Training Physical Networks Like Neural Networks: Deep Physical Neural Networks

Logan G. Wright[*1,2], **Tatsuhiro Onodera***[1,2], **Martin M. Stein**[1],
Tianyu Wang[1], **Darren Schachter**[3], **Zoey Hu**[1], and **Peter L. McMahon**[1] [†]
1. School of Applied and Engineering Physics, Cornell University, Ithaca, NY, USA
2. Physics & Informatics Laboratories, NTT Research, Inc., Sunnyvale, CA, USA
3. School of Electrical and Computer Engineering, Cornell University, Ithaca, NY, USA

Abstract

Deep neural networks (DNNs) are increasingly used to predict physical processes. Here, we invert this relationship, and show that physical processes with adjustable physical parameters (e.g., geometry, voltages) can be trained to emulate DNNs, i.e., to perform machine learning inference tasks. We call these trainable processes *physical neural networks* (PNNs). We train experimental PNNs based on broadband optical pulses propagating in a nonlinear crystal, a nonlinear electronic oscillator, and an oscillating metal plate. As an extension of these laboratory proof-of-concepts, we train (in simulation) a network of coupled oscillators to perform Fashion MNIST classification. Since one cannot apply autodifferentiation directly to physical processes, we introduce a technique that uses a simulation model to efficiently estimate the gradients of the physical system, allowing us to use backpropagation to train PNNs. Using this technique, we train each system's physical transformations (which do not necessarily resemble typical DNN layers) directly to perform inference calculations. Our work may help inspire novel neural network architectures, including ones that can be efficiently realized with particular physical processes, and presents a route to training complex physical systems to take on desired physical functionalities, such as computational sensing. This article is intended as a summary of the previously published work (1) for the NeurIPS 2022 Machine Learning and the Physical Sciences workshop.

1 Introduction

Arguably the foundational insight in physics is that real physical processes can be described by sequences of mathematical transformations. For example, if we want to know the drag experienced by a plane moving at high speed in air, we can use the Navier-Stokes equations to simulate the fluid around the airplane. But we can also reverse this relationship: We can use physical processes to compute the mathematical transformations that describe them. For example, solving the Navier-Stokes equations is hard and expensive with digital computers. Wind tunnels - actual real-world physical systems - have thus routinely been used to "simulate" these equations, informing engineers. More generally, physical processes often compute much more efficiently, or faster, than digital electronic simulations. This is well-known, and is the primary motivation for many kinds of physical analog computers, including analog electronics and emerging optical and quantum computers.

A physics simulation implements a set of sequential mathematical transformations, which are functions of the initial conditions, $\vec{x}$, and other parameters of the physical process, $\vec{\theta}_\mathrm{p}$, such as the shape

[*]These authors contributed equally to this work
[†]Correspondence to: lgw32@cornell.edu; to232@cornell.edu; pmcmahon@cornell.edu

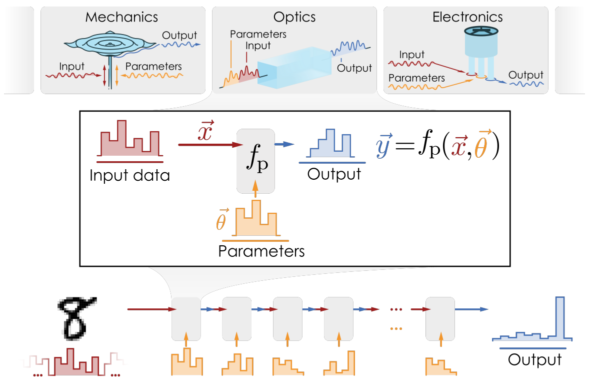

Figure 1: The main idea of physical neural networks. Physical processes implement mathematical transformations on their initial conditions, which can be controlled by physical parameters. By using the outputs of a physical process as the inputs to subsequent physical processes, we realize hierarchical, trainable physical data transformations loosely analogous to deep neural networks. Figure adapted from Ref. (1).

of the airplane (Figure 1). The output of a simulation is a vector of measurable quantities about the physical process, $\vec{y} = f_{\text{sim}}(\vec{x}, \vec{\theta}_{\text{sim}})$, where $\vec{\theta}_{\text{sim}}$ are the simulation's parameters. $\vec{y}$ may be, e.g., the drag forces, or the distribution of velocity or density, typically as a function of space and/or time. This function, $f_{\text{sim}}(\vec{x}, \vec{\theta}_{\text{sim}})$ is ideally a good approximation of the physical process (assuming $\vec{\theta}_{\text{sim}} \approx \vec{\theta}_{\text{p}}$). That physical process is, in turn, also a good approximation of $f_{\text{sim}}(\vec{x}, \vec{\theta}_{\text{sim}})$. Here, we think of a physical process as a physically implemented sequence of mathematical transformations, $f_{\text{p}}(\vec{x}, \vec{\theta}_{\text{p}})$, which can be controlled by adjusting physical parameters, $\vec{\theta}_{\text{p}}$.

Many researchers have realized that we can go beyond the limitations of traditional mathematical models for physical processes by augmenting them with data-driven techniques such as deep learning (e.g., (2; 3; 4)). DNNs appear to be well-suited to describing real physical processes, since they share qualities such as hierarchical structure, nonlinearity, locality, and sparsity (5). DNNs thus provide an effective adjustable nonlinear transformation, $f_{\text{DNN}}(\vec{x}, \vec{\theta})$ that can be trained to approximate physics simulations, $f_{\text{sim}}(\vec{x}, \vec{\theta}_{\text{sim}})$ and/or real physical processes, $f_{\text{p}}(\vec{x}, \vec{\theta}_{\text{p}})$.

Here, instead of training DNNs to approximate physical processes, we train physical processes to perform machine learning inference calculations that would ordinarily be realized with DNNs (1). Specifically, we use physical processes with adjustable parameters as adjustable nonlinear functions, $f_{\text{p}}(\vec{x}, \vec{\theta}_{\text{p}})$, and combine multiple processes to implement something loosely analogous to a DNN: a physical neural network (PNN). In a multi-layer PNN, the outputs of a given physical process, $\vec{y} = f_{\text{p}}(\vec{x}, \vec{\theta}_{\text{p}})$ are used as the input to a subsequent one (Figure 1). Each physical "layer" has its own parameters, and does not necessarily need to implement the same mathematical operations used in conventional DNN layers.

Using PNNs requires being able to train them efficiently, i.e. to learn how to set their *physical* parameters so that they can approximate desired transformations. Backpropagation, the de facto solution for conventional DNNs, is based on autodifferentiation, which cannot be applied to actual physical processes. We can, however, apply autodifferentiation to a simulation of the physical process (6; 7; 1). Simulations (even data-driven ones) do not perfectly predict real physical processes, so this strategy can result in inaccurate training. In our work, we introduced a rather simple algorithm to overcome this, called physics-aware training (PAT). PAT generalizes a common trick used in algorithms like quantization-aware training (QAT) (8). PAT uses the physical process on the forward pass, and a simulation to estimate gradients on the backward pass.

2 Related work

Deep learning accelerators are special-purpose hardware aimed at performing deep learning (9). They are mainly digital electronics, but there is a growing minority devoted to analog electronics or photonics (10). The majority of analog devices are still inference-only, and are designed to

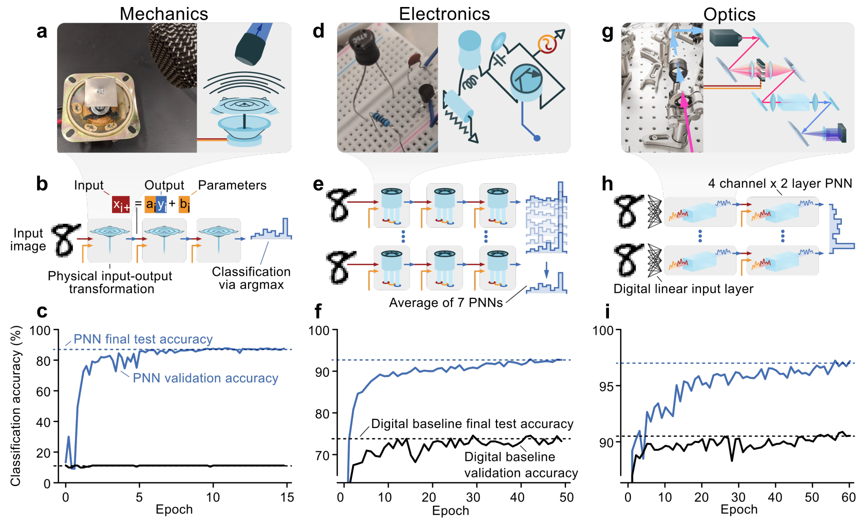

Figure 2: Experimental physical neural networks trained to perform MNIST handwritten-digit image classification. For details of each physical system and PNN architecture, see main text. Figure adapted from Ref. (1).

implement a direct mathematical analogy to a matrix-vector multiplication. In PNNs, no such direct mathematical analogy is required or assumed, though a looser one may be helpful (1).

Neural ordinary differential equations treat DNNs as (mathematical) dynamical systems, reframing backpropagation as adjoint optimization (11; 12). In physical reservoir computing (PRC) (13; 14), physical transformations of data are used as an untrained feature map. A linear output layer, typically realized on a digital computer, is used to combine these features to approximate functions. *In materio* computers use complex, reconfigurable materials as a computational substrate (15; 16). *In materio* computing is similar to PNNs and PRC, in that no specific mathematical analogy is required or assumed. It is further similar to PNNs because the physical transformations of the material are optimized, historically by use of a black-box, gradient free algorithm (but more recently by gradient descent (17)). PNNs are trained with backpropagation, and assume a more general network of more general physical processes (e.g., all our PNNs consist of multiple interconnected physical layers, and none of the physical transformations we consider are based on reconfigurable materials). Beyond *in materio* computing, there are recent proposals, e.g. in photonics (7; 6; 18; 19) and quantum computing (20), for training specific physical systems with adjustable parameters for machine learning inference, without requiring strict mathematical analogies to standard DNNs.

Finally, there are proposals for physics-based learning (21; 22; 23). These methods use physical processes for training, rather than just inference. They may eventually accelerate the training of PNNs, which so far has partially taken place within conventional computers (1).

3 Results

Figure 2 shows a summary of several experimental PNNs we have built and trained to perform MNIST image classification. We provide a summary here; for complete experimental details, and architectures, see Ref. (1), for data and code (24; 25). Training used PyTorch (26) and Optuna (27). For simplicity, in these examples, we encode parameters by applying a trained digital mask to a one-dimensional vector sent to the physical system (e.g., a time series of voltages). In other words, each time a physical transformation is used, it is in the form $f_{\mathrm{p}}(\vec{x} \odot \vec{a} + \vec{b})$, where $\vec{a}$ and $\vec{b}$ are trainable parameters. This choice simplified some parts of the experimental design and allowed us to take a uniform approach to all three physically distinct systems. It means however that each network should be considered a digital-physical hybrid, even though the majority of computations occur through the physical transformations of each physical system. To show this, in Figure 2 we also show a digital reference, in which the physical device is replaced by an identity operation, i.e., $f_{\mathrm{p}}(\vec{x}) \rightarrow \vec{x}$.

In addition to involving a different physical system, all three PNNs presented in Figure 2 have a different network architecture. Just as modern DNNs can involve different architectures, such as different numbers or types of layers, interconnected, trained or initialized in different ways, in PNNs

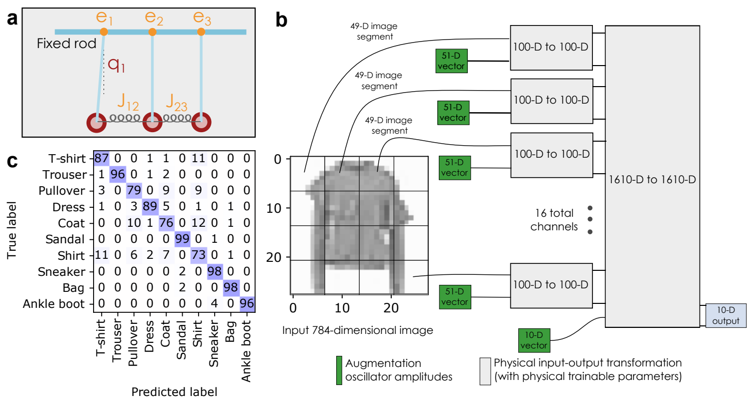

Figure 3: A simulated physical neural network, based entirely on coupled oscillators, trained to perform Fashion MNIST (28). The oscillators can be imagined as pendula, where the spring couplings between, and drives to each pendulum, are trainable parameters. The diagram in (a) is simplified; We consider larger networks with all-to-all coupling. In each physical transformation (grey box in part b), a network of coupled oscillators is simulated. The final read-out is from the final amplitude of 10 oscillators. (c) Confusion matrix for test set, 89.9% test accuracy. Figure adapted from Ref. (1).

the trained physical transformations can be sequenced/interconnected/trained/initialized in essentially limitless number of different ways. We have so far explored a variety of PNN architectures, including multiple layers, multiple channels, skip connections, and hybrid digital-physical designs (1). We have found that, as with conventional DNNs, choice of architecture has a significant influence on how a PNN performs on a given task. The difference in performance between the three PNNs presented in Figure 2 is thus due in part to the differences in the underlying physical transformations, as well as the architectures. Overall, the results presented here should be viewed merely as initial explorations: Improving the design of future PNNs will require more effort to understand both which kinds of physical transformations are most useful (and most scalable) as well as which kinds of PNN architectures are most effective.

The first PNN (Fig. 2a-c) is based on the oscillations of a metal plate, driven by a speaker and recorded by a microphone. The architecture includes three physical layers, i.e., the metal plate is used three times, with the output of the previous layer provided as input, and with new trained parameters. Since the oscillations of the plate are mostly linear, the performance of this PNN is somewhat limited, reaching only 87% test accuracy. On the first layer, the MNIST image is read as an unrolled 1D vector or time-series of forces. The final output of the PNN is a 10-D vector taken from the end of the microphone's recording.

The second PNN (Fig. 2d-f) is based on a bulk nonlinear electronic oscillator. Input and output are voltage time series, similar to the metal-plate-PNN. The architecture is based on an ensemble of seven, three-layer networks. Each of the seven, 3-layer networks has different parameters, and the final output of the PNN is the average of all seven multilayer networks. We find that the ensemble architecture allows for stable classification performance superior to a linear digital classifier, even in the presence of large analog noise.

A third PNN (Fig. 2g-i) is based on the propagation of a broadband (femtosecond duration) optical pulse in a nonlinear optical medium. Data and parameters are encoded via the amplitudes of different frequency components of this pulse, and the output of the transformation is the optical spectrum measured after the nonlinear crystal. For this PNN, we include linear digital input layers, prior to a 4-channel, 2-layer physical network. We included these digital layers to show that digital and physical subparts of a network can be co-trained naturally using backpropagation. Such an approach could be useful to automatically learn how to split up a complicated task so it may be maximally accelerated by (i.e., maximally offloaded to) a limited analog physical co-processor.

Digital operations are not necessary for a PNN to perform machine learning inference. To show this, we performed a vowel classification task using the nonlinear optical PNN, but where the trained physical parameters are not a digital mask on the input vector, but instead amplitudes of a second portion of the optical spectrum (1).

To further show both that PNNs can perform computations entirely with trained physical transformations, and that they can, when scaled up, achieve competitive performance on more challenging tasks, we performed the Fashion MNIST (28) task with a simulated PNN based entirely on networks of coupled oscillators (Figure 3a). Using this physical system, we constructed a physical network loosely inspired by the MLP-Mixer network (29) (Figure 3b). This 2-layer PNN classifies Fashion MNIST images with 89.9 % test accuracy, roughly equivalent to multilayer CNNs that use about 10^6 to 10^7 MACs per inference (30). The trained physical parameters are the coupling rates between the oscillators, and the fixed torque bias applied to each. Further details are available in Refs. (1; 25).

In realizing PNNs, one needs to contend not only with the challenges imposed by the gap between simulations and reality, but also the gap between individual physical devices. The simulated PNN in Fig. 3 allowed us to study the resilience of PNN models trained using physics-aware training to increasingly large simulation-reality gaps, and to study how models trained on one physical device transfer to slightly different physical devices. These investigations are described in detail in Ref. (1), and provide encouraging indications. First, we find that even when the system used for the forward pass and gradient estimation differ significantly (quantitatively, having differences in untrained parameters that differ by 20%), PAT still results in nearly identical inference accuracy, 89.1 %. This shows that PAT can train physical systems accurately even with only approximate gradients. Second, we find that a model learned using one physical device can be transferred to a second, different device. We find that the maximum difference between the original device and new device for which the trained model transfers with minimal loss of performance is essentially set by simulation-reality gap. In other words, if a PNN model was trained initially and the physical device and simulation model differed by some gap (e.g., in terms of untrained physical parameters), then that trained model will transfer accurately to a second device that differs from the first by less than the original device-simulation gap (1). Although this conclusion is still preliminary, it follows as an intuitive generalization of quantization-aware training (QAT); For QAT, a model trained with x-bit quantization will transfer without loss of accuracy to any hardware whose precision is at least x-bits.

The results presented here are at the proof-of-concept stage, and no attempt has been made to make them competitive with modern digital electronics. For example, the mechanical PNN operates at audio frequencies, and each physical layer takes several 10s of milliseconds. In comparison, a typical laptop computer can execute an inference of a single-layer perceptron, which can achieve similar accuracy as the 3-layer mechanical PNN on the MNIST digit recognition task, in less than a millisecond. The other PNNs we have realized are faster (milliseconds or shorter, depending on the model parameters), but they are still much slower than equivalent digital DNNs, being limited by the speed of our digital-to-analog converters. Nonetheless, theoretical calculations suggest that, given current technologies in nanophotonics, analog electronics, and spintronics, it may be possible to realize PNNs that perform inference more quickly, and with lower energy consumption, compared to equivalent-performance DNNs executed on state-of-the-art GPUs (1). These calculations imply that PNNs could one day offer many orders of magnitude faster and more efficient machine learning than digital electronics. Realizing this performance in real devices will require both better understanding of which physical transformations and network architectures are most effective, as well as careful consideration of the costs associated with digital-analog conversion and digital memory access, which will likely be by far the dominant time and energy bottlenecks in these hypothetical future PNNs (1).

4 Conclusion

We have introduced a new machine learning technique, physical neural networks (PNNs), based on sequences of parameterized physical processes, trained using a modified backpropagation algorithm (1). Our work is at the proof-of-concept stage, and none of our demonstrations are faster or more energy efficient than conventional GPU-based neural network inference. While in theory (1) many physical processes, such as those based on nanophotonics (31; 7) or spintronics (32), offer a route to energy efficient deep learning based on this principle, this remains to be proven experimentally. In the near future, we are optimistic about PNNs being useful for computational imaging or sensing (33; 34; 35). More broadly, there are fundamental questions to answer about PNNs, for which we have only educated guesses so far. For example: Which sorts of physical processes achieve the best balance between trainability, expressiveness, and efficient hardware? What architectures make best use of PNNs, particularly in the context of hybrid physical-digital computation? How can PNNs be used to accelerate and improve the efficiency of training, in addition to inference?

References

[1] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon, Deep physical neural networks trained with backpropagation. *Nature* **601**, 549–555 (2022).

[2] M. Raissi, P. Perdikaris, and G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* **378**, 686–707 (2019).

[3] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, Physics-informed machine learning. *Nature Reviews Physics* **3**, 422–440 (2021).

[4] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Machine learning and the physical sciences. *Reviews of Modern Physics* **91**, 045002 (2019).

[5] H. W. Lin, M. Tegmark, and D. Rolnick, Why does deep and cheap learning work so well? *Journal of Statistical Physics* **168**, 1223–1247 (2017).

[6] G. Furuhata, T. Niiyama, and S. Sunada, Physical deep learning based on optimal control of dynamical systems. *Physical Review Applied* **15**, 034092 (2021).

[7] T. W. Hughes, I. A. Williamson, M. Minkov, and S. Fan, Wave physics as an analog recurrent neural network. *Science Advances* **5**, eaay6946 (2019).

[8] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research* **18**, 6869–6898 (2017).

[9] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, Survey of Machine Learning Accelerators. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, 1–12 (2020).

[10] G. Wetzstein, A. Ozcan, S. Gigan, S. Fan, D. Englund, M. Soljačić, C. Denz, D. A. Miller, and D. Psaltis, Inference in artificial intelligence with deep optics and photonics. *Nature* **588**, 39–47 (2020).

[11] W. E, A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics* **5**, 1–11 (2017).

[12] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 6571–6583 (2018).

[13] K. Nakajima, Physical reservoir computing—an introductory perspective. *Japanese Journal of Applied Physics* **59**, 060501 (2020).

[14] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, Recent advances in physical reservoir computing: a review. *Neural Networks* **115**, 100–123 (2019).

[15] J. F. Miller, S. L. Harding, and G. Tufte, Evolution-in-materio: evolving computation in materials. *Evolutionary Intelligence* **7**, 49–67 (2014).

[16] T. Chen, J. van Gelder, B. van de Ven, S. V. Amitonov, B. de Wilde, H.-C. R. Euler, H. Broersma, P. A. Bobbert, F. A. Zwanenburg, and W. G. van der Wiel, Classification with a disordered dopant-atom network in silicon. *Nature* **577**, 341–345 (2020).

[17] H.-C. R. Euler, M. N. Boon, J. T. Wildeboer, B. van de Ven, T. Chen, H. Broersma, P. A. Bobbert, and W. G. van der Wiel, A deep-learning approach to realizing functionality in nanoelectronic devices. *Nature Nanotechnology* **15**, 992–998 (2020).

[18] M. Nakajima, K. Tanaka, and T. Hashimoto, Neural Schrödinger Equation: physical Law as Neural Network. *arXiv:2006.13541* (2020).

[19] Z. Wu, M. Zhou, E. Khoram, B. Liu, and Z. Yu, Neuromorphic metasurface. *Photonics Research* **8**, 46 (2020).

[20] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning. *Physical Review A* **98**, 032309 (2018).

[21] B. Scellier and Y. Bengio, Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience* **11**, 24 (2017).

[22] M. Stern and A. Murugan, Learning without neurons in physical systems. *arXiv preprint arXiv:2206.05831* (2022).

[23] S. Dillavou, M. Stern, A. J. Liu, and D. J. Durian, Demonstration of Decentralized Physics-Driven Learning. *Physical Review Applied* **18**, 014040 (2022).

[24] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon, *Data repository of the paper "Deep physical neural networks enabled by a backpropagation algorithm for arbitrary physical systems", https://doi.org/10.5281/zenodo.4719150.* (2021).

[25] *https://github.com/mcmahon-lab/Physics-Aware-Training.* (2022).

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, 8024–8035 (2019).

[27] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631 (2019).

[28] H. Xiao, K. Rasul, and R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[29] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems* **34**, 24261–24272 (2021).

[30] *https://github.com/zalandoresearch/fashion-mnist.* (2022).

[31] M. Jankowski, C. Langrock, B. Desiatov, A. Marandi, C. Wang, M. Zhang, C. R. Phillips, M. Lončar, and M. Fejer, Ultrabroadband nonlinear optics in nanophotonic periodically poled lithium niobate waveguides. *Optica* **7**, 40–46 (2020).

[32] J. Grollier, D. Querlioz, K. Camsari, K. Everschor-Sitte, S. Fukami, and M. D. Stiles, Neuromorphic spintronics. *Nature Electronics* **3**, 360–370 (2020).

[33] T. Wang, M. M. Sohoni, L. G. Wright, M. M. Stein, S.-Y. Ma, T. Onodera, M. G. Anderson, and P. L. McMahon, Image sensing with multilayer, nonlinear optical neural networks. *arXiv preprint arXiv:2207.14293* (2022).

[34] L. Mennel, J. Symonowicz, S. Wachter, D. K. Polyushkin, A. J. Molina-Mendoza, and T. Mueller, Ultrafast machine vision with 2D material neural network image sensors. *Nature* **579**, 62–66 (2020).

[35] J. N. Martel, L. K. Mueller, S. J. Carey, P. Dudek, and G. Wetzstein, Neural sensors: Learning pixel exposures for HDR imaging and video compressive sensing with programmable sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **42**, 1642–1653 (2020).

Impact statement

The potential impact of physical neural networks may be considered in (1) computational sensing and imaging, (2) machine learning acceleration, and (3) inspiration of new neural network algorithms. In the near-term, we are most optimistic about the impact of physical neural networks (and related concepts) for sensing and imaging applications, in which the physical device is used to process or partially process data in its native domain. For example, physical neural networks based on mechanics may be useful for acoustic sensors, such as for detecting wake words in edge devices, and optics-based physical neural networks could be well-suited to performing portions of computational image processing in the photonic, analog domain. Such devices could be useful for developing rich scientific sensing tools, and could perhaps find applications in low-power robotics, or high-speed machine vision for feedback control or quality assurance of manufacturing processes. On a longer timescale, physical neural networks could also eventually be useful for performing machine learning at large scales with lower energy costs than conventional digital electronics. Straightforward theoretical analyses suggest that many analog physical systems could perform machine learning inference, and perhaps training/learning, with vastly lower power consumption than current digital electronics, in principle by many orders of magnitude. Designing practical systems that achieve this performance will, however, require much more research and engineering. Supposing that such highly efficient physical analog neural networks are ultimately constructed, they would, on the one hand, vastly reduce the energy and carbon costs of large-scale machine learning models, making large-scale models accessible to a much broader range of users and researchers. On the other hand, they may be even more challenging than conventional neural networks to interpret or to provide mathematical guarantees for, or will at least require new techniques beyond those already developed for conventional neural networks. Overall, we emphasize that the work described here is at an early stage and is primarily conceptual, rather than an established technology. Thus, while the above comments are based on reasonable assumptions and in some cases calculations, they are of course fundamentally speculations.

Checklist

1. For all authors...
    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
    (b) Did you describe the limitations of your work? [Yes] We did not include an entire section for this, but it is mentioned throughout. See also discussion and supplementary material of (1) for very extensive discussion of limitations.
    (c) Did you discuss any potential negative societal impacts of your work? [No] This is very early stage basic research
    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
    (a) Did you state the full set of assumptions of all theoretical results? [N/A]
    (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
    (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We cited our data and code.
    (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [No] Please see the original paper for these details (1)
    (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
    (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
    (a) If your work uses existing assets, did you cite the creators? [Yes]
    (b) Did you mention the license of the assets? [N/A]
    (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
    (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
    (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
    (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
    (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
    (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]