SCIENCE CHINA
Information Sciences

• Supplementary File •

Robust Forward–Forward training of physical neural networks enabled by 2D materials

Xueyuan ZHANG[†1], Pengfei WANG[†2], Yichen ZHAO[2], Chen PAN[1], Yinzhi HUANG[2], Yuekun YANG[3], Cong WANG[3], Bin CHENG[1,2], Wentao YU[1] & Shi-Jun LIANG[2,4]

[1]*Institute of Interdisciplinary Physical Sciences, School of Physics, Nanjing University of Science and Technology, Nanjing 210094, China;*
[2]*Institute of Brain-Inspired Intelligence, National Laboratory of Solid State Microstructures, School of Physics, Collaborative Innovation Center of Advanced Microstructures, Nanjing 210093, China;*
[3]*State Key Laboratory for Novel Software Technology, School of Intelligence Science and Technology, Nanjing University, Suzhou, 215163, China*
[4]*Chemistry and Biomedicine Innovation Center (ChemBIC), Naning University, Nanjing 210093, China*

Appendix A The supplementary information of the WSe_2 floating-gate device

Motivated by recent advances in two-dimensional (2D) material–enabled neuromorphic sensory and optoelectronic systems[1-2], this work employs a WSe_2 floating-gate device as a programmable nonlinear memory–computing element to realize a physical neural network, and this appendix provides supplementary device details. The device adopts a Schottky-type heterostructure composed of $WSe_2/Al_2O_3/Au/Al_2O_3$, where WSe_2 serves as the channel, Au as the floating gate, and the Al_2O_3 layers function as tunneling and blocking dielectrics. Unlike conventional floating-gate transistors with ohmic contacts and linear output, the device continuously modulates the source–drain Schottky barrier through dual-field coupling between the drain and floating-gate electric fields. In Figure A1(b), the asymmetry of the Schottky barriers under opposite V_{DS} bias conditions arises from unequal electrostatic coupling between the floating gate and the source and drain electrodes. Specifically, the effective Schottky barriers at the source and drain are determined by the relative electrostatic potentials between the floating gate and each contact. The source-side barrier is controlled by $V_{FG\text{-}S} = V_{FG} - V_S$ (with V_S fixed at 0 V), whereas the drain-side barrier is controlled by $V_{FG\text{-}D} = V_{FG} - V_D$. Consequently, the rectification behavior arises from the combined influence of the floating-gate potential V_{FG}, the applied drain bias V_D, and the fixed source potential V_S. When a positive $V_{program,CG}$ is applied to control gate, negative charges are stored in the floating gate, producing a negative V_{FG} and inducing a p-type channel (Fig. A1(b)(i)). Under $V_D > 0$, the enhanced relative potential $|V_{FG\text{-}D}| = |V_{FG}| + |V_D|$ causes strong upward band bending at the drain interface, significantly thinning the Schottky barrier and enabling efficient hole injection, corresponding to the ON state (Fig. A1(b)(ii)). In contrast, under $V_D < 0$, hole injection from the source is limited because $|V_{FG\text{-}S}| = |V_{FG}|$ is insufficient to induce strong band bending, resulting in a thicker barrier, suppressed injection, and an OFF state (Fig. A1(b)(iv)), thereby producing forward rectification. Similarly, when a negative $V_{program,CG}$ is applied to control gate, positive charges are trapped in the floating gate, inducing an n-type channel (Fig. A1(b)(v)). In this case, electron injection from the drain contact under $V_D < 0$ is more efficient than injection from the source under $V_D > 0$, resulting in reverse rectification behavior (Fig. A1(b)(vi-viii)). This coupling selectively controls the asymmetric injection efficiency of electrons and holes, resulting in intrinsic nonlinear output characteristics dominated by thermionic emission and tunneling transport across the Schottky barrier. By applying programming pulses of opposite polarities to the gate, the floating-gate charge state can be continuously tuned, reversibly modulating both the height and polarity of the Schottky barrier. This enables bidirectional rectification switching, producing over 100 stable current levels for each polarity and giving rise to a robust, nonvolatile, and continuously tunable nonlinear response.

The device's I-V characteristics under various programmed states can be described by $f(x) = W(e^{\beta x} - 1)$, where $f(x)$ is the output current, x the input voltage, W a programmable parameter representing the stored charge state, and β a constant intrinsic parameter. Here, x corresponds to the input voltage V_{in}. In the experiments, $G(V_{in})$ ranges from 0-2 V, but for mathematical convenience we normalize x to the interval 0-1. The factor-of-two scaling between x and V_{in} is already incorporated into the choice of the parameter β. This exponential, tunable nonlinearity forms the physical basis for nonlinear synaptic weighting and dynamic modulation in the PDNN.

To evaluate the retention behavior of the programmed charge states under ambient conditions, we monitored the read currents of multiple programmable states as a function of time. As shown in Supplementary Fig. A3, the programmed current levels remain stable and well distinguishable during the measurement period in both the forward rectifying (read at $V_{DS} = 2$ V) and the reverse rectifying (read at $V_{DS} = -2$ V) characteristics. Only slight current fluctuations are observed, indicating reliable retention of the programmable current states under ambient conditions.

* Corresponding author (email: chenpan@njust.edu.cn, sjliang@nju.edu.cn)
† These authors contributed equally to this work.

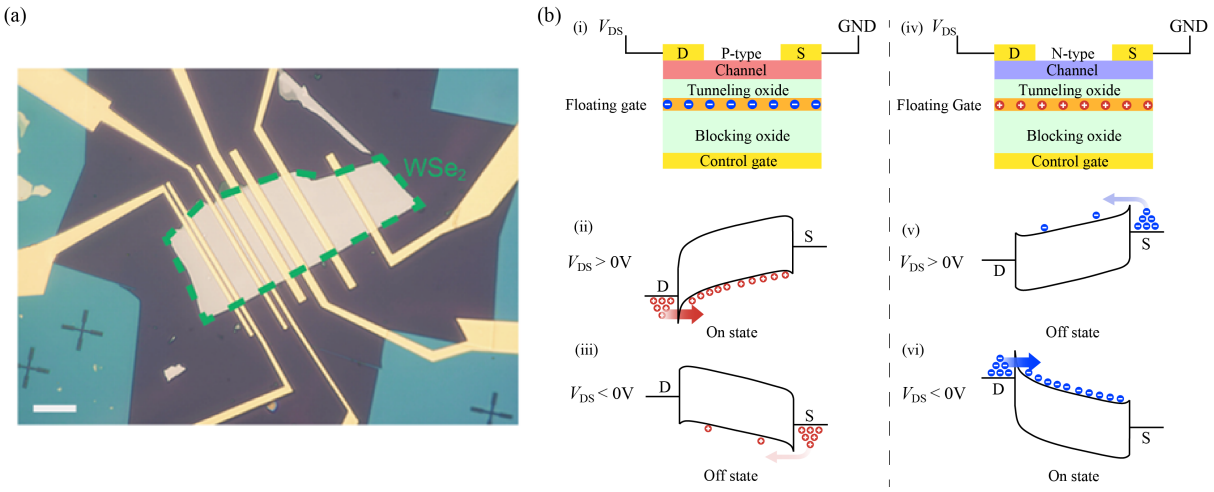


Figure A1 (a) Optical images of the WSe_2 floating-gate devices, featuring channels of varying lengths. Variations in channel length do not alter the fundamental device behavior. Scale bar, 10 µm. The source and drain electrodes of the WSe_2 floating-gate FET are fabricated using identical metal stacks, consisting of Pd (5 nm) / Au (50 nm). (b) (i) Schematic of the biasing configuration applied to each terminal during device operation, along with the corresponding band diagram illustrating that negative charge stored in the floating gate induces a p-doped channel. (ii) - (iv) show the energy-band profiles under this floating-gate state for different V_{DS} polarities. When W^+ and W^- > 0 V, the device is in the ON state, whereas for $V_{DS} < 0$, it switches to the OFF state, demonstrating forward rectification. When $V_{DS} = 0$ V, no current flows due to the equal electrostatic potential between the source and drain. (v) Schematic of the biasing configuration when positive charge is stored in the floating gate, resulting in an n-doped channel. (vi) - (viii) present the corresponding band diagrams under different V_{DS} polarities, showing that the device turns OFF for $V_{DS} > 0$V and turns ON for $V_{DS} < 0$ V, thereby exhibiting reverse rectification. As in the forward-rectifying case, no current flows at $V_{DS} = 0$ V because of the absence of a driving electric field.

Appendix B Schematic illustration of the Forward–Forward algorithm

The Forward-Forward (FF) algorithm is a layer-wise training approach for neural networks, in which positive and negative samples are propagated through each layer separately and their "goodness" scores are evaluated and compared. For a given layer, the activations produced by positive inputs and negative inputs are first computed, and a scalar goodness measure is defined on these activations. The learning objective at that layer is then to increase the goodness of positive samples while decreasing that of negative samples, typically by minimizing a local loss function that encourages the goodness of positive samples to exceed a prescribed threshold and that of negative samples to fall below it. By repeating this procedure independently for each layer, the network progressively shapes internal representations that separate positive and negative data in feature space, without relying on error signals propagated from deeper layers[3]. Such a purely forward and local learning rule avoids the need for global backpropagated gradients, alleviates issues such as vanishing gradients, and is naturally compatible with hardware-friendly implementations and layer-wise or asynchronous training schemes[4]. (see Fig. B1)

Appendix C Hardware architecture and algorithm mapping strategy

Recent reviews have summarized, from different perspectives, neuromorphic hardware spanning materials and device modeling through array- and system-level integration, as well as hardware-efficient computing paradigms under edge constraints[5-7]. This appendix outlines the hardware architecture and the algorithm-to-hardware mapping strategy adopted in this work, and explains how the training procedure is implemented on the hardware side. The proposed physical computing platform employs a tunable nonlinear floating-gate memory array as the core computational unit (Fig. 1(b)). As illustrated in Supplementary Fig. C1(a), the array is organized into multiple banks. Input signals are first latched and scheduled before being applied to each bank. Within the bank, the conductance of individual floating-gate devices can be nonlinearly tuned through controllable adjustment of the stored charge. At the array level, the resulting currents are summed along shared lines, enabling physical realization of vector–matrix multiplication. During inter-layer propagation, outputs from different banks are collected by shared reference circuits and routed to a multichannel ADC for parallel analog-to-digital conversion. The digitized signals are then transmitted to either the host computer or subsequent array layers, facilitating cascaded computation across multiple network layers. The platform also incorporates an on-array weight-update mechanism: error signals are locally used to adjust the floating-gate charge states, providing continuously tunable, nonvolatile weights and enabling dynamic optimization during training. With unified scheduling and buffering control, the system achieves scalable and stable operation, providing full hardware support for physical neural network computation under the FF algorithm.

Based on the proposed hardware platform, we further clarify how the FF algorithm is mapped onto the physical system, illustrating how layer-wise training and optimization are implemented on the floating-gate memory array and its peripheral circuit modules. The overall training procedure follows the FF algorithm proposed by Hinton *et al.*[3] and the optimized goodness calculation formula by Momeni *et al.*[8], adopting its principle of local layer-wise training and physical learning for both supervised and unsupervised models in physical neural networks (PDNNs). The system receives positive and negative samples as inputs; these signals undergo successive nonlinear transformations and linear combinations in each

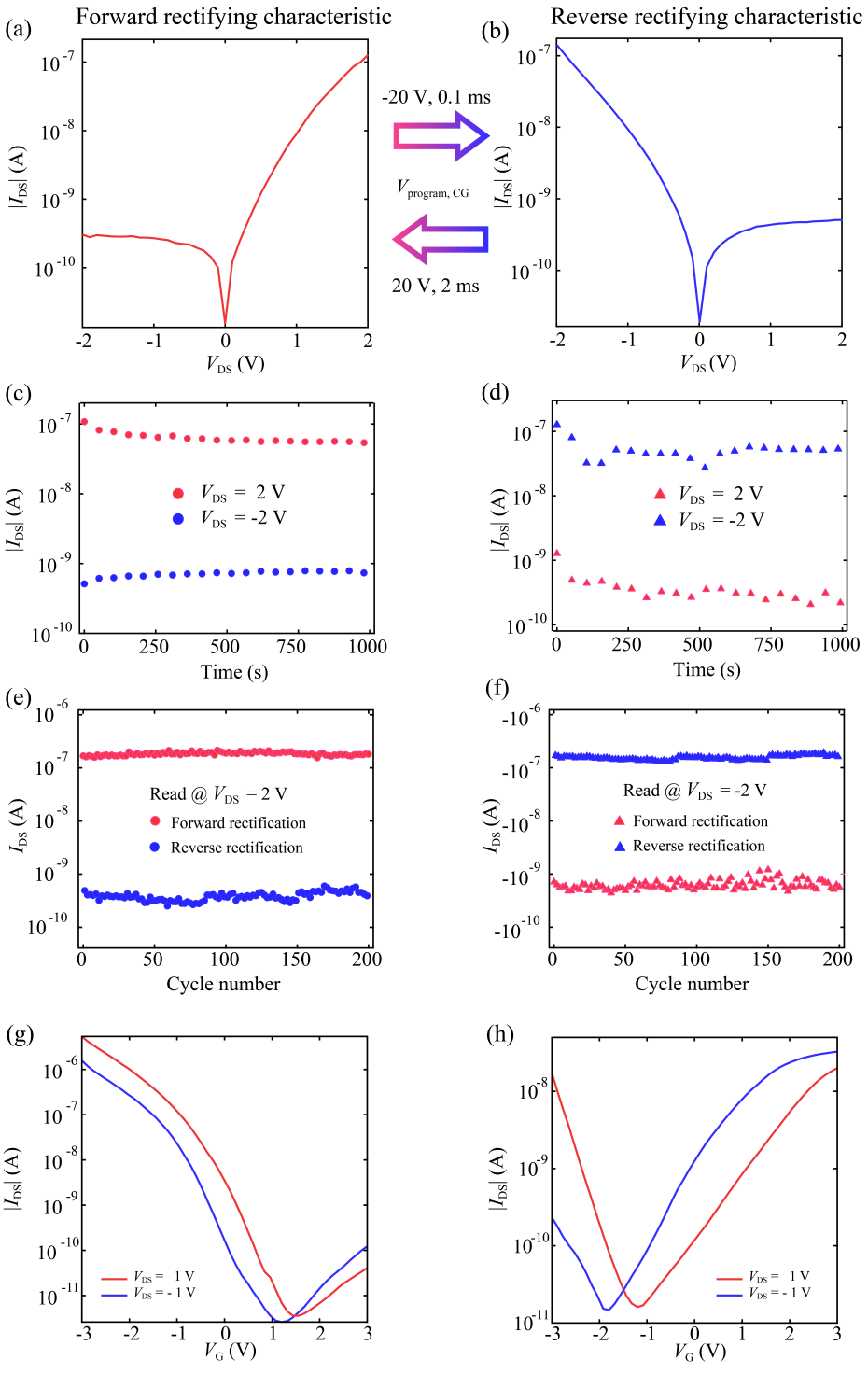


Figure A2 (a) and (b) demonstrate that applying ±20 V, 0.1 ms or 2 ms programming pulses to the control gate enables reversible switching between forward and reverse rectification states. (c) and (d) show the retention characteristics of the device. After programming the device into the forward-rectifying state (corresponding to (a)) or the reverse-rectifying state (corresponding to (b)), the output current is read under V_{DS} = 2 V and V_{DS} = –2 V, respectively. (e) and (f) present the endurance characteristics. When read at V_{DS} = 2 V (corresponding to (e)) or –2 V (corresponding to (f)), the device maintains stable rectification behavior over 200 repeated program–erase cycles.

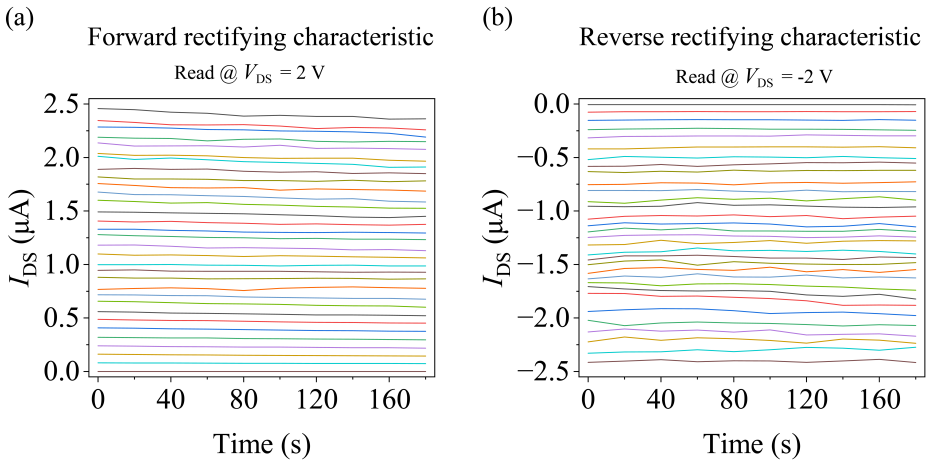


Figure A3 Long-term retention characteristics of programmable current levels under ambient conditions. Measured retention data of 32 distinct states for (a) forward rectifying and (b) reverse rectifying characteristics.

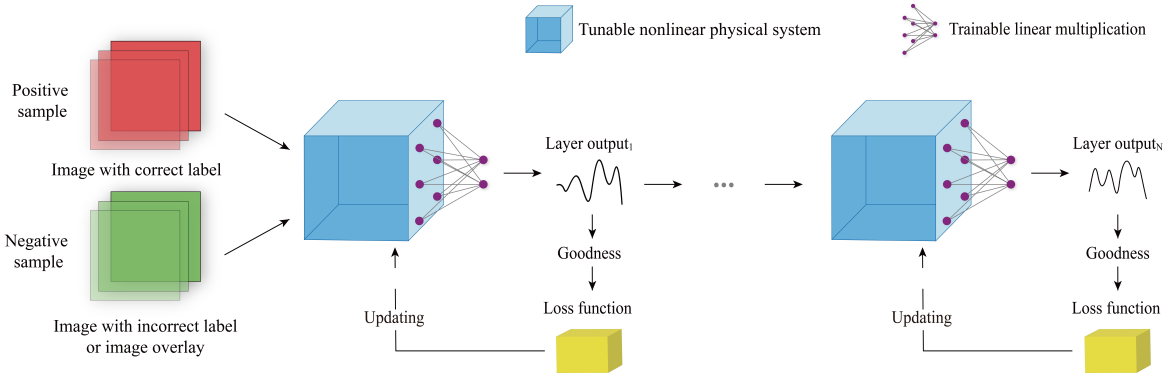


Figure B1 Schematic illustration of the Forward–Forward algorithm.

layer, generating intermediate states through layer-by-layer mappings. A goodness function is then used to evaluate the output of each layer, and the resulting error signal is fed back to the same layer via a local loss function, thereby realizing a local training process without backpropagation.

The detailed training flow is shown in Supplementary Fig. C1(b) and consists of both supervised and unsupervised phases. In supervised local layer-wise training, a local teaching signal must be provided. Before training starts, a random target vector $\xi^{(l)}$ is defined for each layer to guide learning, and the trainable weights $W^{(l)}$ are initialized. Meanwhile, the input images are used to construct the corresponding positive data $x_{pos}^{(l)}$ and negative data $x_{neg}^{(l)}$. In contrast, during unsupervised training the learning process does not rely on external labels or explicit class information, so layer-wise target vectors cannot be predefined. Instead, training focuses on capturing the relative structural relationships and distributional differences among samples. Consequently, the goodness evaluation is no longer based on matching a preset target vector, but is realized through a contrastive learning mechanism that maximizes consistency among positive samples while minimizing similarity between positive and negative samples.

During the training process, the nonlinear response of each layer is realized by nonlinear floating-gate memory devices. For each layer, the input is delivered by ADCs as programming pulses, and the output is generated by multiplying the nonlinear response of the floating-gate memory array with the trainable weight matrix $W^{(l)}$, with the mapping relationship expressed as:

$$f^{(l)}\left(x^{(l)}\right) = W^{(l)}(e^{\beta x^{(l)}} - 1),$$

where $x^{(l)}$ and $f^{(l)}$ represent the input electrical signal and the nonlinear mapping function of the l-th layer, respectively, and β is an intrinsic device constant (≈ 6).

The training procedure departs from traditional forward-backward propagation and instead employs two separate forward passes: one positive forward pass and one negative forward pass, each using different input signals. The positive forward pass is given by:

$$y_{pos}^{(l)} = f^{(l)}\left(x_{pos}^{(l)}\right),$$

where the input includes the dataset and its correct labels. The negative forward pass is given by:

$$y_{neg}^{(l)} = f^{(l)}\left(x_{neg}^{(l)}\right),$$

where the input includes the dataset with incorrect labels or data constructed from different image combinations. The results are summed to comprehensively reflect the physical behavior of the array. In the logic update module, the cosine similarity between the output of each layer and a random vector $\xi^{(l)}$ is computed to determine the corresponding goodness. In supervised training, the goodness function is defined as the cosine similarity between the output and a random vector sampled from a normal distribution. For the positive sample, the goodness is:

$$g\left(y_{pos}^{(l)}\right) = cos_{sim}\left(y_{pos}^{(l)}, \xi^{(l)}\right),$$

and for the negative sample, the goodness is:

$$g\left(y_{neg}^{(l)}\right) = cos_{sim}\left(y_{neg}^{(l)}, \xi^{(l)}\right).$$

During unsupervised training, the goodness function is defined as the cosine similarity between the outputs of the positive and negative samples in each layer:

$$g\left(y_{pos}^{(l)}, y_{neg}^{(l)}\right) = cos_{sim}(y_{pos}^{(l)}, y_{neg}^{(l)}).$$

Meanwhile, unsupervised learning requires the introduction of a Softmax activation function and a cross-entropy loss function at the final layer to guide the training process.

Subsequently, the trainable weight matrix $W^{(l)}$ of each layer is updated by minimizing a custom loss function. In the supervised training process, the loss function is given by:

$$L^{(l)} = \log\left(1 + exp\left\{-\theta\left[cos_{sim}\left(y_{pos}^{(l)}, \xi^{(l)}\right) - cos_{sim}\left(y_{neg}^{(l)}, \xi^{(l)}\right)\right]\right\}\right).$$

In the above expressions, cos_{sim} denotes the cosine similarity between two vectors, θ is a scaling factor, and $\xi^{(l)}$ is the random vector for the current layer.

According to the local update rule of the FF algorithm, the trainable weights $W^{(l)}$ in each layer can be updated. We define:

$$s^{(l)} = g\left(y_{pos}^{(l)}\right) - g\left(y_{neg}^{(l)}\right) = cos_{sim}\left(y_{pos}^{(l)}, \xi^{(l)}\right) - cos_{sim}\left(y_{neg}^{(l)}, \xi^{(l)}\right).$$

The gradient of the weight matrix in this layer is then given by:

$$\frac{\partial L^{(l)}}{\partial W^{(l)}} = \frac{\partial L^{(l)}}{\partial s^{(l)}} \frac{\partial s^{(l)}}{\partial W^{(l)}}$$

$$
\begin{aligned}
&= \frac{-\theta \exp\left(-\theta s^{(l)}\right)}{1+exp\left(-\theta s^{(l)}\right)} \left[\frac{\partial cos_{sim}\left(y_{pos}^{(l)}, \xi^{(l)}\right)}{\partial y_{pos}^{(l)}} \cdot \frac{\partial y_{pos}^{(l)}}{\partial W^{(l)}} - \frac{\partial cos_{sim}\left(y_{neg}^{(l)}, \xi^{(l)}\right)}{\partial y_{neg}^{(l)}} \cdot \frac{\partial y_{neg}^{(l)}}{\partial W^{(l)}} \right] \\
&= \frac{-\theta \exp\left\{-\theta \left[\cos_{sim}\left(y_{pos}^{(l)}, \xi^{(l)}\right) - \cos_{sim}\left(y_{neg}^{(l)}, \xi^{(l)}\right)\right]\right\}}{1+\exp\left\{-\theta \left[\cos_{sim}\left(y_{pos}^{(l)}, \xi^{(l)}\right) - \cos_{sim}\left(y_{neg}^{(l)}, \xi^{(l)}\right)\right]\right\}} \\
&\quad \left[\left(\frac{\xi^{(l)}}{\|y_{pos}^{(l)}\| \|\xi^{(l)}\|} - \frac{\left(y_{pos}^{(l)} \cdot \xi^{(l)}\right) y_{pos}^{(l)}}{\|y_{pos}^{(l)}\|^3 \|\xi^{(l)}\|} \right) \times \left(e^{\beta x_{pos}^{(l)}} - 1 \right)^{\mathrm{T}} \right. \\
&\quad \left. - \left(\frac{\xi^{(l)}}{\|y_{neg}^{(l)}\| \|\xi^{(l)}\|} - \frac{\left(y_{neg}^{(l)} \cdot \xi^{(l)}\right) y_{neg}^{(l)}}{\|y_{neg}^{(l)}\|^3 \|\xi^{(l)}\|} \right) \times \left(e^{\beta x_{neg}^{(l)}} - 1 \right)^{\mathrm{T}} \right] \\
&\propto g'\left(y_{pos}^{(l)}\right) \times \left(e^{\beta x_{pos}^{(l)}} - 1 \right)^{\mathrm{T}} - g'\left(y_{neg}^{(l)}\right) \times \left(e^{\beta x_{neg}^{(l)}} - 1 \right)^{\mathrm{T}}.
\end{aligned}
$$

For unsupervised training, the loss function is expressed as:

$$
L^{(l)} = \log\left(1 + exp\left[\theta \cos_{sim}(y_{pos}^{(l)}, y_{neg}^{(l)})\right]\right).
$$

In the above expression, cos_{sim} denotes the cosine similarity between two vectors, and θ is a scaling factor.

According to the local update rule of the FF algorithm, the trainable weights $W^{(l)}$ in each layer can be updated. We define:

$$
s^{(l)} = g\left(y_{pos}^{(l)}, y_{neg}^{(l)}\right) = cos_{sim}(y_{pos}^{(l)}, y_{neg}^{(l)}).
$$

Then

$$
\begin{aligned}
\frac{\partial L^{(l)}}{\partial W^{(l)}} &= \frac{\partial L^{(l)}}{\partial s^{(l)}} \frac{\partial s^{(l)}}{\partial W^{(l)}} \\
&= \frac{\theta \exp\left(\theta s^{(l)}\right)}{1+exp\left(\theta s^{(l)}\right)} \left[\frac{\partial cos_{sim}\left(y_{pos}^{(l)}, y_{neg}^{(l)}\right)}{\partial y_{pos}^{(l)}} \cdot \frac{\partial y_{pos}^{(l)}}{\partial W^{(l)}} + \frac{\partial cos_{sim}\left(y_{pos}^{(l)}, y_{neg}^{(l)}\right)}{\partial y_{neg}^{(l)}} \cdot \frac{\partial y_{neg}^{(l)}}{\partial W^{(l)}} \right] \\
&= \frac{\theta \exp\left[\theta \cos_{sim}\left(y_{pos}^{(l)}, y_{neg}^{(l)}\right)\right]}{1+\exp\left[\theta \cos_{sim}\left(y_{pos}^{(l)}, y_{neg}^{(l)}\right)\right]} \\
&\quad \left[\left(\frac{y_{neg}^{(l)}}{\|y_{pos}^{(l)}\| \|y_{neg}^{(l)}\|} - \frac{\left(y_{pos}^{(l)} \cdot y_{neg}^{(l)}\right) y_{pos}^{(l)}}{\|y_{pos}^{(l)}\|^3 \|y_{neg}^{(l)}\|} \right) \times \left(e^{\beta x_{pos}^{(l)}} - 1 \right)^{\mathrm{T}} \right. \\
&\quad \left. + \left(\frac{y_{pos}^{(l)}}{\|y_{pos}^{(l)}\| \|y_{neg}^{(l)}\|} - \frac{\left(y_{pos}^{(l)} \cdot y_{neg}^{(l)}\right) y_{neg}^{(l)}}{\|y_{neg}^{(l)}\|^3 \|y_{pos}^{(l)}\|} \right) \times \left(e^{\beta x_{neg}^{(l)}} - 1 \right)^{\mathrm{T}} \right] \\
&\propto g'\left(y_{pos}^{(l)}, y_{neg}^{(l)}\right) \times \left(e^{\beta x_{pos}^{(l)}} - 1 \right)^{\mathrm{T}} + g'\left(y_{pos}^{(l)}, y_{neg}^{(l)}\right) \times \left(e^{\beta x_{neg}^{(l)}} - 1 \right)^{\mathrm{T}}.
\end{aligned}
$$

After updating the weights in the current layer, the output is transmitted through ADCs and serves as the input for the next layer, thus achieving the layer-wise forward mapping of the entire PDNN. In addition, in robustness experiments, to simulate the intrinsic device non-uniformity, Gaussian noise $\mathcal{N}(0, \sigma^2)$ is introduced to the intrinsic parameter β. The forward processes of the FF algorithm for positive and negative samples are given by:

$$
f_{physical}^{(l)}(x_{pos}^{(l)}, \beta + \varepsilon) = W^{(l)}(e^{(\beta+\varepsilon)x_{pos}^{(l)}} - 1), \varepsilon \sim \mathcal{N}\left(0, \sigma^2\right),
$$

$$
f_{physical}^{(l)}(x_{neg}^{(l)}, \beta + \varepsilon) = W^{(l)}(e^{(\beta+\varepsilon)x_{neg}^{(l)}} - 1), \varepsilon \sim \mathcal{N}\left(0, \sigma^2\right).
$$

In supervised training, the gradient update formula is:

$$
\begin{aligned}
\frac{\partial L^{(l)}}{\partial W^{(l)}} &= \frac{\partial L^{(l)}}{\partial s^{(l)}} \frac{\partial s^{(l)}}{\partial W^{(l)}} \\
&= \frac{-\theta \exp\left\{-\theta \left[\cos_{sim}\left(y_{pos}^{(l)}, \xi^{(l)}\right) - \cos_{sim}\left(y_{neg}^{(l)}, \xi^{(l)}\right)\right]\right\}}{1+\exp\left\{-\theta \left[\cos_{sim}\left(y_{pos}^{(l)}, \xi^{(l)}\right) - \cos_{sim}\left(y_{neg}^{(l)}, \xi^{(l)}\right)\right]\right\}} \\
&\quad \left[\left(\frac{\xi^{(l)}}{\|y_{pos}^{(l)}\| \|\xi^{(l)}\|} - \frac{\left(y_{pos}^{(l)} \cdot \xi^{(l)}\right) y_{pos}^{(l)}}{\|y_{pos}^{(l)}\|^3 \|\xi^{(l)}\|} \right) \times \left(e^{(\beta+\varepsilon)x_{pos}^{(l)}} - 1 \right)^{\mathrm{T}} \right.
\end{aligned}
$$

$$-\left(\frac{\xi^{(l)}}{\|y_{neg}^{(l)}\|\,\|\xi^{(l)}\|}-\frac{(y_{neg}^{(l)}\cdot\xi^{(l)})\,y_{neg}^{(l)}}{\|y_{neg}^{(l)}\|^3\,\|\xi^{(l)}\|}\right)\times\left(e^{(\beta+\varepsilon)x_{neg}^{(l)}}-1\right)^{\mathrm{T}}\Bigg]$$
$$\propto g'\left(y_{pos}^{(l)}\right)\times\left(e^{(\beta+\varepsilon)x_{pos}^{(l)}}-1\right)^{\mathrm{T}}-g'\left(y_{neg}^{(l)}\right)\times\left(e^{(\beta+\varepsilon)x_{neg}^{(l)}}-1\right)^{\mathrm{T}},$$

where $f_{physical}^{(l)}$ is the actual mathematical model of each layer during the forward process of the FF and BP algorithms. Similarly, the forward process of the BP algorithm is expressed as:

$$f_{physical}^{(l)}(x^{(l)},\beta+\varepsilon)=W^{(l)}(e^{(\beta+\varepsilon)x^{(l)}}-1),\varepsilon\sim\mathcal{N}\left(0,\sigma^2\right).$$

Since the backward process in BP strictly follows the forward mathematical form but the noise cannot be fully quantified, the expression referenced during BP backward propagation is:

$$f_{model}^{(l)}(x^{(l)},\beta)=W^{(l)}(e^{\beta x^{(l)}}-1),$$

where $f_{model}^{(l)}$ is the ideal mathematical model used for gradient calculation in BP. Thus, the weight update gradient in BP is given by:

$$\Delta W^{(l)}\propto-\frac{\partial L^{(l)}}{\partial W^{(l)}}=-\frac{\partial L^{(l)}}{\partial f_{model}^{(l)}}\frac{\partial f_{model}^{(l)}}{\partial W^{(l)}}=-\frac{\partial L^{(l)}}{\partial f_{model}^{(l)}}\times\left(e^{\beta x^{(l)}}-1\right)^{\mathrm{T}}.$$

Appendix D Numerical simulations for feasibility verification

To verify the feasibility of applying the Forward-Forward (FF) algorithm to training physical neural networks (PDNNs), we conducted numerical simulations on the MNIST handwritten digit dataset using two types of network topologies: (i) fully connected forward-only network, and (ii) a hybrid network consisting of convolutional layers followed by forward-only layers. Both supervised and unsupervised training paradigms were examined. The supervised setting highlights the discriminative capability of FF under explicit label constraints, whereas the unsupervised setting demonstrates its discriminative capability in the absence of label signals, particularly when adopting a deeper network architecture.

With respect to supervised data construction (Supplementary Fig. D1(a)), we incorporate the class label by encoding it as a one-dimensional "label row" and embedding it into the first row of the input image. The label row has the same width as the image and is written into (i.e., replaces) the first row of the input tensor. For a K-class task, the label row is represented as a K-dimensional one-hot vector (padded/extended to match the image width if needed): the entry corresponding to the target class is set to 1 and all others are set to 0. For example, for the digit "3", the third entry in the label row is set to 1 while the remaining entries are 0. Positive samples are constructed by embedding the correct label row, whereas negative samples are constructed by embedding an incorrect label row obtained by randomly shifting/permuting the one-hot position (i.e., assigning the "1" to a non-matching class such as "5"). The label information is embedded and concatenated with the input features, enabling the model to directly capture discriminative features that separate the true class from confusing classes. This design enables FF training to directly contrast the forward-pass goodness between matched (positive) and mismatched (negative) image–label inputs under a local goodness criterion. In the unsupervised training, the positive dataset consists of the original images, while the negative dataset is formed by synthetic images obtained by applying masking and coupling operations to the originals. This strategy disrupts the structural content while avoiding trivial all-black or pure-noise samples, encouraging the model to learn robust features that preserve the underlying semantic structure. Compared with the supervised task, the unsupervised task must rely on the network itself to distill high-level semantic features from the input data in the absence of explicit label constraints. To this end, we adopt a deeper stack of forward-only layers in the unsupervised setting to enhance feature extraction capability and robustness, ensuring that the network can still achieve stable representations and classification performance when it only exploits structural differences in the input. Together, these two training modes demonstrate the applicability and advantages of the FF algorithm under different supervision conditions.

Supplementary Fig. D1(b) and Fig. D1(d) illustrate the network architectures used in the numerical simulations: a fully connected forward-only network and a convolutional forward-only network. The fully connected network serves as the most direct baseline, verifying the feasibility of the FF algorithm under a matrix–vector computation framework. The convolutional network, on the other hand, incorporates local receptive fields and weight sharing, which are commonly used in image tasks, and is employed to further examine the applicability of FF in architectures with convolutional priors. The comparison between these two architectures spans typical models from simple to more complex, and also highlights the adaptability of the proposed physical platform to different task scenarios.

More specifically, the supervised fully connected FF network takes a 784-dimensional input (28×28 images) and applies two forward-only fully connected layers with 1000 and 10 neurons, respectively. The FF layers are trained using positive and negative samples constructed by embedding the correct or incorrect labels into the input, enabling a direct assessment of the discriminative capability of FF under explicit label constraints. The supervised convolutional FF network adopts a lightweight convolutional feature extractor followed by forward-only layers. Specifically, the input is first processed by a convolution–pooling stage (Conv 9×9, 16 channels, followed by 2×2 max pooling) and then flattened before being fed into two forward-only fully connected layers ($400\to840\to10$). This architecture is used to examine FF training when convolutional inductive priors are introduced for image classification.

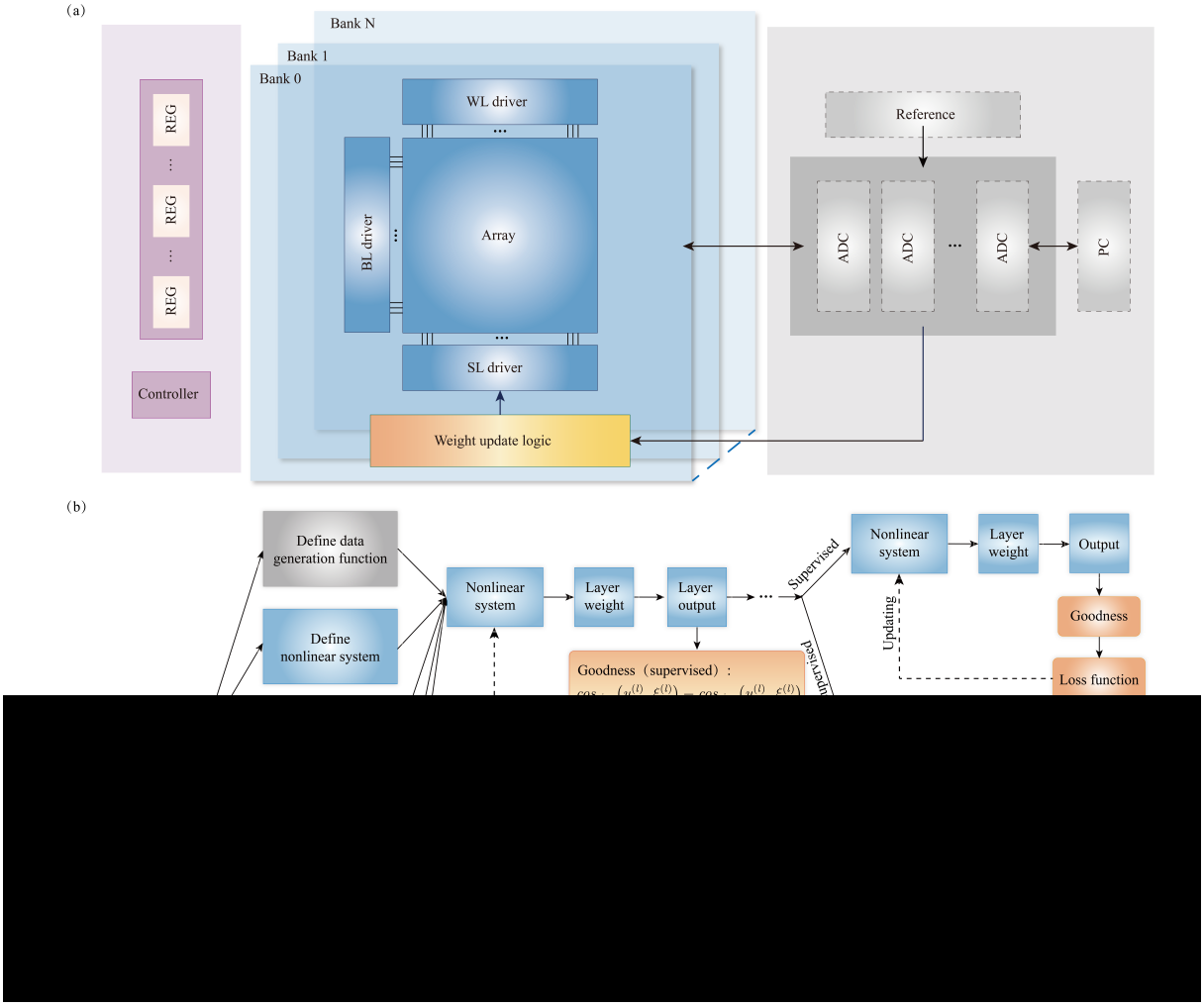


Figure C1 Physical system platform for training with the FF algorithm and algorithm mapping. (a) Overview of the physical system architecture based on a WSe_2 floating-gate device array. The diagram shows the cascaded circuit structure of the array and the necessary peripheral modules, as well as the signal paths that implement forward propagation. (b) Digital modeling flowchart for supervised and unsupervised training. Purple blocks denote the signal input module; blue blocks represent the mathematical model of the physical system; orange blocks are used only for supervised training; yellow blocks are used only for unsupervised training; gray blocks correspond to modules configured on the computer.

For the unsupervised fully connected FF setting, a deeper architecture is adopted to compensate for the absence of label supervision. The network consists of a 784-dimensional input followed by four forward-only fully connected layers with 2000 neurons each. The outputs of the last three forward-only layers are concatenated and passed to a linear 10-way classifier (6000 $\rightarrow$ 10), thereby improving feature discrimination based purely on structural differences in the input. Finally, the unsupervised convolutional FF network combines a LeNet-5-like convolutional front end with a deep forward-only backbone. It employs two convolution–pooling stages (Conv 5 $\times$ 5, 6 channels, followed by 2 $\times$ 2 pooling; Conv 5 $\times$ 5, 16 channels, followed by 2 $\times$ 2 pooling) to extract compact features, which are then fed into four forward-only fully connected layers with 2000 neurons each. The outputs of the last three forward-only layers are concatenated and passed to a separately trained linear classifier to produce 10 output classes.

The detailed training configurations and hyperparameters for different network architectures are summarized as follows. For the supervised fully connected FF network, the Adam optimizer was used with the following hyperparameters: learning rate = 0.001, betas = (0.9, 0.999), eps = 1×10^{-8}, weight decay = 1×10^{-5}, and amsgrad = False. The batch size was set to 100, and the network was trained for 200 epochs. During the local training of each layer, 2 internal updates were performed for each mini-batch. The threshold parameter in the loss function was set to 4. For the unsupervised fully connected FF network, both the forward layers and the final linear classifier used the Adam optimizer with the following hyperparameters: learning rate = 0.001, betas = (0.9, 0.999), eps = 1×10^{-8}, weight decay = 1×10^{-5}, and amsgrad = False. The batch size was set to 64, and the network was trained for 200 epochs. The threshold parameter was set to 4. For the supervised convolutional FF network, the Adam optimizer was used with the following hyperparameters: learning rate = 0.001, betas = (0.9, 0.999), eps = 1×10^{-8}, weight decay = 1×10^{-5}, and amsgrad = False. The batch size was set to 100, and the network was trained for 200 epochs. The threshold parameter was set to 4. For the unsupervised convolutional FF network, both the forward layers and the final linear classifier used the Adam optimizer with the following hyperparameters: learning rate = 0.001, betas = (0.9, 0.999), eps = 1×10^{-8}, weight decay = 1×10^{-5}, and amsgrad = False. The batch size was set to 64, and the network was trained for 200 epochs. The threshold parameter was set to 4.

By comparing these four architectures, the applicability and robustness of the FF algorithm across different network types and supervision regimes can be comprehensively assessed. In the numerical simulation results, the FF networks achieve satisfactory classification performance under supervised conditions: the supervised FF attains accuracies of 85.99% and 85.94% on the training and test sets, respectively, while the convolutional FF network further improves the performance to 88.37% and 89.61% (see Supplementary Fig. D1(c)). Under unsupervised conditions, the deeper architectures exhibit stronger representation capability: the FF reaches 92.41% and 91.85% accuracy on the training and test sets, respectively, and the convolutional FF network further improves these to 94.53% and 94.50% (see Supplementary Fig. D1(e)). In summary, the supervised simulations highlight the discriminative power of FF under label constraints, whereas the unsupervised simulations demonstrate its ability to learn robust representations in the absence of labels. At the same time, the convolutional architectures provide additional performance gains in both paradigms, indicating good compatibility between the FF algorithm and convolutional priors.

Appendix E Numerical simulations for robustness evaluation

We compared the robustness of the conventional backpropagation (BP) algorithm and the FF algorithm using the same physical system model. (Detailed formulas are provided in Supplementary Material C.) As illustrated in Fig. E1, additive noise was introduced to emulate the intrinsic nonidealities and device-level nonuniformity of the hardware. During forward propagation, the inference of both FF and BP is affected by these perturbations, effectively shifting the device transfer functions, which must be accounted for in simulations. In practice, however, the noise sources in physical systems are complex and cannot be fully quantified. Moreover, BP relies on global gradient descent, and its backward pass is typically executed off-chip, preventing the backward computation from faithfully reflecting the actual on-chip variability. This mismatch renders BP particularly sensitive to device nonidealities. In contrast, FF updates weights locally and layer by layer, directly based on the physical system's forward response, inherently accommodating hardware-induced fluctuations and thus exhibiting greater noise tolerance.

More specifically, the supervised FF network consists of an input layer with 784 neurons (corresponding to 28 $\times$ 28 images), two hidden layers with 1000 neurons each, and an output layer with 10 neurons. The supervised BP network has the same input and output sizes, but uses two hidden layers with 128 neurons each.

Under ideal conditions, the FF architecture achieves a training accuracy of 81.7% and a test accuracy of 82.18%. In contrast, the BP architecture achieves a training accuracy of 91.57% and a test accuracy of 91.35%. To emulate practical conditions, we introduce zero-mean Gaussian noise into the parameter β, i.e.

$$\beta' = \beta + \varepsilon, \varepsilon \sim \mathcal{N}\left(0, \sigma^2\right).$$

The standard deviation σ is set to 0, 0.5, 0.75, 1.0, and 1.25, where $\sigma = 0$ corresponds to the ideal baseline without perturbation. In the FF architecture, the training accuracies are 80.87%, 79.21%, 78.72%, and 78.4%, and the test accuracies are 80.98%, 79.28%, 78.66%, and 78.2%, respectively. For the BP architecture the training accuracies are 83.5%, 71.05%, 61.19%, and 43.98%, and the test accuracies are 84.13%, 72.23%, 62.74%, and 46.2%. As the noise variance increases, the accuracy of BP drops rapidly and its convergence becomes markedly slower, while FF shows only a modest degradation in accuracy and maintains a relatively fast convergence speed. Further degradation-rate analysis reveals that, the training accuracy degradation is only 0.83% to 3.3%, and the test accuracy degradation is 1.2% to 3.98%. By contrast, in the BP architecture, the training accuracy degradation reaches 8.07% to 47.59%, and the test accuracy degradation is 7.22% to 45.15% (see Fig. 1(h)-(j)).

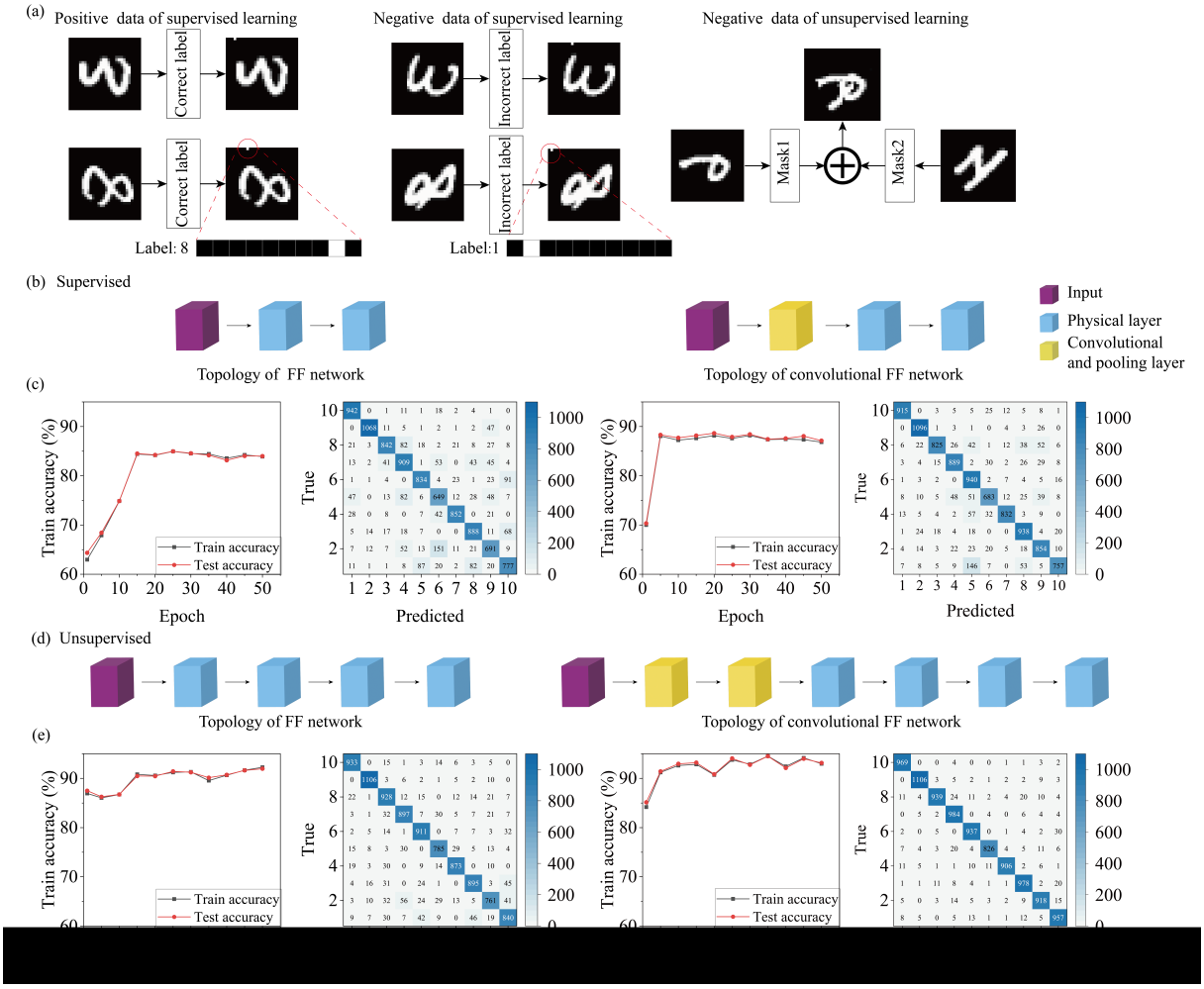


Figure D1 Performance demonstration of the physically trained FF neural network on the MNIST task. (a) Illustration of dataset sample generation. In supervised training, positives are original images with correct label embeddings and negatives use incorrect labels; in unsupervised training, positives are original images and negatives are obtained by masking and coupling the originals. (b) Schematic of the supervised PDNN topologies. The left panel shows the network topology with two forward-only layers, and the right panel shows a convolution–forward network consisting of one convolutional layer followed by two forward-only layers. Purple denotes signal inputs, and blue denotes the digital model of the physical system used for the forward-only layers. (c) Supervised training results. Classification accuracy on the training and test sets versus training epochs for the two network topologies on the MNIST classification task under supervised training. (d) Schematic of the unsupervised PDNN topologies. The left panel shows the network topology with four forward-only layers, and the right panel shows a convolution–forward network consisting of two convolutional layers followed by four forward-only layers. Purple denotes signal inputs, and blue denotes the digital model of the physical system used for the forward-only layers. (e) Unsupervised training results. Classification accuracy on the training and test sets versus training epochs for the two network topologies on the MNIST classification task under unsupervised training.

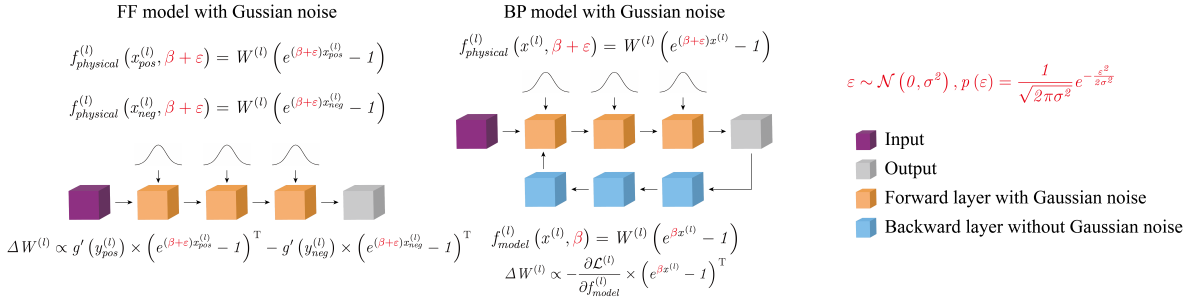


Figure E1 PDNN topologies used in the robustness experiments. The left panel shows the digital PDNN model based on the FF algorithm for learning on the MNIST handwritten digit dataset, and the right panel shows the digital PDNN model based on the BP algorithm for learning on the same dataset. Purple denotes signal inputs; orange denotes the digital model of the physical system under noise perturbations used for the forward-propagation layers; blue denotes the corresponding noisy digital model used for the backward-propagation layers; grey denotes the output signals.

The core of BP relies on the asymmetry between forward and backward passes, while the backward computation is carried out off chip. Its mathematical form is therefore difficult to align with the noise present in the physical system, so under noisy conditions the mismatch causes errors to accumulate and be amplified layer by layer, leading to severe performance degradation (see Fig. 1(f)-(g)). In contrast, FF is entirely based on local forward computation and discrimination; its update process directly leverages the nonlinear outputs of the physical system and is intrinsically compatible with noise. As a result, FF can maintain high classification performance and stable convergence even in perturbed environments. Taken together, these results indicate that FF exhibits significant robustness and stability in noisy scenarios and is better suited as a training strategy for physical neural networks.

References

1 Nam S, Kang D, Jo J W, et al. Recent progress of neuromorphic sensory and optoelectronic systems. *International Journal of Extreme Manufacturing*, 2025.
2 Sun L, Yu H, Wang D, et al. Selective growth of monolayer semiconductors for diverse synaptic junctions. *2D Materials*, 2018, 6(1): 015029.
3 Hinton G. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint*, 2022, arXiv:2212.13345.
4 Xue Z, Zhou T, Xu Z, et al. Fully forward mode training for optical neural networks. *Nature*, 2024, 632(8024): 280–286.
5 Huo Z, Sun Q, Yu J, et al. Neuromorphic devices assisted by machine learning algorithms. *International Journal of Extreme Manufacturing*, 2025, 7(4): 042007.
6 Lin Y, Chen X, Zhang Q, et al. Nano device fabrication for in-memory and in-sensor reservoir computing. *International Journal of Extreme Manufacturing*, 2024, 7(1): 012002.
7 Bian J, Liu Z, Tao Y, et al. Advances in memristor based artificial neuron fabrication-materials, models, and applications. *International Journal of Extreme Manufacturing*, 2023, 6(1): 012002.
8 Momeni A, Rahmani B, Malléjac M, et al. Backpropagation-free training of deep physical neural networks. *Science*, 2023, 382(6676): 1297–1303.