsky, J. *et al.* Hydragen: High-throughput llm inference with shared prefixes (2024). [2402.05099](#).
191. Kwon, W. *et al.* Efficient memory management for large language model serving with pagedattention (2023). [2309.06180](#).
192. Hamerly, R., Bernstein, L., Sludds, A., Soljačić, M. & Englund, D. Large-scale optical neural networks based on photoelectric multiplication. *Phys. Rev. X* **9**, 021032 (2019).
193. Nahmias, M. A. *et al.* Photonic multiply-accumulate operations for neural networks. *IEEE J. Sel. Top. Quantum Electron.* **26**, 1–18 (2019).
194. Wang, T. *et al.* An optical neural network using less than 1 photon per multiplication. *Nat. Commun.* **13**, 123 (2022).
195. Tait, A. N. Quantifying power in silicon photonic neural networks. *Phys. Rev. Appl.* **17**, 054029 (2022).
196. Laydevant, J., Wright, L. G., Wang, T. & McMahon, P. L. The hardware is the software. *Neuron* **112**, 180–183 (2024).
197. Hooker, S. The hardware lottery. *Commun. ACM* **64**, 58–65 (2021).
198. Cerezo, M., Verdon, G., Huang, H.-Y., Cincio, L. & Coles, P. J. Challenges and opportunities in quantum machine learning. *Nat. Comput. Sci.* **2**, 567–576 (2022).
199. Cai, F. *et al.* A fully integrated reprogrammable memristor–cmos system for efficient multiply–accumulate operations. *Nat. electronics* **2**, 290–299 (2019).
200. Grollier, J. *et al.* Neuromorphic spintronics. *Nat. electronics* **3**, 360–370 (2020).
201. Zhou, F. & Chai, Y. Near-sensor and in-sensor computing. *Nat. Electron.* **3**, 664–671 (2020).
202. Saigre-Tardif, C., Faqiri, R., Zhao, H., Li, L. & del Hougne, P. Intelligent meta-imagers: from compressed to learned sensing. *Appl. Phys. Rev.* **9** (2022).
203. Zhou, M.-G. *et al.* Quantum neural network for quantum neural computing. *Res.* **6**, 0134 (2023).
204. Pierangeli, D., Marcucci, G., Brunner, D. & Conti, C. Noise-enhanced spatial-photonic ising machine. *Nanophotonics* **9**, 4109–4116 (2020).
205. Marsh, B. P. *et al.* Enhancing associative memory recall and storage capacity using confocal cavity qed. *Phys. Rev. X* **11**, 021048 (2021).
206. Momeni, A., Guo, X., Lissek, H. & Fleury, R. Physics-inspired neuroacoustic computing based on tunable nonlinear multiple-scattering. *arXiv preprint arXiv:2304.08380* (2023).
207. Kashif, M. & Shafique, M. Hqnet: Harnessing quantum noise for effective training of quantum neural networks in nisq era. *arXiv preprint arXiv:2402.08475* (2024).
208. Tian, J. *et al.* Recent advances for quantum neural networks in generative learning. *IEEE Transactions on Pattern Analysis Mach. Intell.* (2023).
209. Cerezo, M. *et al.* Variational quantum algorithms. *Nat. Rev. Phys.* **3**, 625–644 (2021).
210. Niazi, S. *et al.* Training deep boltzmann networks with sparse ising machines. *arXiv preprint arXiv:2303.10728* (2023).
211. McMahon, P. L. The physics of optical computing. *Nat. Rev. Phys.* **5**, 717–734 (2023).
212. Keeling, J. & Berloff, N. G. Exciton–polariton condensation. *Contemp. Phys.* **52**, 131–151 (2011).
213. Carusotto, I. & Ciuti, C. Quantum fluids of light. *Rev. Mod. Phys.* **85**, 299 (2013).
214. Berloff, N. G. *et al.* Realizing the classical xy hamiltonian in polariton simulators. *Nat. materials* **16**, 1120–1126 (2017).
215. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N. & Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, 2256–2265 (PMLR, 2015).

216. Johnston, A. & Berloff, N. G. Macroscopic noise amplification by asymmetric dyads in non-hermitian optical systems for generative diffusion models. *Phys. Rev. Lett.* **132**, 096901 (2024).
217. Horstmeyer, R., Chen, R. Y., Kappes, B. & Judkewitz, B. Convolutional neural networks that teach microscopes how to image. *arXiv preprint arXiv:1709.07223* (2017).
218. Sitzmann, V. *et al.* End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging. *ACM Trans. Graph. (TOG)* **37**, 1–13 (2018).
219. del Hougne, P., F. Imani, M., Diebold, A. V., Horstmeyer, R. & Smith, D. R. Learned integrated sensing pipeline: reconfigurable metasurface transceivers as trainable physical layer in an artificial neural network. *Adv. Sci.* **7**, 1901913 (2020).
220. Zhou, T. *et al.* Large-scale neuromorphic optoelectronic computing with a reconfigurable diffractive processing unit. *Nat. Photonics* **15**, 367–373 (2021).
221. Wang, T. *et al.* Image sensing with multilayer nonlinear optical neural networks. *Nat. Photonics* **17**, 408–415 (2023).
222. You, X. *et al.* Towards 6g wireless communication networks: Vision, enabling technologies, and new paradigm shifts. *Sci. China Inf. Sci.* **64**, 1–74 (2021).
223. Rajabalipanah, H., Abdolali, A., Shabanpour, J., Momeni, A. & Cheldavi, A. Addition theorem revisiting for phase/amplitude-encoded metasurfaces: Asymmetric spatial power dividers. *arXiv preprint arXiv:1901.04063* (2019).
224. Tahmasebi, O., Abdolali, A., Rajabalipanah, H., Momeni, A. & Fleury, R. Parallel temporal signal processing enabled by polarization-multiplexed programmable thz metasurfaces. *Opt. Express* **30**, 45221–45232 (2022).
225. Rajabalipanah, H., Momeni, A., Rahmanzadeh, M., Abdolali, A. & Fleury, R. Parallel wave-based analog computing using metagratings. *Nanophotonics* **11**, 1561–1571 (2022).
226. Bai, B. *et al.* To image, or not to image: class-specific diffractive cameras with all-optical erasure of undesired objects. *eLight* **2**, 14 (2022).
227. Li, J. *et al.* Rapid sensing of hidden objects and defects using a single-pixel diffractive terahertz sensor. *Nat. Commun.* **14**, 6791 (2023).
228. Qian, C. & del Hougne, P. Noise-adaptive intelligent programmable meta-imager. *Intell. Comput.* (2022).
229. Ríos, C. *et al.* In-memory computing on a photonic platform. *Sci. Adv.* **5**, eaau5759 (2019).
230. Sludds, A. *et al.* Delocalized photonic deep learning on the internet’s edge. *Sci.* **378**, 270–276 (2022).
231. Rahman, M. S. S. *et al.* Learning diffractive optical communication around arbitrary opaque occlusions. *Nat. Commun.* **14**, 6830 (2023).
232. López, A., Pérez, D., DasMahapatra, P. & Capmany, J. Auto-routing algorithm for field-programmable photonic gate arrays. *Opt. Express* **28**, 737–752 (2020).
233. Sol, J., Alhulaymi, A., Stone, A. D. & del Hougne, P. Reflectionless programmable signal routers. *Sci. Adv.* **9**, eadf0323 (2023).
234. Mengü, D., Zhao, Y., Tabassum, A., Jarrahi, M. & Ozcan, A. Diffractive interconnects: all-optical permutation operation using diffractive networks. *Nanophotonics* **12**, 905–923 (2023).
235. Dinc, N. U., Yildirim, M., Moser, C. & Psaltis, D. Multicasting optical reconfigurable switch. *arXiv preprint arXiv:2401.14173* (2024).
236. SeyedinNavadeh, S. *et al.* Determining the optimal communication channels of arbitrary optical systems using integrated photonic processors. *Nat. Photonics* **18**, 149–155 (2024).
237. Miller, D. A. Self-configuring universal linear optical component. *Photonics Res.* **1**, 1–15 (2013).
238. Roques-Carmes, C., Fan, S. & Miller, D. Measuring, processing, and generating partially coherent light with self-configuring optics. *arXiv preprint arXiv:2402.00704* (2024).

- 239.** Momeni, A., Guo, X., Lissek, H. & Fleury, R. Learning to compute with sound in nonlinear disordered cavities. In *2022 Sixteenth International Congress on Artificial Materials for Novel Wave Phenomena (Metamaterials)*, X-314 (IEEE, 2022).