Mixed-Precision Continual Learning Based on Computational Resistance Random Access Memory

Yi Li, Woyu Zhang, Xiaoxin Xu, Yifan He, Danian Dong, Nanjia Jiang, Fei Wang, Zeyu Guo, Shaocong Wang, Chunmeng Dou, Yongpan Liu, Zhongrui Wang,* and Dashan Shang*

1. Introduction

With the rapid development of deep learning algorithms in recent years, artificial neural networks (ANNs) have obtained qualitative breakthroughs in various fields of artificial intelligence (AI), such as face recognition, voice recognition, natural language processing, robotics, and autonomous driving. However, when performing continual learning on sequential tasks, ANNs suffer from catastrophic forgetting. That is, when these networks are trained on new tasks, previously learned tasks are quickly forgotten (see Figure 1). To address this issue, several continual learning neural networks, e.g., the gradient episodic memory (GEM), the elastic weight consolidation (EWC), the continual learning through synaptic intelligence (SI), the general replay (GR), and the progressive neural networks (PNN) have been proposed. These models can effectively mitigate catastrophic forgetting and achieve outstanding performance in benchmarking continual learning with split-MNIST and split-FashionMNIST. Nonetheless, the implementations of the aforementioned models on conventional CMOS circuit systems face great challenges due to the complex network structures and enormous parameters. The frequent and massive data shuttling between memory and processing units incurs unaffordable computing costs, especially for resource-limited edge applications.

Artificial neural networks have acquired remarkable achievements in the field of artificial intelligence. However, it suffers from catastrophic forgetting when dealing with continual learning problems, i.e., the loss of previously learned knowledge upon learning new information. Although several continual learning algorithms have been proposed, it remains a challenge to implement these algorithms efficiently on conventional digital systems due to the physical separation between memory and processing units. Herein, a software–hardware codesigned in-memory computing paradigm is proposed, where a mixed-precision continual learning (MPCL) model is deployed on a hybrid analogue–digital hardware system equipped with resistance random access memory chip. Software-wise, the MPCL effectively alleviates catastrophic forgetting and circumvents the requirement for high-precision weights. Hardware-wise, the hybrid analogue–digital system takes advantage of the colocation of memory and processing units, greatly improving energy efficiency. By combining the MPCL with an in situ fine-tuning method, high classification accuracies of 94.9% and 95.3% (software baseline 97.0% and 97.7%) on the 5-split-MNIST and 5-split-FashionMNIST are achieved, respectively. The proposed system reduces ≈200 times energy consumption of the multiply-and-accumulation operations during the inference phase compared to the conventional digital systems. This work paves the way for future autonomous systems at the edge.
One promising approach is to perform continual learning using the in-memory computing (IMC) paradigm, where certain arithmetic operations are carried out by the memory itself. As a result, the IMC paradigm is characterized by superior parallelism and energy efficiency, especially when handling data-intensive tasks.\cite{16–18} Resistive nonvolatile memories (NVM), such as resistance random access memory (RRAM), phase-change memory, and magnetic random access memory, have been actively pursued for demonstrating the highly efficient IMC paradigm.\cite{19–21} However, due to device nonidealities, e.g., device variation, programming nonlinearity and asymmetry, and limited conductance states, the present IMC devices are still struggling to parallel the high-precision (e.g., 32-bit floating-point) calculations, which is critical for these continual learning neural network models.

Human brains excel at continual learning in a lifelong manner. The ability of brains to incrementally adapt is mediated by a rich set of neurophysiological processing principles that regulate the stability–plasticity balance of synapses. Synaptic plasticity is an essential feature of the brain that allows us to learn, remember, and forget.\cite{22} On the one hand, the synaptic weights need to be plastic enough to maintain the learning potential for new tasks; on the other hand, the weights need to remain stable to avoid being overwritten extensively during the training process. Metaplasticity, which refers to activity-dependent changes in neural functions that modulate subsequent synaptic plasticity,\cite{23} has been viewed as an important rule to balance the stability and plasticity of synapses.\cite{24} Recently, Laborieux et al. reported a binarized neural network (BNN) with metaplastic hidden weights to mitigate catastrophic forgetting, where the metaplasticity of a synapse was used as a criterion of importance concerning the tasks that have been learned throughout.\cite{25} This type of BNN not only embodies the potential benefits of the synergy between neurosciences and machine learning research but also provides a low-precision way to achieve continual learning with resistive IMC technology. However, the characterization and hardware implementation of BNN are still missing, and thus, the impact of device nonidealities and weight precision on the system performance remains to be elucidated.

Here, we propose a metaplasticity-inspired mixed-precision continual learning (MPCL) model for the hardware implementation of continual learning. The balance between plasticity and stability is regulated by the underlying sensitivity to data changes. The MPCL model is deployed on a hybrid analogue–digital computing platform equipped with a 256 kb RRAM IMC chip to perform 5-split-MNIST and 5-split-FashionMNIST continual learning tasks. By taking advantage of the IMC paradigm of the RRAM chip, both an \(\approx200\times\) reduction of the energy consumption for the multiply-and-accumulation (MAC) operations compared to traditional CMOS digital systems and high average recognition accuracies comparable to the state-of-the-art performance have been demonstrated during the inference phase, providing a promising solution to future autonomous edge AI systems.

2. Mixed-Precision Continual Learning Model

To reduce the precision requirement, we propose the MPCL neural network model with mixed-precision weights. Floating-point weights, which are precise to reflect minor weight changes associated with learning, are used as plastic memory for selectively forgetting unimportant information in subsequent task learning. In contrast, binary weights, which are less likely to respond to small weight changes, are used as the stable memory for storing important information. The entire training procedure is shown in Figure 2a. Inspired by metaplasticity, the MPCL adopts an asymmetric weight update strategy regulated by the memory coefficient \(m\) for the floating-point weights’ back-propagation during training and uses binary weights for task inference.

Figure 1. Schematics of the catastrophic forgetting. When ANNs perform multitask continual learning, the hidden weights are expected to be plastic to adapt to new tasks while remaining stable to avoid being overwritten and losing previous tasks’ information. This phenomenon is known as the plasticity–stability dilemma. The catastrophic forgetting is caused by failing to balance plasticity and stability, leading to a sharp accuracy decline on the previous cat task (the orange curve) while learning the present dog task (the green curve).
When training on a new task, if the floating-point weight updates in the same direction as its sign, the corresponding binary weight shall not change so that the MPCL will retain the high inference accuracy of the learned tasks. If the floating-point weight updates in the opposite direction as its sign, the corresponding binary weight may switch. In contrast, the rest of the floating-point weights sharing the same sign with their changes can freely update (see Experimental Section). The MPCL model increases only a small amount of training time compared to traditional BNNs due to the additional computations, while the inference time cost is the same (see Figure S4, Supporting Information). Owing to the asymmetric update strategy, the MPCL limits the switching of the binary weights that are less relevant with the new task to avoid massive weights overwriting, thus effectively mitigating catastrophic forgetting.

As an appropriate \( m \) is crucial for the MPCL to achieve high-performance multitask classification, we further investigate the strategy to refine \( m \). First, to analyze the correlation between the MPCL performance and \( m \), we randomly select two tasks out of the 5-split-FashionMNIST as the previous and current tasks. The definition of multitask average accuracy is given by

\[
\text{Acc}_{\text{aver}} = \frac{N_{\text{PT}} \times \text{Acc}_{\text{PT}} + N_{\text{CT}} \times \text{Acc}_{\text{CT}}}{N_{\text{PT}} + N_{\text{CT}}}
\]

where the \( \text{Acc}_{\text{PT}} \) and \( \text{Acc}_{\text{CT}} \) represent the accuracy of previous and current tasks, respectively, and the \( N_{\text{PT}} \) and \( N_{\text{CT}} \) represent the data size of the previous and current tasks, respectively. Figure 2b illustrates the impact of \( m \) on the binary weight switch rate and the average accuracy. The simulation result shows that, when \( m = 0.001 \), the binary weight switch rate exceeds 45% after...

---

**Figure 2.** a) Flowchart of the MPCL and weight update strategy. The MPCL model is a fully connected feed-forward neural network (hidden layer sizes are 64-256-10) with batch normalization and sign activation. The hidden layers use mixed-precision weights, where the binary weights are used in the forward-propagation and loss function evaluation, and the floating-point weights are used for back-propagation. b) The simulation results of the weight switch rate and two tasks average accuracy under different \( m \) values. After learning the current task, the weight switch rate is defined as the ratio of the switched binary weights. The best MPCL performance is obtained when the weight switch rate equals 3.48% and \( m = 0.005 \). c) Confusion matrices of the recognition accuracy for 5-split-FashionMNIST dataset with \( m = 0 \) and \( m = 0.005 \). d) The inference accuracy of the MPCL and other reported continual learning models on the 5-split-FashionMNIST dataset. We choose the GEM,[11] the EWC,[12] and the continual learning through SI[13] models as the comparison. e) The comparison results of noise immunity. The normal distribution noise is introduced to the hidden layer weights represented by the conductance of the RRAM cells. The “conductance noise” represents the size of the normal distribution noises introduced to the weights of hidden layers during simulation. Thanks to binary weights, the MPCL shows stronger robustness than the reported floating-point-type models, including GEM,[11] EWC,[12] and SI.[13]
learning the current task due to the less bounded update step size of the corresponding floating-point weight, resulting in a severe catastrophic forgetting on the previous task. With an increased $m$, the accuracy of the previous task gradually recovers due to the decreased rate of binary weight flipping. The best average accuracy of the two tasks appears $m = 0.005$ (see Figure S1, Supporting Information). Further increasing $m$, the weight switch rate dramatically decreases to 1%, revealing that the network loses the capability to learn the new task due to the lack of plastic weights. Figure 2c shows the confusion matrices of recognition accuracy by simulating the training with 5-split-FashionMNIST, clearly demonstrating the continual learning ability of the MPCL with an appropriate $m$ value.

We then compare the MPCL with other existing continual learning algorithms on the 5-split-FashionMNIST dataset. As baselines, the model freely updated on each new task is denoted as None, while the model trained on both current and previous tasks information is referred to as Joint. As shown in Figure 2d, the MPCL effectively mitigates the catastrophic forgetting and achieves comparable continual learning accuracy (average accuracy of five tasks is 97.7%) with respect to other reported continual learning models. Finally, to probe the robustness of MPCL, normal distribution noises are introduced to the weights of hidden layers (see Experimental Section) as a simulation of the impact of device variation on the system’s inference performance. As shown in Figure 2e, the MPCL is significantly less prone to noise than those floating-point-type models. Therefore, the MPCL model is a natural choice for hardware implementation of continual learning considering the evitable device nonidealities of resistive NVMs.

It should be pointed out that the MPCL model will still face catastrophic forgetting or loss of learning capability with overwhelming learning tasks. This challenge, according to Gido et al., applies to all existing continual learning models to scale with the number of learning tasks. A possible explanation for this phenomenon is that the hypothesis space for weight search gradually decreases with the increasing number of learning tasks, resulting in the inability to achieve the global optimal solution. This phenomenon can be mitigated by expanding the network architecture and increasing the network’s capacity, e.g., the progressive neural networks.

3. Hybrid Analogue–Digital Hardware System for Continual Learning

To leverage the IMC paradigm for continual learning, a hybrid analogue–digital hardware system has been developed to implement the MPCL neural network (Figure 3). The MPCL contains a three-layer fully connected feed-forward neural network with both binary and floating-point weights. During continual learning, the binary weights are used in forward-propagation and loss evaluation, while the floating-point weights are used for weight update in back-propagation and updating corresponding binary weights through a sign function (see Experimental Section).

![Figure 3](image-url) Figure 3. The hybrid analogue–digital computing system. The MPCL consists of a three-layer fully connected feed-forward neural network with binary (pink) and floating-point (blue) mixed-precision weights. By replacing the most computationally expensive floating-point VMM operations with binary ones, the MPCL significantly lowers the requirements for weight precision during the inference phase. The hybrid analogue–digital system consists of a 256 kb RRAM computing-in-memory chip, a general digital processor, and a PCB. During hardware implementations, the binary weights are physically represented by the normalized conductance of the RRAM differential pairs, and the floating-point operations are carried out by the general digital processor. The 5-split-FashionMNIST and 5-split-MNIST continual learning datasets are used to benchmark the continual learning performance.
The hybrid analogue–digital hardware system integrates an RRAM computing-in-memory chip and a general digital processor on a printed circuit board (PCB). The RRAM chip physically embodies binary weights to accelerate the low-precision binary vector–matrix multiplication (VMM) in the MPCL forward-propagation. The high-precision digital processor consists of a field-programmable gate array (FPGA) together with an advanced RISC machine (ARM) processor for both floating-point operations, e.g., normalization and activation, and commanding the RRAM chip. When deploying the MPCL to the hybrid analog–digital system, the RRAM array is programmed according to the pretrained binary weights, where each +1 or −1 binary weight is encoded by the RRAM differential pair (see Experimental Section and Supplementary Text1, Supporting Information). As the weights storage and processing takes place on the same RRAM chip, the energy and time consumption to transfer data between processor and memory can be minimized during the inference phase.

To validate the continual learning performance, we synthesize the 5-split-FashionMNIST and 5-split-MNIST datasets by evenly splitting the FashionMNIST and MNIST datasets into five tasks. Each task contains 12 000 training images and 2000 test images from two categories. During the inference phase, all images are first down-sampled and binarized before being converted into voltage signals that are fed into the RRAM chip through digital-to-analogue converters (DACs). The VMM results are in the form of accumulated currents that are sampled by analogue-to-digital converters (ADCs) and fetched by the general digital processor for the downstream normalization and activation (see Experimental Section).

4. In Situ Fine-Tuning Method

Although resistive IMC reduces the O(N²) computational complexity of VMM down to O(1) by exploiting Kirchoff’s law and Ohm’s law,[1] crucial to computationally intensive tasks, e.g., neural networks,[18] increasing hardware parallelism tends to accumulate errors rapidly and eventually be detrimental to hardware performance due to nonideal factors such as device noise and IR-Drop.[28,29] To reduce the impact of the hardware nonidealities, we design an in situ fine-tuning method that allows the hardware to accelerate the MPCL through massively parallel processing (MPP) without significantly compromising inference accuracy, as shown in Figure 4.

On the hardware side, we first map software weights to program RRAM differential pairs. These hardware weights are parallelized in forward-VMM operations by simultaneously applying read voltages to the RRAM chip’s several bit lines (BLs), and the parallelism is thus defined as the number of rows of these BLs. The modeled equivalent values of these hardware weights follow a quasi-Gaussian distribution due to RRAM device variation, which differs from the pretrained weights and results in deviations between hardware and software calculating results. Using the build-in quantization of ADC, we introduce an Int operation during the current summation, converting the fluctuating hardware weights to stable fixed-point weights. Although the Int operation lowers the precision of weights, it significantly reduces the randomness caused by the conductance fluctuation of RRAM during computation and also benefits computational cost.

On the software side, we develop the same computational flow as the hardware. To reconstruct the hardware’s Int operation, we

![Image of In Situ Fine-Tuning Method](https://www.advancedsciencenews.com)

**Figure 4.** In situ fine-tuning method. The in situ fine-tuning method is designed for optimizing hardware performance with parallel processing. Step 1: the equivalent hardware weights are read out from the programmed RRAM array. The \( W_{\text{para}} \) used in hardware VMM operations is obtained by applying read voltages to multiple BLs simultaneously on both positive and negative memristor differential pair arrays. Step 2: normalized weights that contain nonideal noises are converted into fixed-point weights by the Int operation through ADC quantization. The Int operation significantly reduces the noise caused by the conductance fluctuations of RRAM devices. Step 3: the equivalent hardware parallel weights are decomposed into two sparse matrices \( W^\varepsilon \) and \( W^\theta \) to repeat the hardware computational flow in the software simulation. By enforcing the Int operation to the VMM results of \( W^\varepsilon \) and input vector \( x \), the simulation results match with the hardware implementation. Step 4: the first two layers of the MPCL network are fixed, and the in situ fine-tuning method is used to retrain the last layer. Finally, the well-retrained weights are remapped to the RRAM array.
propose a matrix decomposition method, which decomposes the equivalent hardware weights into two sparse matrices. The left matrix $W^L$ is the equivalent hardware weight matrix divided according to the hardware computational parallelism used to simulate the parallel hardware operations. The right matrix $W^R$ is a zero-one matrix, which sums the intermediate results after weight decomposition, and restores the output matrix to the original dimension (see Figure S2, Supporting Information). The weight matrix decomposition avoids using the entire row or column during VMM. Meanwhile, the insertion of Int operations after the parallel multiply–accumulation further regulates the VMM results, converting the obtained results into the fixed-point type according to the hardware implementation. As the hardware computational flow is precisely reflected by the software one, we can obtain the matched parallel weights ($W^{\text{para}}$) in both software and hardware. The parallel weights are stable enough for the subsequent in situ fine-tuning, where the last layer of the MPCL neural network is further optimized with rest layers fixed before being mapped again to the RRAM array.

5. Hardware Implementation of MPCL

To benchmark the continual learning performance of the MPCL and the designed in situ fine-tuning method, we deploy the

---

Figure 5. a) The normalized conductance map (left) and resistance distribution of the RRAM array (right). The conductance map contains $220 \times 256$ RRAM differential pairs (56 300 devices on each positive and negative array) and is selectively formed by applying electric pulses (3 V, 10 μs) according to the predesigned patterns. The resistance of the RRAM array shows two quasi-Gaussian distributions, constituting the low-noise binary weights. b) Retention of the LRS and HRS. Inset shows the LRS fluctuation rate represented by the maximum conductance deviation (see Experimental Section), lower than 15%. c) Hardware implementation accuracies before and after in situ fine-tuning with increasing parallelism. d) The estimated MAC operations with increasing parallelism for single picture inference. e) Simulation and hardware-measured training accuracies on the 5-split-FashionMNIST and 5-split-MNIST datasets.
MPCL on the hybrid analogue-digital hardware system. Figure 5a illustrates the reliability of the binary weights on the RRAM chip. The normalized conductance map of the RRAM array reveals a weight mapping accuracy of 99% thanks to the high yield of the RRAM array. The narrow quasi-Gaussian distributions of the low resistance state (LRS) and high resistance state (HRS) improve the precision of the binary weights. Figure 5b demonstrates the conductance retention over 10,000 reading operations. The majority of RRAM devices show stable nonvolatile conductance in carrying IMC. It should be pointed out that the RRAM state before forming is defined as the HRS in this case, as it shows high energy efficiency due to the low conductance values. Figure 5c shows the relationship between the computational parallelism and the average inference accuracy of the hardware-implemented MPCL. The in situ fine-tuning significantly suppresses the error accumulation caused by the parallel hardware operations, achieving 95.3% accuracy on the 5-split-FashionMNIST dataset with a parallelism of 16. Thanks to the improvement of parallelism, the estimated number of VMM in a single image inference dramatically decreases to 3600 operations (OPs), revealing a 17 times improvement over the conventional hardware implementation (62,000 OPs), as shown in Figure 5d. Finally, Figure 5e shows the simulation and hardware-measured recognition accuracy of the MPCL on both the 5-split-MNIST and 5-split-FashionMNIST datasets. The average hardware-measured accuracies on the 5-split-MNIST and 5-split-FashionMNIST datasets are 94.9% and 95.3%, respectively, which are just 2.1% and 2.4% lower than the software baseline of 97.0% and 97.7%.

A comparison between the MPCL results and existing continual learning models is shown in Table 1. In software simulation, the MPCL shows a comparable accuracy with the reported continual learning models, which lift the strict requirement of the weight precision, making it friendly to resistive NVMs for IMC. In addition, taking advantage of the IMC paradigm of the RRAM computing-in-memory chip, the energy consumption for the MAC operations during the inference phase is improved $\approx 200 \times$ compared to the traditional CMOS digital systems. In the hardware implementation, the MPCL shows $\approx 40 \times$ improvement compared to the reported RRAM-based IMC continual learning system,\(^\text{(30)}\) where 4-bit weight precision is used for a hybrid network consisting of convolutional neural network (CNN) and spiking neural network (SNN).

### 6. Conclusion

In this work, we have developed a metaplasticity-inspired MPCL model experimentally implemented on a hybrid analogue-digital computing system to solve continual learning problems. The MPCL leverages the different precision of floating-point and binary weights to balance the plasticity and stability of synapse, which thus mitigates the catastrophic forgetting. The in situ fine-tuning method helps to further reduce the impact of RRAM device nonidealities on the parallel operation. Finally, using the hybrid analogue-digital hardware system, we achieve an average recognition accuracy of 94.9% and 95.3% (software baseline 97.0% and 97.7%) on the 5-split-MNIST and 5-split-FashionMNIST continual learning datasets, respectively. Relying on the IMC paradigm of the RRAM chip, the energy efficiency of the MAC operations is improved $\approx 200 \times$ compared to the traditional CMOS digital systems during the inference phase. The ability to introduce balanced synaptic plasticity and stability through mixed-precision weights makes the MPCL an ideal model to perform continual learning tasks in autonomous edge AI systems.

### 7. Experimental Section

**The RRAM Chip Fabrication:** The RRAM chip with a crossbar structure contains 256 kb RRAM cells (512 rows $\times$ 512 columns). Each of these cells is integrated on the 40 nm standard logic platform between metal 4 (M4) and metal 5 (M5), including top electrodes (TEs), a TaO$_x$-based oxide resistive layer, and bottom electrodes (BEs). The TE comprising 3 nm Ta and 40 nm TiN is deposited by sputtering in sequence. The resistive layer consisting of 10 nm TaN and 5 nm Ta is deposited by physical vapor deposition on the BE via, where the Ta is further oxidized in an oxygen atmosphere to form an 8 nm TaO$_x$ dielectric layer. The BE via with a size of 60 nm is patterned by photolithography and etching, where the via is filled with TaN using physical vapor deposition and then polished by chemical mechanical polishing (CMP). After fabrication, the logic BEOL metal (Ta) is deposited as the standard logic process, and the cells in the same columns share the BE via while those in the same rows share the BE to generate the RRAM array chip. Finally, the chip is heated at 400°C for 30 min to perform the postannealing process.

**The Hybrid Analogue–Digital Computing System:** This hybrid analogue–digital computing platform consists of three parts, the RRAM computing-in-memory chip, the general digital processor (Xilinx ZYNQ XC7Z020 system-on-chip), and the high-speed PCBs. The high-speed PCBs contain an 8-channel digital-to-analogue converter (DAC80508, TEXAS INSTRUMENTS, 16-bit resolution) with two 8-bit shift registers.

### Table 1. Comparison of this work with recent works.

| Approach       | EWC\(^\text{(11)}\) | GEM\(^\text{(11)}\) | SI\(^\text{(13)}\) | This work | CNN + SNN\(^\text{(20)}\) | This work |
|----------------|----------------|----------------|-------------|------------|----------------|-----------|
| Implementation | Simulation     | Simulation     | Simulation  | Simulation | Hardware       | Hardware  |
| Average accuracy (%)\(^\text{(a)}\) | 99.0 | 94.5 | 95.7 | 97.5 | 82.0 | 95.2 |
| Class no.      | 10             | 10             | 10          | 10         | 3              | 10        |
| Inference precision | FP             | FP             | FP          | 1 bit      | 4 bits         | 1 bit     |
| Noise tolerance (%)\(^\text{(b)}\) | 13              | 15             | 10          | 36         | N/A            | N/A       |
| MAC energy (J)\(^\text{(c)}\) | $5.4 \times 10^{-7}$ | $5.4 \times 10^{-7}$ | $5.4 \times 10^{-7}$ | $2.7 \times 10^{-6}$ | $\approx 10^{-7}$ | $2.3 \times 10^{-9}$ |

\(^{a}\)The average accuracy is obtained on the 5-split-MNIST under the same network architecture 192-256-256-10; \(^{b}\)The noise tolerance is calculated when the model’s inference accuracy drops to 90%; \(^{c}\)The MAC energy refers to the energy consumption of the MAC operations for a single picture inference. The simulation MAC energy is estimated based on 45 nm standard CMOS technology, while that of the hardware implementation is estimated based on memristor units.
(SN74HC595, TEXAS INSTRUMENTS), providing 64-ways parallel analogue input voltages ranging from 0 V to 5 V. During VMM calculations, the input vectors are converted into the DC voltage signals and applied to the BSs of the RRAM chip through a 4-channel analogue multiplexer (CD4051B, TEXAS INSTRUMENTS). The calculation results are represented by the source line’s (SL) convergence currents and converted into voltage outputs through trans-impedance amplifiers (OPA4322-Q1, TEXAS INSTRUMENTS). Finally, the voltage outputs are read by ADCs (ADS8324, TEXAS INSTRUMENTS, 14-bit resolution) and sent to the general digital processor for the following calculations.

Details of the Algorithms
Algorithm 1: The MPCL model. The MPCL is a three-layer fully connected feed-forward neural network using binary weight \( W_b \) and floating-point weight \( W_f \) as a set of mixed-precision weights. During the training phase, the binary weights used for forward-propagation are obtained by exerting the sign function to the corresponding floating-point weight.

\[
W_b = \text{Sign}(W_f) \tag{2}
\]

After calculating the entropy loss, the MPCL uses the momentum-based Adam optimizer\(^{[31]} \) where the \( \Delta W_f \) is the update of floating-point weight. The asymmetric weight update is determined by the sign of the product of \( \Delta W_f \) and \( W_f \). If \( \Delta W_f \cdot W_f > 0 \), the floating-point weight is updated in the opposite direction to its sign, and the maximum changes will be limited during each update step.

\[
W_{f,\text{Max}} = \frac{|W_f|}{\text{Update\_step}} \tag{3}
\]

The \( W_{f,\text{Max}} \) is further regulated by the memory coefficient \( m \) to obtain the maximum allowed update during each update.

\[
W_{f,\text{Max\_allowed}} = \frac{W_{f,\text{Max}}}{m} \tag{4}
\]

To avoid massive binary weights switching, the floating-point weights are updated using the smaller of \( |\Delta W_f| \) and \( |W_{f,\text{Max\_allowed}}| \).

\[
W_f = W_f - \text{Sign}(\Delta W_f) \times \min(|W_{f,\text{Max\_allowed}}|, |\Delta W_f|) \tag{5}
\]

where the \( \eta \) is the learning rate. If \( \Delta W_f \cdot W_f \leq 0 \), the floating-point weight updates in the same direction of its sign, and the floating-point weight will be freely updated.

\[
W_f = W_f - \eta \Delta W_f \tag{6}
\]

Finally, the number of update steps will be minus one to end one update process.

\[
\text{Update\_step} = \text{Update\_step} - 1 \tag{7}
\]

If the model’s inference accuracy does not increase within five epochs, the training phase will be early stopped to avoid over-fitting.

Algorithm 2: The noise injection. To probe the robustness of the MPCL compared with the reported floating-point-type models, we simulate the impact of device variation during inference by introducing the normal distribution noise to the hidden layer weights \( W_{f,\text{noise}} \) to obtain the nonideal weight \( W_{f,\text{with\_noise}} \). Both the \( W_{i,j} \) and \( W_{f,\text{with\_noise}} \) follow a normal distribution.

\[
(W_{i,j}, W_{f,\text{with\_noise}}) \sim N(W_{i,j}, (W_{i,j} \times \rho)^2) \tag{8}
\]

The mean and standard deviation of the normal distribution are \( W_{i,j} \) and \( W_{i,j} \times \rho \), respectively, and the \( \rho \) is defined as the conductance noise \%, which determines the size of conductance noise during reading operations, ranging from 0% to 100%. After noise introduction, all continual learning models are tested on the 5-split-FashionMNIST.

Algorithm 3: The maximum conductance deviation. The maximum conductance deviation is used to illustrate the memristor’s conductance fluctuation during reading operations and is defined as

\[
\text{Deviation} = \frac{\text{MAX}(|g_{\text{max}}, g_{\text{ave}}, g_{\text{min}}|)}{g_{\text{ave}}} \tag{9}
\]

where the \( g_{\text{max}}, g_{\text{ave}}, \) and \( g_{\text{min}} \) are the max, min, and average conductance of the memristors during reading operations.

The Method of Estimating Power Consumption: To estimate the energy consumption of MAC operations in the inference of a single picture, we calculate the energy cost of RRAM cells with binary weights. During inference, as the computational parallelism of the RRAM chip is smaller than the horizontal dimension of weight matrices, some of the summation operations need to be implemented on the ARM core. The method of estimating power consumption is given by

\[
\text{MAC\_Energy} = \text{read\_energy\_RRAM} \times \text{mult\_num} + \text{plus\_energy\_ARM} \times \text{plus\_num} \tag{10}
\]

where the mult_num and plus_num are the number of multiplication and plus operations, respectively. According to the weight matrices and computational parallelism, the number of multiplication and summation operations is calculated according to

\[
\text{mult\_num} = \sum_i \text{row}(W_i) \times \text{column}(W_i) \tag{11}
\]

\[
\text{plus\_num} = \sum_i \text{row}(W_i) \times (\text{column}(W_i) \times \text{parallelism} - 1) \tag{12}
\]

where the layer_num is the number of the hidden layers. The MAC operations performed on the RRAM chip are then carried out by the read operations of memristor cells, and the read_energy_RRAM is given by

\[
\text{read\_energy\_RRAM} = \text{read\_voltage}^2 \times \left( \frac{R_{\text{LRS}} \times R_{\text{HRS}}}{R_{\text{LRS}} + R_{\text{HRS}}} \right) \times \text{read\_time} \tag{13}
\]

where the read_voltage and read_time are the amplitude of the reading pulse and the memristor’s read-response time (see Figure S3, Supporting Information). The \( R_{\text{LRS}} \) and \( R_{\text{HRS}} \) are the average resistance of LRS and HRS, respectively. All the parameters used for the evaluation of energy consumption are summarized in Table S1. Supporting Information. Note that a few of the RRAM devices (<1%) will be reset to about 135 kΩ as the value of \( R_{\text{HRS}} \) during in situ fine-tuning, which has a negligible effect on the energy estimation. The plus_energy_ARM refers to the energy consumption of the summation operations performed on the ARM core and is calculated based on the 45 nm standard CMOS technology. By taking advantage of the IMC paradigm, the energy consumption for the MAC operations during inference is improved ~200× compared to the 45 nm ARM digital systems. As MAC operations account for the majority of the energy consumption in the forward-propagation, the RRAM computing in-memory chip significantly improves the energy efficiency of the MPCL model during inference.

Supporting Information
Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements
This work was supported by the National High Technology Research Development Program under grant no. 2018YFA0701500, the National...
Conflict of Interest
The authors declare no conflict of interest.

Data Availability Statement
The data that support the findings of this study are available from the corresponding author upon reasonable request.

Keywords
continual learning, in-memory computing, mixed precision, resistance random access memory

Received: January 29, 2022
Revised: March 24, 2022
Published online: May 17, 2022

[1] P. Yao, H. Wu, B. Gao, S. B. Eryilmaz, X. Huang, W. Zhang, Q. Zhang, N. Deng, L. Shi, H.-S. P. Wong, Nat. Commun. 2017, 8, 1.
[2] A. Andics, J. M. McQueen, K. M. Petersson, V. Gál, G. Rudas, Z. Vidnyánszky, Neuroimage 2010, 52, 1528.
[3] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, arXiv preprint arXiv:1810.04805, 2018.
[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Nature 2017, 550, 354.
[5] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scroffier, K. Kavukcuoglu, U. Muller, Y. LeCun, J. Field Rob. 2009, 26, 120.
[6] R. M. French, Trends Cognit. Sci. 1999, 3, 128.
[7] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, B.-T. Zhang, arXiv preprint arXiv:1703.08475, 2017.
[8] R. Kernker, M. McClure, A. Abitino, T. Hayes, C. Kan, presented at AAAI Conf. on Artificial Intelligence, New Orleans, LA, USA, April 2018.
[9] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, Y. Bengio, arXiv preprint arXiv:1312.6211, 2013.
[10] A. Robins, Connect. Sci. 1995, 7, 123.
[11] D. Lopez-Paz, M. A. Ranzato, Adv. Neural Inf. Process. Syst. 2017, 30, 6467.
[12] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, Proc. Natl. Acad. Sci. 2017, 114, 3521.
[13] F. Zhenke, B. Poole, S. Ganguli, presented at Int. Conf. on Machine Learning, Stockholm, SE, July 2017.
[14] H. Shin, J. K. Lee, J. Kim, arXiv preprint arXiv:1705.08690, 2017.
[15] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, arXiv preprint arXiv:1606.04671, 2016.
[16] A. Sebastian, M. Le Gallo, R. Khaddar-Aljameh, E. Leftheriou, Nanotechnol. 2020, 15, 529.
[17] M. M. Waldrop, Nat. News 2016, 530, 144.
[18] R. S. Williams, Eng., Comput. Sci. Eng. 2017, 19, 7.
[19] Z. Wang, C. Li, W. Song, M. Rao, D. Belkin, Y. Li, P. Yan, H. Jiang, P. Lin, M. Hu, Nat. Electron. 2019, 2, 115.
[20] K. Ding, J. Wang, Y. Zhou, H. Tian, L. Lu, R. Mazzarello, C. Jia, W. Zhang, F. Rao, E. Ma, Science 2019, 366, 210.
[21] Z. Luo, A. Hrabec, T. P. Dao, G. Sala, S. Finizio, J. Feng, S. Mayr, J. Raabe, P. Gambardella, L. Heyderman, Nature 2020, 579, 214.
[22] L. F. Abbott, M. A. Nelson, Nat. Neurosci. 2000, 3, 1178.
[23] F. Muller-Dahlhaus, U. Ziemann, Neuroscientist 2015, 21, 185.
[24] W. C. Abraham, Nat. Rev. Neurosci. 2008, 9, 387.
[25] A. Laborieux, M. Ernoult, T. Hirtzlin, D. Querlioz, Nat. Commun. 2021, 12, 1.
[26] G. Ven, H. T. Siegelmann, A. S. Tolias, Nat. Commun. 2020, 11, 1.
[27] Y. Wang, Q. Yao, J. T. Kwok, L. M. Ni, ACM Comput. Surv. 2020, 53, 1.
[28] A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, arXiv preprint arXiv:1606.04671, 2016.
[29] R. Saleh, S. Z. Hussain, S. Rochel, D. Overhauser, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 2000, 19, 635.
[30] I. Murhio-Martin, S. Bianchi, E. Covi, G. Piccolboni, A. Bricalli, A. Regev, J. Nodin, E. Nowak, G. Mulas, D. Ielmini, presented at IEEE Symp. on VLSI Technology, Montgomery, USA, July 2020.
[31] D. P. Kingma, J. Ba, arXiv preprint arXiv:1412.6980, 2014.