Efficient Training Algorithms for Neural Networks Based on Memristive Crossbar Circuits

Irina Kataeva
Advanced Research Division
DENSO CORPORATION
Komenoki-cho, Nisshin, Japan, 470-0111
Email: irina_kataeva@denso.co.jp

Farnood Merrikh-Bayat, Elham Zamanidoost and Dmitri Strukov
Electrical and Computer Engineering Department
University of California Santa Barbara
CA 93106-9560, USA
E-mail: {farnoodmb, elham, strukov}@ece.ucsb.edu

***Abstract*—We have adapted backpropagation algorithm for training multilayer perceptron classifier implemented with memristive crossbar circuits. The proposed training approach takes into account switching dynamics of a particular, though very typical, type of memristive devices and weight update restrictions imposed by crossbar topology. The simulation results show that for crossbar-based multilayer perceptron with one hidden layer of 300 neurons misclassification rate on MNIST benchmark could be as low as 1.47% and 4.06% for batch and stochastic algorithms, respectively, which is comparable to the best reported results for similar neural networks.**

I. Introduction

The field of artificial neural networks is experiencing yet another renascence with more attention than ever directed to the development of specialized hardware, which would be essential to utilize neural network's potentials for low-power and high-speed information processing [1], [2]. The majority of such efforts rely on conventional technology [3]–[6], e.g. complimentary metal-oxide-semiconductor (CMOS) circuits to implement artificial neurons and dynamic random access memory [7], static random access memory [8] or non-volatile floating memory gate memory [9], [10] to implement artificial synapses. Emerging memory device technologies [11], while not yet mature for large scale implementations, could offer further improvements in artificial neural network performance in the future [12]. One of the most promising proposals utilizing emerging memory technologies is hybrid CMOS/nanocrossbar circuits [2] with integrated two- or three-terminal resistive switching (memristive) devices [13], [14], which can be implemented with phase change memories [15], magnetic tunnel junctions [16], ferroelectric memories [17], metal oxide [18] or solid state electrolyte resistive switching devices [19], [20].

Given rapid advances in memristor technology [21] and, in particular, very encouraging experimental results for pattern classification with single layer perceptron [18], [22], the motivation for this work is to evaluate classification performance of large-scale networks implemented with hybrid CMOS/memristive crossbar circuits. Our specific focus is on multilayer perceptron (MLP) networks trained with backpropagation algorithm [23], [24], partially because it is a natural extension of the recent experimental work [18], [22] and also due to its close resemblance to the best pattern classifiers at the moment - deep convolutional neural networks [25]. Though a number of similar studies have been reported already [26]–[33], the novelty of this work and our specific contribution are as follows:

- We propose crossbar circuit compatible training approach that takes into account switching dynamics of metal oxide memristive devices. Simulation results for novel training approach show that classification fidelity on MNIST benchmark is comparable to the state-of-the-art results for similar-size MLP networks [23], and significantly better than that of earlier reported hybrid CMOS/crossbar circuits [29], [31].
- Simulations are carried out with accurate model of specific memristive devices [34], [35] that have been recently utilized in successful experimental demonstration of small-scale pattern classifier [18]. This differentiates our work from those based on crude models and also gives us confidence that our approach is practical.

The rest of the paper is organized as follows: Section II describes model assumptions for considered artificial neural network, including model of memristive device, implementation details of neural network, and simulated benchmark. The major contribution of the paper - the novel training approach is presented in Section III. Simulation results for classification performance and defects tolerance are summarized in Section IV, while detailed discussion of these results and concluding remarks are presented in the last two sections.

II. Model Assumptions

A. *Pt/TiO$_{2-x}$/Pt Memristive Devices*

Analog switching, good retention, and excellent scaling prospects make metal oxide memristive devices very attractive candidates for implementing artificial synapses [14]. In this work, we considered specific high-yield Pt/TiO$_{2-x}$/Pt memristive devices [18], for which we have recently developed accurate SPICE-like model [34], [35]. For these devices, conductance can be changed reversibly and continuously, e.g. increased by applying relatively high negative voltage $V_{SET} \leq V_{SET}{}^{TH}$ (SET process) and decreased (RESET) by applying relatively high positive voltage $V_{RESET} \geq V_{RESET}{}^{TH}$, where $V_{SET}{}^{TH}$ and $V_{RESET}{}^{TH}$ are corresponding effective threshold voltages (Fig. 1a). Due to highly nonlinear switching kinetics, device conductance can be sensed (i.e. read) at smaller biases

978-1-4799-1959-8/15/$31.00

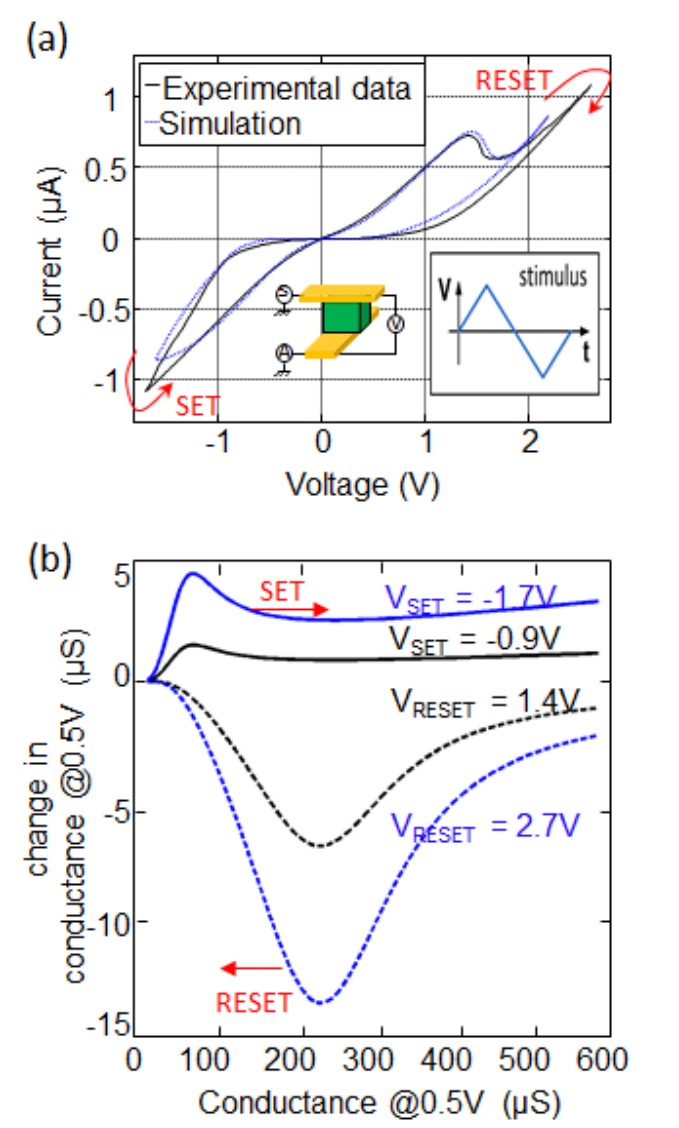

Fig. 1. Pt/TiO_{2-x}/Pt memristive device: (a) Typical switching *I-V* [34], [35] and (b) simulated change in device conductance as a function of device conductance and applied voltage stimuli ($\tau_{SET} = \tau_{RESET} = 10\mu s$ write pulses).

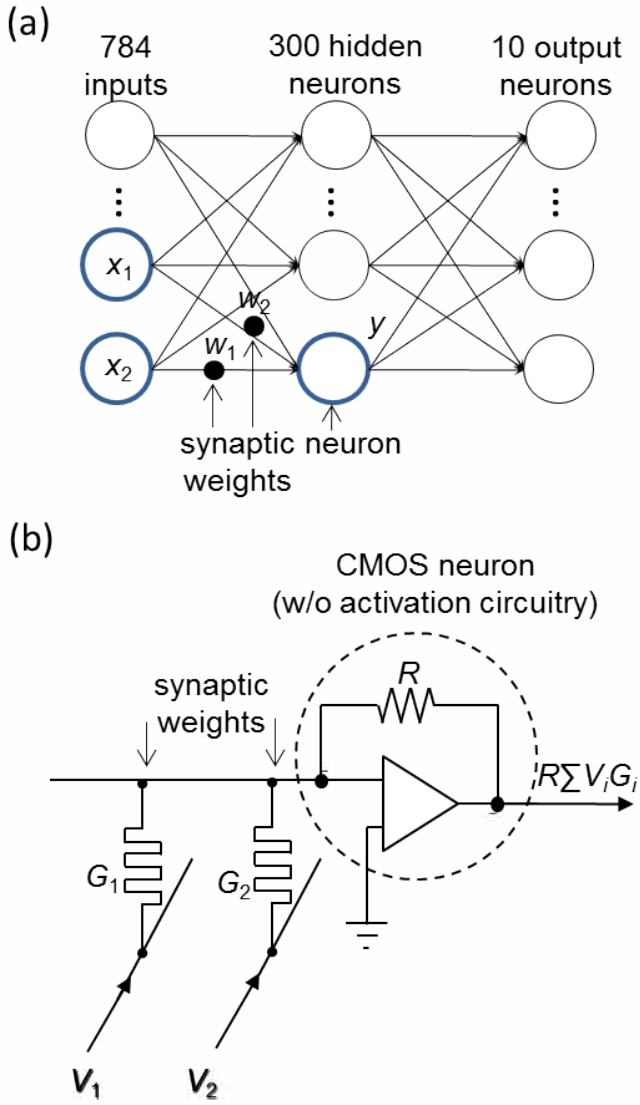

Fig. 2. Hybrid circuit ANN implementation: (a) Abstracted graph representation of one-hidden-layer MLP and (b) equivalent electrical circuit for a portion of a graph.

$|V| \leq V_{READ}{}^{TH} \approx 0.5V$ without disturbing the state of the device [18]. In general, the effective switching threshold voltages depend on duration of write stimulus. For the considered pulse durations for SET and RESET switching ($\tau_{SET} = \tau_{RESET} \approx 10\mu s$) we assume $V_{SET}{}^{TH} \approx -0.9V$ and $V_{RESET}{}^{TH} \approx 1.4V$.

Fig. 1b shows simulation results for the change in device conductance ΔG upon application of write voltages [34], [35]. Note that the change in conductance strongly depends on the current state G and different for SET and RESET switching.

B. Neural Network based on Hybrid Circuits

In typical hybrid circuit based neural networks [2], memristive devices are integrated into crossbar circuits to implement density-critical analog weights (i.e. synapses) and combined with CMOS circuitry, which implements neuron functionality and other peripheral functions (Fig. 2). In the simplest case, neuron output x is encoded by voltages $V \equiv x$, while synaptic weight w - by memristive device conductance $G \equiv w$. With virtually grounded neuron's input, V is multiplied by corresponding conductance G, so that current $I = GV$ is injected to the common wire as a result of Ohm's law (Fig. 2b). Individual currents are then summed up on a common wire according to Kirchhoff's law, and the total current is converted to a voltage, e.g. using operational amplifier. Therefore, such hybrid circuit enables compact analog implementation of dot-product operation $\sum G_i V_i$, which is a typical bottleneck operation in ANNs [1], [2].

To encode negative values with strictly positive conductances, one approach is to implement each weight as a differential pair of memristive devices $G \equiv G^+ - G^-$ [18], [22], [26]. In such differential scheme, currents corresponding to G^+ and G^- weights are summed separately as shown in Fig. 2b and then subtracted with additional CMOS circuitry (such as differential voltage amplifier, which could also perform

activation function). As a result, each neuron of hidden and output layers implements $f(\sum(G_i^+ - G_i^-)V_i)$, where f is a specific activation function.

C. Case Study: Multilayer Perceptron

Multilayer perceptron is one of the simplest artificial neural networks with competitive MNIST benchmark classification performance [23], [24] that easily maps to hybrid circuits [2] (Fig. 2). For this study we considered a multilayer perceptron with one hidden layer of 300 neurons, the smallest MLP previously published for MNIST benchmark [23]. It has 784 (28×28) inputs, one hidden layer with 300 neurons, and 10 outputs, so that the total number of memristive devices accounting for differential representation and bias weights is 2×(784+1)×300 + 2×(300+1)×10 = 477,020. The activation function is hyperbolic tangent

$$f = {V_{\text{READ}}}^{\text{TH}} \tanh[\beta \Sigma (G_i^+ - G_i^-) V_i] \quad (1)$$

with constant β = 3.4 mA^{-1} chosen according to [24].

The device model predicts that conductance will not decrease below $G_{\text{MIN}} \approx 14\mu$S. It would be also natural to expect that there is certain maximum conductance for memristive devices but such value is outside of the modeled range of conductances [34], [35]. For simplicity, in this paper we assume G_{MAX} = 590μS. According to our simulations, the performance results are insensitive to the value of G_{MAX} as long as G_{MAX} is sufficiently far away from peak values for conductance change for RESET process (Fig. 1b). With such G_{MAX} and G_{MIN} values, the weight range for considered differential implementation is [-576μS, +576μS].

MNIST training set is split into two parts - one with 50,000 images is used for training, and another with 10,000 images for validation. Pixel intensity of grey scale images is linearly mapped to [$-{V_{\text{READ}}}^{\text{TH}}$, $+{V_{\text{READ}}}^{\text{TH}}$] range.

III. Training Approach

Training of the considered MLP comes with challenges specific to memristive devices and crossbar circuits. Firstly, programming memristor-based weights precisely (i.e. with the same precision as in software simulations) is hardly practical. Secondly, to ensure scalability of the training approach, weights must be programmed in practical amount of time. The specific focus of this paper is exactly on these two issues. To this end, we have considered several different variations of backpropagation algorithm [23], [24] and its implementation strategies in memristive crossbar circuits with 5 different combinations in total and compared with a regular software implementation of the same MLP (Table I).

A. Batch vs. stochastic backpropagation training

At the algorithmic level, the considered training algorithms are classified as batch (sometimes called offline) and stochastic (online) backpropagation [24]. For batch training, training set is first partitioned into subsets (batches) of patterns. Patterns from a batch are applied one by one to the input of the network. After application of the whole batch, the weights are modified by

$$\Delta w = \eta \sum_{n \in P} e(n)x(n), \quad (2)$$

where η is a learning rate, x is neuron output from the previous layer (input or hidden) and e is an error backpropagated from the next layer (hidden or output).

In the case of one hidden layer perceptron, the error for the output layer is given by

$$e_{output} = f'_{output} \cdot (y_{desired} - y), \quad (3)$$

where y_{desired} is the desired output of the output layer neurons, y is their actual output and f'_{output} is the derivative of the activation function for the output layer neurons. For the input-to-hidden layer weights the backpropagated error is given by

$$e = f'_{\text{hidden}} \cdot (w_{\text{hidden-to-output}} \times e_{\text{output}}), \quad (4)$$

where e_{output} is the error of the output layer, $w_{\text{hidden-to-output}}$ are the weights between hidden and output layers and f'_{hidden} is the derivative of the activation function for the hidden layer neurons. The detailed derivation of backpropagation training rule and calculation of error e can be found in [23], [24], [36]. Note that the error backpropagation (4) can be implemented in the hardware using the same memristive crossbars and in the same way as described for pattern feedforward propagation in Section II.

If there is only one pattern in a batch, the training algorithm is called stochastic with weights updated after each pattern application. Note that in this case both x and e are peripheral to the crossbar array for each weight update, which simplifies implementation of the training procedure in hardware.

B. Ex-situ vs. in-situ training

At the hardware implementation level, we considered ex-situ and in-situ training [18].

In ex-situ training, the weights are calculated in a software-mirror network and then imported into the hardware by programming individual memristive devices, e.g. using feedback tuning algorithm [37]. The advantage is that because tuning circuitry implementation does not depend on software-based training algorithm, any complex training algorithm can be considered for ex-situ approach. The disadvantage is, however, that the weights are imported with a cruder precision as compared to software simulations and, furthermore, fabrication defects, such as stuck-on-close and stuck-on-open devices, can make it impossible to import the desired weights.

The alternative, in-situ approach relies on implementing weight adjustment directly in the hardware during training. Obviously, using tuning algorithm to update each weight would be too time consuming and hence not practical for in-situ approach. Instead, each weight is assumed to be modified with a single voltage pulse. More specifically, two schemes with *variable-amplitude* and *fixed-amplitude* voltage pulse implementation of training rule are proposed.

In a variable-amplitude scheme, the idea is to change device conductance proportionally to the weight update prescribed by the training rule (2) by applying a single voltage pulse with appropriate amplitude. This is achieved by applying logarithmically scaled voltages to the crossbar lines. For example, for stochastic algorithm for the case $x > 0$ and $e > 0$ (or $x < 0$ and $e < 0$), using

$$\begin{aligned} V_X &\propto log[x] \\ V_E &\propto -log[e] \end{aligned} \quad (5)$$

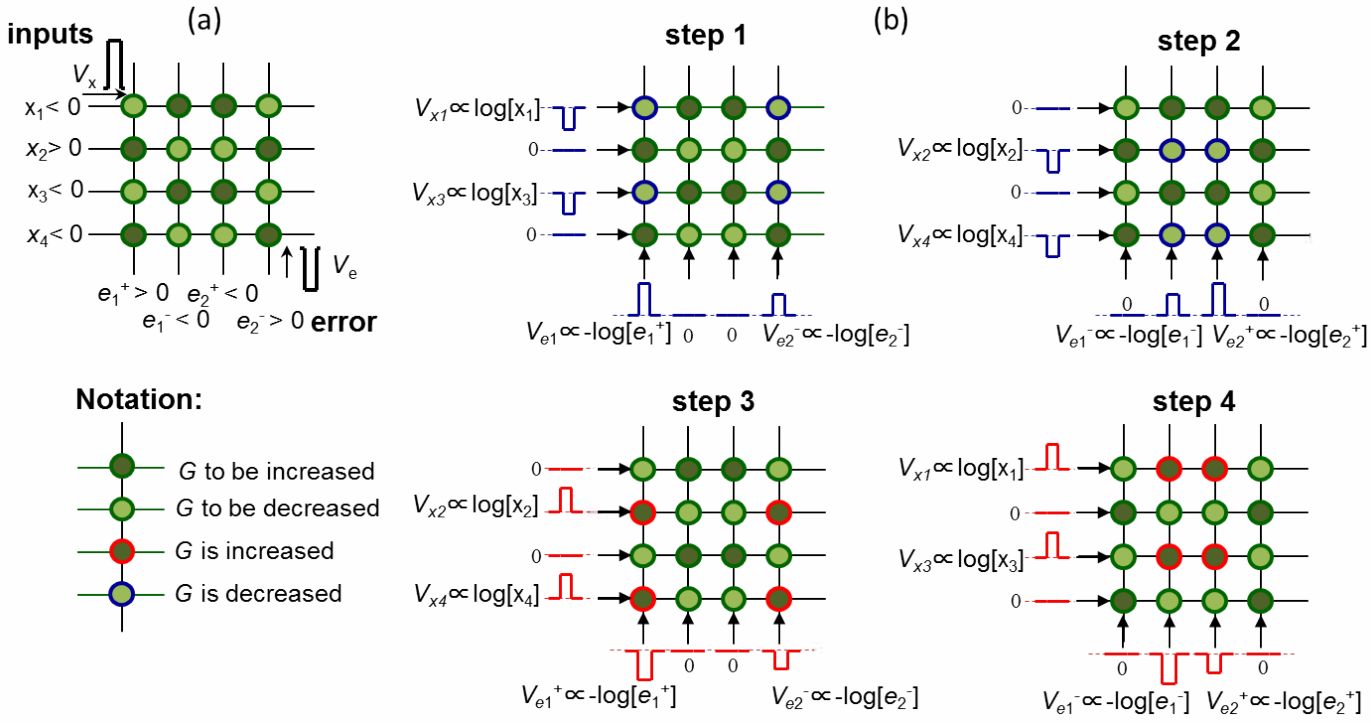

Fig. 3. Stochastic algorithm training for a single layer with 4 input and 2 differential output neurons mapped to memristive crossbar circuit: (a) Considered example and (b) corresponding 4-step weight update: step 1 - RESET pulses applied to modify devices for inputs $x < 0$ and error $e > 0$ (blue); step 2 - RESET pulses applied to modify devices for inputs $x > 0$ and error $e < 0$ (blue); step 3 - SET pulses applied to modify devices for inputs $x > 0$ and error $e > 0$ (red); step 4 - SET pulses applied to modify devices for inputs $x < 0$ and error $e < 0$ (red).

results in applying

$$V_X + V_E \propto log[xe] \tag{6}$$

full bias across a selected device. Because both RESET and SET switching rates are approximately exponential with respect to the applied bias across the device [34], [35], such scheme effectively results in update

$$\Delta G \propto exp[V] \propto xe \tag{7}$$

as specified by training rule (2). The pulse duration is used to control the learning rate η. The sign of V_X and V_E should be flipped for other cases of the signs of e and x (Fig. 3). Note that in order not to disturb half-selected devices [38], V_X and V_E have to be below the corresponding effective switching thresholds, while $V_X + V_E$ have to be larger than the threshold. Such variable-amplitude mapping allows fully parallel weight update for stochastic algorithm, e.g. using four-step technique described in [18] (Fig. 3).

A similar variable-amplitude voltage mapping scheme is assumed for batch in-situ algorithm, though in this case the weight updates (2), are no longer correlated with specific peripheral e and x values so that crossbar devices are updated line-by-line. In this case, e.g., for $\Delta w > 0$,

$$\begin{aligned} V_X &\propto log[k] \\ V_E &\propto -log[\Delta w/k] \end{aligned} \tag{8}$$

so that

$$\Delta G \propto \Delta w, \tag{9}$$

where k is a specific constant chosen to minimize half-selection disturbance. A major drawback of batch training in comparison with stochastic is that it requires to update crossbar serially line-by-line and, therefore, it might be potentially slower for large crossbars. Even more importantly, it requires extra hardware to calculate weight updates (2) and store intermediate values during training.

Finally, we have also considered a much simpler fixed-amplitude scheme in which a single voltage pulse of a fixed amplitude is applied to change the device conductance in appropriate direction (but hardly proportionally to the amplitude prescribed by the training rule). In this case, only sign information from the training rule (2) is utilized. This is somewhat similar to the Manhattan update rule [39] with the difference that in the considered implementation, conductance update also depends on current conductance state of a device (Fig. 1b).

IV. Simulation Results

A. Ex-situ training

A regular software-implemented MLP (i.e. without considering the device model and memristive crossbar hardware) was trained using stochastic gradient descent with a learning rate η=0.001 for 150 epochs and the weights that achieved the best misclassification rate of 1.57% were used for ex-situ training (Table I). The ex-situ training was simulated by sampling each value from the corresponding normal distribution, which were centered around the values for the trained weights, with the standard deviation (σ) equal to 2% of device conductance

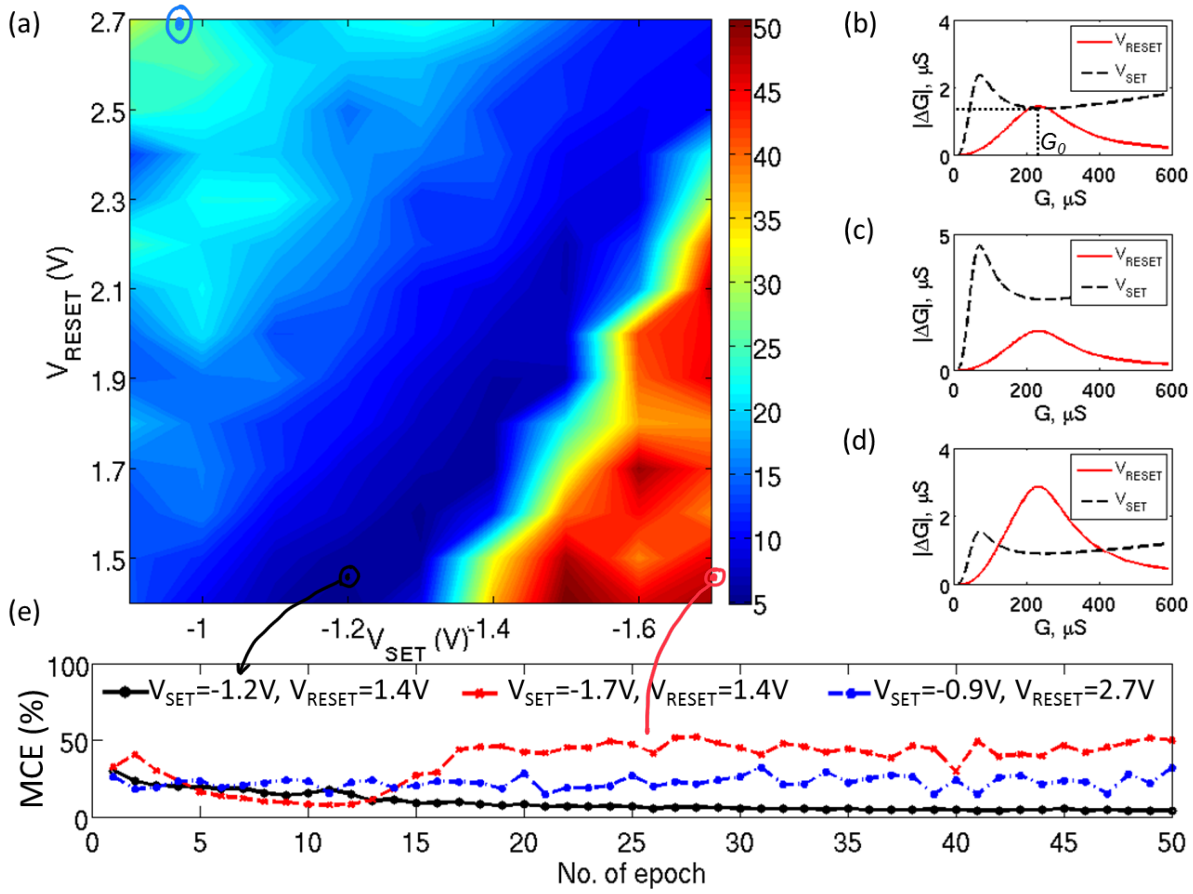

Fig. 4. Classification performance for fixed-amplitude batch in-situ training: (a) misclassification rate (%) for considered range of voltages with τ_{RESET} = 2μs, batch size = 1000 patterns, and 50 epochs of training, (b, c, d) switching dynamics for three representative pairs of SET/RESET voltages, and (e) their corresponding convergence plots. In particular, panel (b) shows switching dynamics for the case V_{SET} = -1.2V, V_{RESET} =1.4V, panel (c) for V_{SET} = -1.7V, V_{RESET} =1.4V, and panel (d) for V_{SET} = -0.9V, V_{RESET} =2.7V.

value. Such σ seems reasonable in light of recently reported experimental results [18], [37]. The best and average performance results for 20 ex-situ training experiments are presented in Table I. Note that according to our simulations, a more precise import does not add much to the performance.

B. *In-situ fixed-amplitude training*

For in-situ training the device conductances were randomly initialized from a normal distribution with mean of 100μS and standard deviation of 10μS, which was confirmed as an optimum choice for both stochastic and batch algorithms.

To utilize as large range of voltages as possible for variable-amplitude training, we assume that SET voltages are in a range [V_{SET}^{TH}, $2V_{SET}^{TH}$] and RESET voltages are in a range [V_{RESET}^{TH}, $2V_{RESET}^{TH}$]. Such voltages can be applied without causing much disturbance to half-selected devices.

TABLE I. BEST(AVERAGE) MISCLASSIFICATION ERROR (%) FOR MNIST BENCHMARK.

	in-situ		**ex-situ**	**regular software implementation**
	fixed-amplitude	variable-amplitude	σ=2% weight import	
batch (500)	1.98 (2.06)	1.47 (1.62)	-	-
stochastic	19.26 (20.16)	4.06 (4.31)	1.54 (1.62)	1.57 (1.75)

To explore how non-linear asymmetric device dynamics (Fig.1b) influence the performance of the memristive MLP, first, we implemented the simplest fixed-amplitude training procedure. It works well only for full batch training so we calculated the conductance updates ΔG for batches of 500 and 1000 patterns (for MNIST data set it is large enough to converge to the same results as full batch training). The SET pulse duration was τ_{SET} = 10μs, while RESET pulse duration τ_{RESET} was set to 2μs to match amplitudes of the corresponding SET and RESET peaks for effective threshold voltages V_{SET}^{TH} = -0.9V and V_{RESET}^{TH} = 1.4V (Fig. 1b). The results clearly show that there exists an optimum choice of training voltages that achieves the best performance of about 5% after 50 epochs (Fig. 4a,e), which correspond to a case of $\Delta G_{SET} = \Delta G_{RESET}$ of device conductance characteristics in a certain range (Fig. 4b).

Based on results shown on Fig. 4a and in order to make use of the whole range of voltages we adjusted the width of

the RESET pulse to be 1.3μs while keeping SET pulse width 10μs, so that SET and RESET conductances for the effective threshold voltages behave as shown on Fig. 4b. This ratio of SET and RESET pulse duration was always kept the same during training. Such pulse width-matched pair of effective threshold voltages $V_{\mathrm{SET}}{}^{\mathrm{TH}}$ and $V_{\mathrm{RESET}}{}^{\mathrm{TH}}$ was used to train the MLP with Manhattan rule. The batch training (500 patterns per batch) achieves relatively good misclassification rate of 1.98% after 100 epochs, while stochastic gradient descent as expected performs very poorly and converges at 19.26% after 50 epochs of training (Table I).

C. In-situ variable-amplitude training

Similarly pulse width-matching pairs were chosen for the whole range of training voltages and conductance change ΔG_0 was estimated at the equilibrium point that corresponds to G_0=230μS (Fig. 4b). The estimated ΔG_0 was used for variable-amplitude scheme to map $V = a\log[*] + b$ as a function of either e or x for stochastic algorithm, and k or $\Delta w/k$ for batch (indicated as [*]). The values of fitting parameters a and b are given in Table II. Note that argument [*] is normalized to be in [0, 1] range and voltages, which map outside of the considered ranges, are clipped to the corresponding minimum ($V_{\mathrm{SET}}{}^{\mathrm{TH}}$ and $V_{\mathrm{RESET}}{}^{\mathrm{TH}}$) or maximum ($2V_{\mathrm{SET}}{}^{\mathrm{TH}}$ and $2V_{\mathrm{RESET}}{}^{\mathrm{TH}}$) values.

This mapping was used to calculate training voltages for variable-amplitude batch (500 patterns per batch) and stochastic training and the results are presented in Table I. As one can see the result of batch training improved down to 1.47% misclassification rate and became comparable with that of a regular software implementation of the MLP (i.e. no device model and specialized memristive hardware considered). Stochastic training performance, while not yet on par with the regular MLP, was also improved drastically down to 4.06%.

D. Defect Tolerance

We have also analyzed defect tolerance of MLP classifier with respect to stuck-on-open (conductances are assumed to be stuck on G_{MIN} value and cannot be changed during training) and stuck-on-close (conductances stuck on G_{MAX} value) defects. Fig. 5 shows results for the best in-situ (variable-amplitude batch) and ex-situ with σ = 2% conductance import training approaches. In-situ training was calculated for two values of activation function coefficient β = 3.4mA^{-1} (the same as used in the simulations described above) and β = 0.87mA^{-1}. Use of a smaller β caused a slight degradation of the classifier performance as compared to the numbers given in Table I, but the resulting network became very robust to both stuck-on-open and stuck-on-close defects and classification performance is degraded by only 4% even in the presence of 90% defective devices (Fig. 5).

TABLE II. FITTING PARAMETERS FOR IN-SITU TRAINING.

	batch		stochastic	
	SET	RESET	SET	RESET
a	-0.74	1.87	-0.74	1.87
b	-1.41	2.64	-0.71	1.32

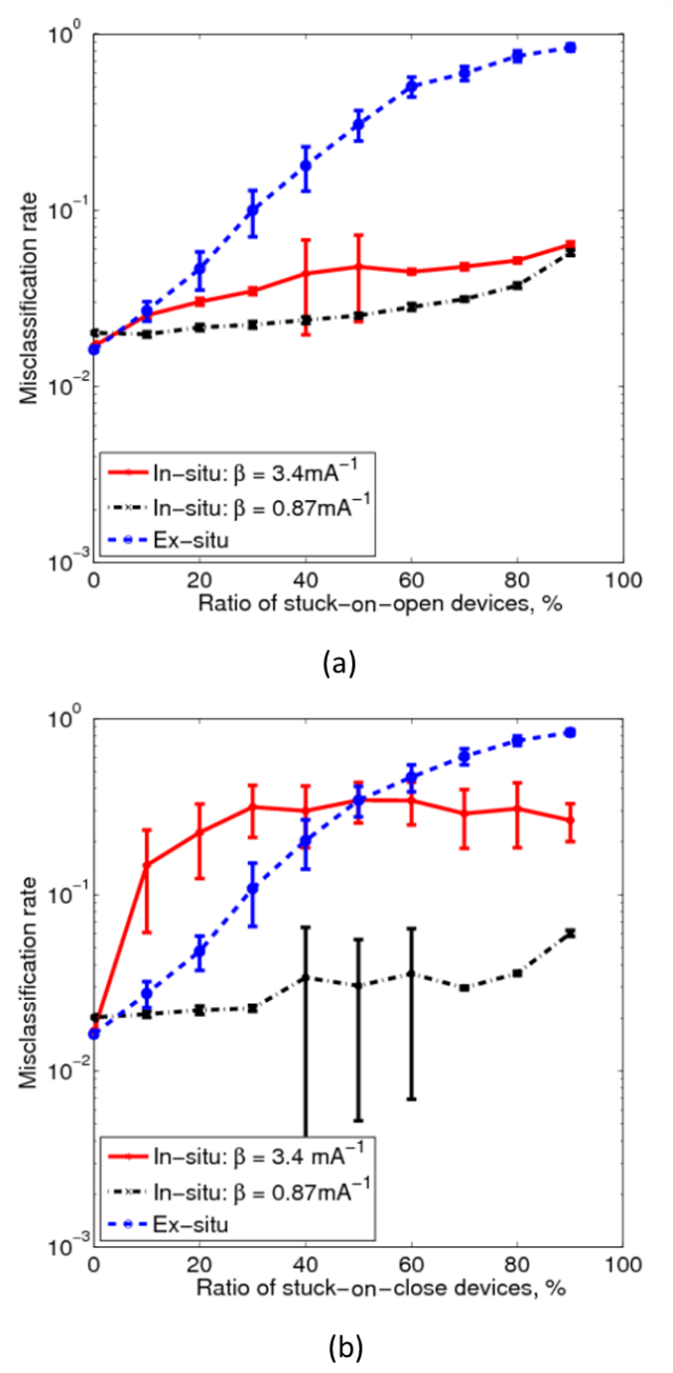

Fig. 5. Defect tolerance of MLP classifier with respect to a) stuck-on-open and b) stuck-on-close defects for 500-pattern-batch variable-voltage in-situ and ex-situ with σ = 2% weight import. Statistics are based on 20 runs.

V. DISCUSSION

Each of the considered training approaches comes with unique combination of implementation overhead / performance / defect tolerance, and therefore, represents an interesting design solution. The best performance of 1.47% and 1.54% are achieved for variable-amplitude batch in-situ and 2% precision weight import ex-situ training approaches, respectively (Table I), which is the same as the best reported results for similar networks [23].

The implementation overhead is perhaps the smallest for ex-situ training approach, however, it performs very poorly in the presence of defects and its performance quickly degrades even with a small fraction of both stuck-on-close and stuck-on-open devices. (Note that acquiring detailed information on defective devices should improve performance for ex-situ approach but would be hardly practical for large-scale networks).

On the other hand, with the right choice of parameters in-situ approach can efficiently adjust to defects and variations. Our experiments showed that in the case of a steep activation function slope, $\beta = 3.4\text{mA}^{-1}$, the MLP is very sensitive to stuck-on-close defects. It happens because if one device in a differential pair is stuck-on-close, it is very likely to result in an initial differential weight value close to its maximum ${G_{\text{MAX}}}^{\text{DIFF}} = 576\mu\text{S}$. Subsequently, the corresponding neuron's output is in the saturation region of the activation function, its derivative equal to zero and the network cannot learn the weights (3), (4). To a lesser degree a similar effect occurs in the case of stuck-on-open defects if one of the differential pair devices is stuck-on-open and the other non-defective device is accidentally close to the maximum value G_{MAX}. The use of the activation function with a lower slope of $\beta = 0.87\text{mA}^{-1}$ moves the neuron outputs with defective devices in differential pair weights away from the saturation region. As a result the MLP classifier becomes very robust to stuck-on-close defects and virtually insensitive to stuck-on-open defects and the performance is degraded by only 4% with 90% defective devices.

In-situ batch training is significantly better as compared to in-situ stochastic one, though the former relies on slower line-by-line updates and requires more extra circuitry, most importantly to store Δw between updates. In both cases variable-amplitude training takes into account device dynamics and carefully balances SET and RESET switching characteristics to dramatically improve performance and achieve misclassification rate comparable with that of the regular software implementation of the MLP (Table I). Examination of the evolution of misclassification rate and conductance distributions during in-situ fixed-amplitude training and corresponding ΔG curves (Fig. 4b-e) showed that the symmetry of ΔG_{SET} and ΔG_{RESET} is crucial for the convergence and stability of the memristive crossbar MLP. The best results are achieved when there exists a region of $\Delta G_{\text{SET}} = \Delta G_{\text{RESET}}$ around $G_0 = 230\mu\text{S}$ (Fig. 4b). On the other hand the performance degrades significantly when there is only one or two points of $\Delta G_{\text{SET}} = \Delta G_{\text{RESET}}$ (Fig. 4d) as all devices get pulled towards that single value of conductance state. The performance of the memristive crossbar MLP is the worst when SET process dominates for the whole range of conductances (Fig. 4c), since in this case all devices quickly collapse to the maximum value of G_{MAX} and the network weights become equal to zero.

In the best case scenario, ΔG_{SET} and ΔG_{RESET} symmetry is desirable for all values of conductances, but as long as such region is sufficiently wide amplitude and duration of applied voltage stimuli can be used to control memristors to stay there. Our results show that even SET/RESET conductance characteristics such as shown on Fig. 4b, which are far from perfect, achieve good performance (Table I). Engineering memristive devices to have wider symmetry regions of switching characteristics could help improve stochastic training performance further.

Finally, it is worth noting that there is a number of simplifications made in this preliminary study, which we are planning to address in a future work. For example, we neglected the effect of *I-V* non-linearity during classification and did not model the disturbance of half-selected devices.

VI. Conclusion

We have adapted backpropagation algorithm for training multilayer perceptron classifier implemented with memristive crossbar circuits. Taking into account device dynamics and balancing of SET and RESET switching characteristics dramatically improved performance of memristive multilayer perceptron classifier and achieved misclassification rate as low as 1.47% and 4.06% for batch and stochastic in-situ training algorithms comparable with that of a regular software implemented classifier. Pt/TiO_{2-x}/Pt devices used in this study are typical and the proposed method can be easily adapted for other memristive devices.

We have also performed preliminary analysis of defect tolerance and proposed a method to deal with memristive neural networks sensitivity to stuck-on-close defects.

References

[1] C. Mead, *Analog VLSI and Neural Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[2] K. K. Likharev, "Crossnets: Neuromorphic hybrid cmos/nanoelectronic networks," *Sci. Adv. Mater.*, vol. 3, pp. 322–331, 2011.

[3] S. K. Kim, L. C. McAfee, P. L. McMahon, and K. Olukotun, "A highly scalable restricted boltzmann machine fpga implementation," in *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pp. 367–372, IEEE, 2009.

[4] J. Kim, M. Kim, S. Lee, J. Oh, K. Kim, and H. Yoo, "A 201.4 GOPS 496 mw real-time multi-object recognition processor with bio-inspired neural perception engine," *J. Solid-State Circuits*, vol. 45, no. 1, pp. 32–45, 2010.

[5] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "Neuflow: A runtime reconfigurable dataflow processor for vision," in *Proceedings of Computer Vision and Pattern Recognition Workshops (CVPRW'11)*, pp. 109–116, June 2011.

[6] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," *SIGARCH Comput. Archit. News*, vol. 38, pp. 247–257, June 2010.

[7] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, pp. 699–716, 2014.

[8] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm," in *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, pp. 1–4, IEEE, 2011.

[9] S. Ramakrishnan and J. Hasler, "Vector-matrix multiply and winner-take-all as an analog classifier," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 22, pp. 353–361, Feb. 2014.

[10] J. Lu, S. Young, I. Arel, and J. Holleman, "A 1tops/w analog deep machine-learning engine with floating-gate storage in 0.13 μm cmos," in *IEEE Int. Solid-State Circuits Conf.(ISSCC) Dig. Tech. Papers*, 2014.

[11] *The International Technology Roadmap for Semiconductors (ITRS)*. 2013 ed. Available online at http://www.itrs.net/.

[12] D. Strukov, "Nanotechnology: Smart connections," *Nature*, vol. 476, 2011.

[13] R. Williams, "How we found the missing memristor," *IEEE Spectr.*, vol. 45, pp. 28–35, Dec. 2008.

[14] D. Strukov and H. Kohlstedt, "Resistive switching phenomena in thin films: Materials, devices, and applications," *MRS Bulletin*, vol. 37, February 2012.

[15] D. Kuzum, R. G. D. Jeyasingh, B. Lee, and H.-S. P. Wong, "Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing," *Nano Letters*, vol. 12, no. 5, pp. 2179–2186, 2012.

[16] N. Locatelli, V. Cros, and J. Grollier, "Spin-torque building blocks," *Nature materials*, vol. 13, no. 1, pp. 11–20, 2014.

[17] Y. Kaneko, Y. Nishitani, and M. Ueda, "Ferroelectric artificial synapses for recognition of a multishaded image," 2014.

[18] F. Alibart, E. Zamanidoost, and D. Strukov, "Pattern classification by memristive crossbar circuits with ex-situ and in-situ training," *Nature Communications*, vol. 4, p. 2072, 2013.

[19] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Letters*, vol. 10, no. 4, pp. 1297–1301, 2010.

[20] T. Ohno, T. Hasegawa, T. Tsuruoka, K. Terabe, J. K. Gimzewski, and M. Aono, "Short-term plasticity and long-term potentiation mimicked in single inorganic synapses," *Nature materials*, vol. 10, no. 8, pp. 591–595, 2011.

[21] J. Yang, D. Strukov, and D. Stewart, "Memristive devices for computing," *Nature Nanotechnology*, vol. 8, pp. 13–24, 2013.

[22] M. Prezioso, F. Merrikh-Bayat, B. Hoskins, G. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, May 2015 (in print).

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, November 1998.

[24] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *Neural Networks: Tricks of the trade* (G. Orr and M. K., eds.), Springer, 1998.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[26] J. H. Lee and K. K. Likharev, "Defect-tolerant nanoelectronic pattern classifiers," *International Journal of Circuit Theory and Applications*, vol. 35, no. 3, pp. 239–264, 2007.

[27] D. Chabi, D. Querlioz, W. Zhao, and J. Klein, "Robust learning approach for neuro-inspired nanoscale crossbar architecture," *JETC*, vol. 10, no. 1, p. 5, 2014.

[28] C. Yakopcic and T. M. Taha, "Energy efficient perceptron pattern recognition using segmented memristor crossbar arrays," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–8, IEEE, 2013.

[29] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a memristor-based spiking neural network immune to device variations," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pp. 1775–1781, IEEE, 2011.

[30] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, "Memristor crossbar-based neuromorphic computing system: A case study," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 10, pp. 1864–1878, 2014.

[31] J. Starzyk and Basawaraj, "Memristor crossbar architecture for synchronous neural networks," *IEEE Trans. Circ. Syst. - I*, vol. 61, pp. 2390 – 2401, 2014.

[32] P. Sheridan, W. Ma, and W. Lu, "Pattern recognition with memristor networks," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pp. 1078–1081, IEEE, 2014.

[33] Y. Kim, Y. Zhang, and P. Li, "A digital neuromorphic vlsi architecture with memristor crossbar synaptic array for machine learning," in *SOC Conference (SOCC), 2012 IEEE International*, pp. 328–333, IEEE, 2012.

[34] F. Merrikh-Bayat, B. Hoskins, and D. B. Strukov, "Phenomenological modeling of memristive devices," available online at https://www.ece.ucsb.edu/~strukov/papers/2014/mmodel2014.pdf.

[35] F. Merrikh Bayat, B. Hoskins, and D. Strukov, "Phenomenological modeling of memristive devices," *Applied Physics A*, pp. 1–8, 2015.

[36] J. Hertz, A. Krogh, and R. Palmer, "Introduction to the theory of neural networks," *Lecture Notes*, vol. 1, 1991.

[37] F. Alibart, L. Gao, B. Hoskins, and D. Strukov, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, p. 075201, 2012.

[38] K. K. Likharev, "Hybrid cmos/nanoelectronic circuits: Opportunities and challenges," *Journal of Nanoelectronics and Optoelectronics*, vol. 3, no. 3, pp. 203–230, 2008.

[39] W. Schiffmann, M. Joost, and R. Werner, "Optimization of the backpropagation algorithm for training multilayer perceptrons," *Univ. Koblenz, Inst. Physics, Rheinau*, pp. 3–4, 1994.