Exponential-Weight Multilayer Perceptron

Farnood Merrikh Bayat, Xinjie Guo and Dmitri Strukov
ECE Department
University of California Santa Barbara
CA 93106-9560, USA
E-mail: farnoodmb@ece.ucsb.edu

***Abstract*—Analog integrated circuits may increase the neuromorphic network performance dramatically, leaving far behind their digital and biological counterparts, while approaching the energy efficiency of the brain. The key component of the most advanced analog circuit implementations is a nanodevice with adjustable conductance - essentially an analog nonvolatile memory cell, which could mimic synaptic transmission function by multiplying signal from the input neuron (e.g. encoded as voltage applied to the memory device) by its analog weight (device conductance) and passing the product (the resulting current) to the output neuron. Such functionality enables very dense, fast, and low power implementation of dot-product computation, the most common operation in many artificial neural networks. The most promising analog memory devices, however, have nonlinear, typically exponential, I-V characteristics, which result in nonlinear synaptic transmission, thus limiting their application in analog dot-product circuits. Here we investigate multilayer perceptron with exponential transmission function synapses which maps naturally to the most advanced analog neuromorphic circuits. Our simulation results show that the proposed exponential-weight multilayer perceptron with 300 hidden neurons achieves classification performance comparable to the similar-size linear-weight network when benchmarked on MNIST dataset. Moreover, we verify the proposed idea experimentally by implementing small-scale single-layer exponential-weight perceptron classifier with an NOR-flash memory integrated circuit.**

I. Introduction

The name "perceptron" today typically implies a well-known algorithm for supervised training of linear classifiers. Interestingly, a more general version of perceptron algorithm was proposed in early 1960s (Fig. 1(a)) [1]. In such algorithm, all *Sensory* (input), *Association* (hidden) and *Response* (output) neurons are of the form $f(\gamma_i)$, where $f(\cdot)$ is neuron's activation function and γ_i is the algebraic sum of all input signals while synaptic transmission function is a function of input and current synaptic state of the synapse, i.e. $g(x_i, w_i)$, so that the output of the neuron is written as:

$$y = f\left[\sum_i g(x_i, w_i)\right] \tag{1}$$

For linear synaptic transmission function, i.e. the most commonly used case today, this equation simplifies to a typical weighted-sum (dot-product) operation:

$$y = f\left[\sum_i x_i w_i\right] \tag{2}$$

with the corresponding well-known single-layer perceptron network shown in Fig. 1(b).

In the early years of artificial neural networks, the motivation for more general architecture was rather obvious considering the inherent limitations of linear perceptron, which can perform classification for only linearly-separable patterns [1]. It seems though that this idea has not received much attention, due to subsequent invention of more powerful multilayer networks in which the needed nonlinearity is provided by neurons rather than synapses [2]. Additionally, simple product operation is certainly easier to compute using digital circuits, which have been traditionally used to implement artificial neural networks.

The recent advances in nanoelectronic memory devices and the opportunity to use such devices in analog integrated circuit for neuromorphic computing make the idea of nonlinear synaptic transmission function very appealing again [3], [4]. Indeed, the potential advantages of specialized analog-circuit hardware for neuromorphic computing have been realized long time ago [5], but their practical realization requires very compact circuit elements with adjustable conductance - essentially nanoscale analog nonvolatile memory cells, whose state can be tuned with a high precision. Such memory cells are utilized to implement synapses, the most numerous devices in artificial neural networks, while the integration of memory devices in dense crossbar circuits in turn enables very compact, fast, and energy-efficient analog circuit computation of a dot-product [6]–[8], a typical bottleneck operation for many neural networks. In particular, in such crossbar circuits memory cell mimics synapse functionality by multiplying the signal passing through the device (e.g. encoded as a voltage applied to the memory cell) by its weights (device conductance) and passing the product (the resulting current) to the output neuron. A number of promising analog nanoscale nonvolatile memories have been developed in recent years [3], [9], however, many of such devices feature nonlinear I-V characteristics, thus limiting the application of such devices for purely linear-weight artificial neural networks.

In this paper we are revisiting a more general case of perceptron algorithm, in particular focusing on the exponential synaptic transmission function ("exponential-weight") networks and investigate this approach in the context of a specific analog circuit implementation. Although neuromorphic hardware with nonlinear weights has been proposed before [10], [11], in the earlier works the considered nonlinear synaptic element lacked efficient hardware implementation and/or there was inconsistency between the proposed algorithm and the underlying hardware.

978-1-5090-6182-2/17/$31.00 ©2017 IEEE

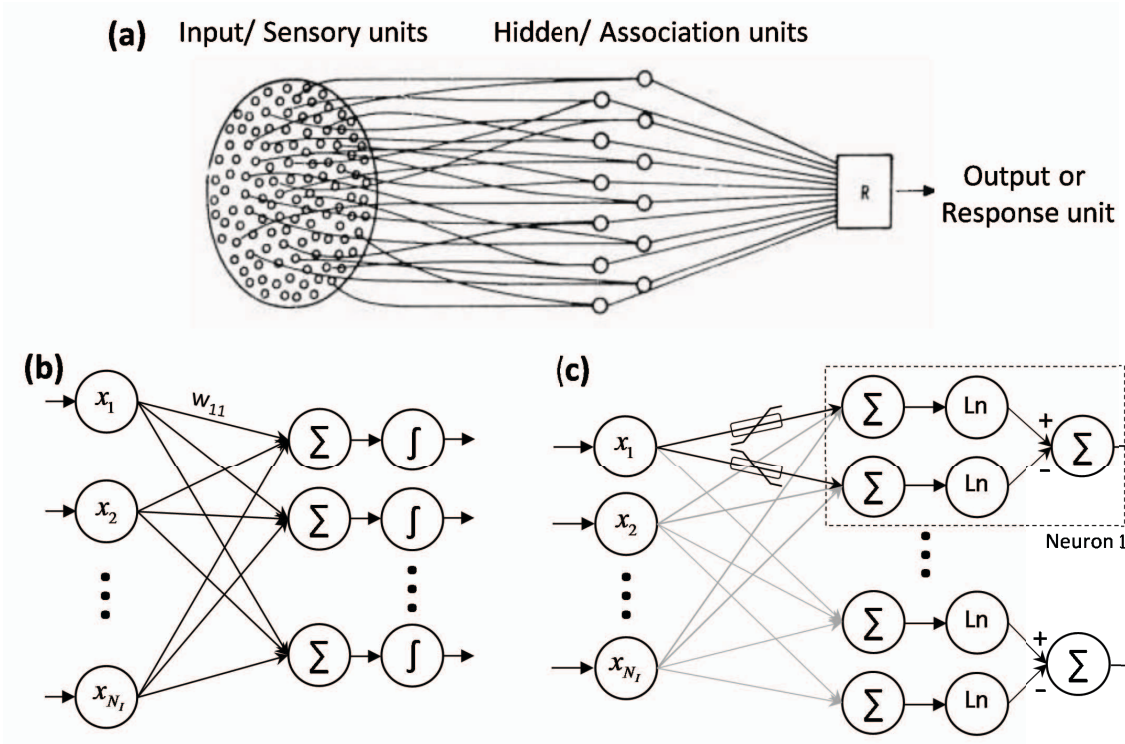

Fig. 1: (a) A general perceptron network proposed by Rosenblatt in 1961 [1]. (b) Implementation of perceptron using synapses with linear transmission function. (c) The proposed perceptron implementation with nonlinear (exponential) synaptic transmission functions.

The remainder of the paper is organized as follows. In the next section we derive backpropagation algorithm for the exponential synaptic transmission function, which is representative of the considered analog floating-gate nanoelectronic implementation. In Section 3 we discuss simulation results for exponential-weight multilayer perceptron trained with modified backpropagation algorithm and compare it with conventional linear-weight networks. Two representative nanoelectronic implementations for the proposed nonlinear-weight networks and preliminary experimental results are discussed in Section 4 and the paper is concluded in Section 5.

II. Neural Network with Exponential Weights

A. Feedforward computation

In order to develop a network with exponential weight, let's start with the conventional perceptron (Eq. (2)). Since exponential functions are strictly positive, let's put a fairly weak constraint on this network by assuming that all of its weights are always non-negative. The effective bipolar synaptic weights could be then implemented as a difference of two non-negative weights as follows [4], [12]:

$$y = f\left[\sum_i x_i w_i\right] = f\left[\sum_i x_i (w_i^+ - w_i^-)\right] = f\left[\sum_i x_i w_i^+ - \sum_i x_i w_i^-\right], \quad \forall w_i^+, w_i^- \geq 0 \qquad (3)$$

Now by replacing linear synaptic weights with nonlinear ones we get:

$$y = f\left[\sum_i g(x_i, w_i^+) - g(x_i, w_i^-)\right], \quad \forall w_i^+, w_i^- \geq 0 \qquad (4)$$

Note that by substituting $g(\cdot,\cdot)$ with the multiplication function, (4) simplifies to (3). Although in general any properly defined nonlinear function can be used as $g(\cdot,\cdot)$, for the reason that would become clear later, we define $g(\cdot,\cdot)$ as $g(x,w) = e^{\alpha(x-w)}$ where α is a positive constant. Also, we are only interested in those activation functions that meet the property of $f(A-B) = g(A) - g(B)$. In this case, by defining $g(\cdot) = \beta \ln(\cdot)$ which would also automatically bound negative and positive sums, (4) can be expressed as:

$$y = \beta\left(\ln\left[\sum_i e^{\alpha(x_i - w_i^+)}\right] - \ln\left[\sum_i e^{\alpha(x_i - w_i^-)}\right]\right) = \beta \ln\left(\frac{\sum_i e^{\alpha(x_i - w_i^+)}}{\sum_i e^{\alpha(x_i - w_i^-)}}\right), \quad \forall w_i^+, w_i^- \geq 0 \qquad (5)$$

where here $\beta > 0$ is the output gain of the neuron. The corresponding graphical model of this equation is shown in Fig. 1(c). Interestingly, in contrast to (3) where neuron output is proportional to the difference between the contributions of positive (i.e. $\sum_i x_i w_i^+$) and negative weights (i.e. $\sum_i x_i w_i^-$), here the output is proportional to the ratio of these two terms

(i.e. $\sum_i e^{\alpha(x_i-w_i^+)}$ and $\sum_i e^{\alpha(x_i-w_i^-)}$). However, for both cases the output is positive only if the contribution of positive weights is larger than the contribution of negative weights. Note that exchanging the place of subtraction and $ln(\cdot)$ in (5) (i.e. using $\ln(A)-\ln(B)$ instead of $\ln(A-B)$) is mainly done to assure that the argument of the $ln(\cdot)$ function is always strictly positive. Also note that due to the nonlinear nature of (5), no extra activation function is required.

B. Gradient calculation

To train the proposed network for any particular task like pattern classification, parameters w_i^+ and w_i^- need to be optimized, for example, by using ordinary steepest-descent optimization method which requires calculating derivative of the cost function with respect to these parameters. Fortunately, since the synaptic transmission functions are in the form of exponential function, calculation of gradient is simple and straightforward. Although the following discussion would also be valid for any other cost function, for the sake of simplicity, here let's assume the cost function to be the mean square error between the output and the target for all training data written as:

$$E=\frac{1}{2}\sum_{p=1}^{N_p}(y_p-t_p)^2 \tag{6}$$

where N_p is the number of training data patterns and t_p is the target value corresponding to the applied input. Taking derivative of this cost function with respect to w_i^+ and w_i^- yields to:

$$\frac{\sigma E}{\sigma w_i^{\pm}}=\sum_{p=1}^{N_p}\left((y_p-t_p)\frac{\sigma y_p}{\sigma w_i^{\pm}}\right)= \sum_{p=1}^{N_p}\left((y_p-t_p)\times\beta\frac{\mp\alpha e^{\alpha(x_i-w_i^{\pm})}}{\sum_j e^{\alpha(x_j-w_j^{\pm})}}\right) \tag{7}$$

Note that in these equations $\sum_j e^{\alpha(x_j-w_j^{\pm})}$ and $e^{\alpha(x_i-w_i^{\pm})}$ are two terms which are previously calculated during the forward operation of the network in (5) and don't need to be recalculated. This equation implies that the gradient of each weight is proportional to its contribution to the total outcome of all weights of its kind. In other words, weights are increased or decreased during training proportionally to thier contribution to the error (y_p-t_p). As a comparison, in the case of linear weights (see, e.g. Eq. (3)), the term $\frac{-e^{\alpha(x_i-w_i^+)}}{\sum_j e^{\alpha(x_j-w_j^+)}}$ in a gradient function becomes $x_i f'(\sum_j x_j w_j)$ which means that a small input with negligible contribution to a weighted sum can still result in a large gradient (except for activation functions like *tanh* or *sigmoid* where their derivative goes to zero for larger $\sum_j x_j w_j$.

C. Generalization to multilayer networks

Construction of a multilayer network with exponential weights is also straightforward and can be done by the concatenation of single-layer networks introduced in Sec. II-A and shown in Fig. 1(c). In this case, output of one layer is directly applied to the input of another layer. For a network with only one hidden layer, forward computation can be written as:

$$h_j=\beta\ln\left(\frac{\sum_{i=1}^{N_I}e^{\alpha(x_i-w_{ij,1}^+)}}{\sum_{i=1}^{N_I}e^{\alpha(x_i-w_{ij,1}^-)}}\right), \quad \forall w_{ij,1}^+, w_{ij,1}^- \geq 0, \quad j=1,\cdots,N_H \tag{8}$$

$$y_k=\beta\ln\left(\frac{\sum_{i=1}^{N_H}e^{\alpha(h_i-w_{ik,2}^+)}}{\sum_{i=1}^{N_H}e^{\alpha(h_i-w_{ik,2}^-)}}\right), \quad \forall w_{ik,2}^+, w_{ik,2}^- \geq 0, \quad k=1,\cdots,N_O \tag{9}$$

where N_I, N_H and N_O respectively are numbers of input, hidden and output neurons, $w_{ij,1}$ is the weight connecting the ith input neuron to jth hidden neuron, $w_{ij,2}$ is the weight that connects the ith hidden neuron to jth output neuron, and x, h and y are the outputs of input, hidden and output neurons, respectively. Taking derivative of the cost function with respect to the variable parameters of the network, i.e. $w_{ij,1}^{\pm}$ or $w_{ij,2}^{\pm}$, is straightforward so here we only summarize final gradient equations:

$$\frac{\sigma E}{\sigma w_{ij,2}^{\pm}}=\sum_{p=1}^{N_p}\left((y_j-t_j)\beta\frac{\mp\alpha e^{\alpha(h_i-w_{ij,2}^{\pm})}}{\sum_{l=1}^{N_H}e^{\alpha(h_l-w_{lj,2}^{\pm})}}\right) \tag{10}$$

$$\frac{\sigma E}{\sigma w_{ij,1}^{\pm}}=\sum_{p=1}^{N_p}\sum_{k=1}^{N_O}(y_k-t_k)\frac{\sigma y_k}{\sigma h_j}\times\frac{\sigma h_j}{\sigma w_{ij,1}^{\pm}}= \sum_{p=1}^{N_p}\left(\frac{\mp\beta\alpha e^{\alpha(x_i-w_{ij,1}^{\pm})}}{\sum_{l=1}^{N_I}e^{\alpha(x_l-w_{lj,1}^{\pm})}}\times\sum_{k=1}^{N_O}\left((y_k-t_k)\frac{\beta\alpha e^{\alpha(h_j-w_{jk,2}^{\pm})}}{\sum_{l=1}^{N_H}e^{\alpha(h_l-w_{lk,2}^{\pm})}}\right)\right) \tag{11}$$

We next use the modified backpropagation algorithm to simulate multilayer neural networks with exponential synaptic transmission functions.

III. Simulation results

As pointed out by Rosenblatt and others, different neural networks with different synaptic functions can generate comparable results provided that they have similar numbers of independent parameters. Since for the selected synaptic function of $g(x,w)=e^{\alpha(x-w)}=e^{\alpha x}e^{-\alpha w}$, there is still a sort of multiplication between x and w , it is natural to expect the performance of the proposed network to be close to that of the same size conventional linear-weight networks. Figure 2 compares classification performance of the networks with linear and exponential synapses for batch trainings. MNIST dataset were used as a benchmark which consists of 60,000 28×28-pixel grayscale training images and 10,000 test patterns

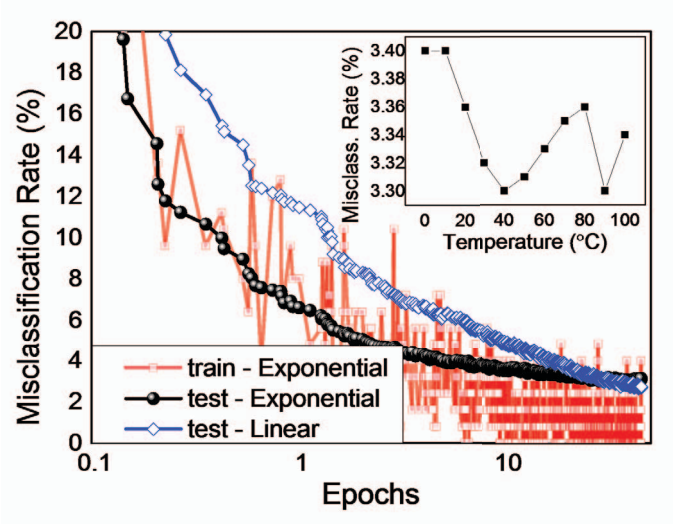

Fig. 2: Simulation results for exponential- and linear-weight networks trained in batch mode. The figure shows how for the network with exponential weights the misclassification rate has been changing during training for both training and test data sets. For the network with linear weights, the learning curve is shown only for the test data set. The inset shows that despite working in subthreshold region, the performance of the network after training doesn't change with the change of ambient temperature.

whose pixel intensities were normalized to [0, 1]. In order to be consistent with other reported results [13], in all networks, the number of neurons in the hidden layer was set to 300 and each synapse was implemented in differential mode using two non-negative weights. Moreover, for the network with linear weight the activation function was *ReLU* [14]. For the proposed network α and β were respectively set to 8.7 and 8 (which is based on the target hardware) and for the batch training, each batch had 125 patterns. Both linear and exponential networks were trained using ADAM stochastic optimizer and were regularized during training using Dropout [15] technique. As Fig. 2 illustrates, the performance of the proposed network is comparable to that of the conventional linear-weight network.

In the next section we experimentally verify that the proposed approach is suitable for the most advanced analog circuit implementations of artificial neural networks.

IV. Hardware implementation and experimental results

A. Emerging memory technologies

Memristive crossbar circuits [7] and NOR-based flash memories [16], [17] are very promising candidates for implementing neural network architectures [4], [7]. Synaptic weights are implemented naturally as a crosspoint device conductances in memristive crossbar circuit (Fig. 3(a)), which could be made very compact (Fig. 3(b)). However, as shown in Fig. 3(c), in order to have a good linear weighted-sum operation, the maximum working voltage and resistance should be kept as low as possible, which, in turn, increases the power consumption (due to the decrease of resistance) and decreases signal-to-noise-ratio (due to the decrease of operating voltage). By increasing the amplitude of input signal to higher voltages (but still keeping it below the switching threshold of the device), the static section of the I-V curve becomes more and more nonlinear. In this case each device inherently implements semi-exponential synaptic transmission function with stronger nonlinearity for larger resistances [18].

Another device with memory that has very strong nonlinearity in a wide range of working conditions is floating-gate or flash transistor. Figures 3(d) and (e) show how a densely fabricated NOR-like memory array of flash transistors can be used to implement the neural network architecture of Fig. 1(c). In this circuit, analog inputs are directly applied to the gates of flash transistors so the current of each device is representing function $g(x_i, w_i)$ in (1). As demonstrated by the I-V characteristics of in Fig. 3(f), the drain currents depend exponentially on the applied gate voltage in subthreshold region ($I_d < 100$ nA) which similar to ordinary transistor can be expressed as:

$$I_d = I_{d_0} e^{\left(\frac{V_{gs}-V_{th}}{nV_T}\right)} \overset{V_s=0}{=} I_{d_0} e^{\left(\frac{V_g-V_{th}}{nV_T}\right)} \quad \text{when} \quad V_{gs} < V_{th} \quad \text{and} \quad V_{ds} >\sim 4V_T \tag{12}$$

where V_{gs} is gate-to-source bias voltage, V_{th} is the threshold voltage of the device, I_{d_0} is the current at $V_{gs} = V_{th}$, $V_T = kT/q \simeq 25$ mV, and the slope factor n is a function of capacitances of the depletion and oxide layers. For the fabricated devices of Fig. 3(f), $I_{d_0} = 10$ nA, V_{th} can change from 0.45 V (for fully erased device) to 3.35 V (for fully programmed transistor) while $n = 1 + 3.6 \times \tanh(1.1(V_{th} - 0.3))$.

B. Experimental results

Power consumption of analog circuits can be drastically decreased by biasing transistors in subthreshold region. Comparing (12) with $g(x, w) = e^{\alpha(x-w)}$ suggests that synaptic transmission function of $g(\cdot, \cdot)$ can be implemented in hardware using a single transistor biased in subthreshold regime, provided that we be able to tune/program its threshold voltage V_{th} - a property which has been around for decades thanks to floating-gate or flash transistors. This means that in contrast to linear but slow and bulky analog/digital multipliers used in hardware implementing conventional neural networks, now we can implement each synapse with a single transistor which not only is acting as a memory by storing synaptic weight (as its threshold voltage which can be easily tuned), but also is performing low-power exponential operation. However, when dealing with ultra large scale neural networks or deep neural networks, every part of the system, ranging from physical devices to architecture, should be in harmony with each other and only having a perfect way to implement synaptic weight doesn't guarantee a high quality network. Fortunately, at architecture level the proposed network of (5) maps straightforwardly to the array of memory elements (floating-gate transistors) as

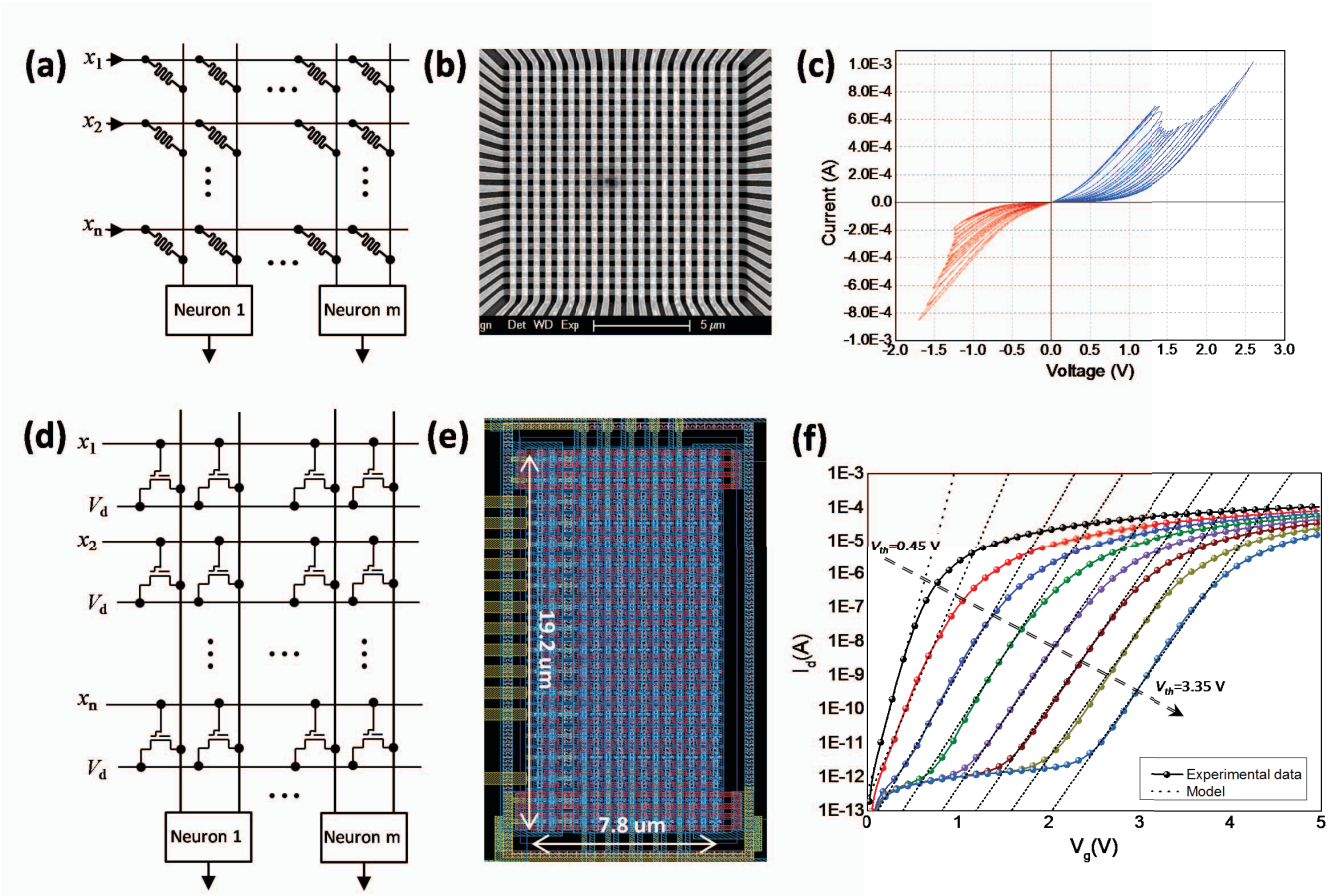

Fig. 3: Emerging hardware technologies for the proposed exponential-weight perceptron networks: (a) Circuit schematics for memristor-based approach for implementing Eq.(1), (b) micrograph of the fabricated 20×20 memristor crossbar, and (c) typical switching I-V curves of a single memristor showing the inherent nonlinearity in the static behavior of the device. (d) Circuit schematics for floating-gate memory approach for implementing (1), (e) layout of the fabricated 10×10 memory array in 180 nm process (based on the modified industrial-grade NOR flash memory [16]), and (c) Typical I_d vs V_g curves of a single floating-gate transistor programmed to different states, in particular showing the inherent exponential relationship between I_d and V_g in subthreshold region.

shown in Fig. 3(d). In this circuit, scaled inputs (to assure that all floating-gate transistors are always in subthreshold region independent from their threshold voltages) are directly applied to the control gates of floating-gate transistors with virtually grounded sources so each transistor is performing the function of $g(x,w) = I_{d_0} e^{\alpha(x-w)}$ for $\alpha = 1/0.115 = 8.7$ based on its gate voltage and preprogrammed V_{th}. The summation operation in (5) is done on the shared source lines thanks to the Kirchhoff's current law. The resulting current is then scaled logarithmically using a transistor placed at a feedback of an operational amplifier. Finally, the subtraction can be performed using a simple voltage subtractor circuitry.

The memory array used in the architecture of Fig. 3(d) is fabricated with very high density by modifying a commercially available NOR-based digital flash memories [16]. This small modification, i.e. routing word/gate lines vertically rather than horizontally, allows individual programmability of floating-gate transistors. This is a fundamental requirement for our architecture because in both ex-situ and in-situ training, we need to be able to update/tune particular floating-gate transistors inside the array without altering other devices. To summarize, using a modified version of NOR flash memories and CMOS-based neurons, the proposed neural network with exponential synapses can be implemented in analog domain with ultra high density which can be easily scaled up to much larger networks. Moreover, power consumption of the hardware is drastically decreased by using all flash transistors in subthreshold region, while processing speed is increased considerably (compared to digital implementations) by avoiding the time wasted for fetching weights from external memory and also by performing all synaptic transmission functions in parallel and at once.

Functionality of the proposed neural network was tested by fabricating a small scale (10×10) flash memory array in 180 nm process using SST's SuperFlash technology [19]. The

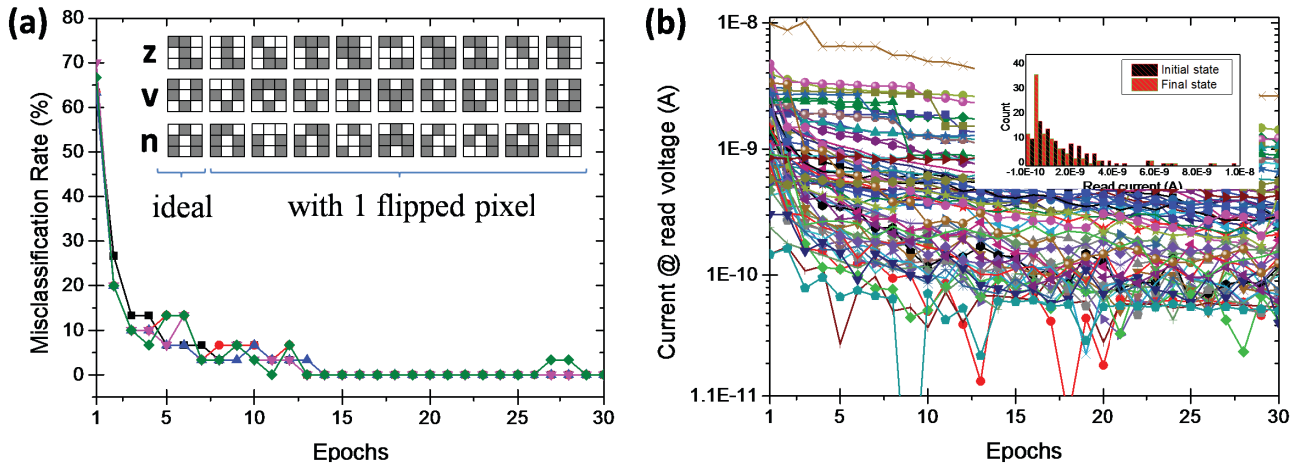

Fig. 4: Experimental results for the proposed network with exponential weights: (a) In-situ training of the hardware in batch mode to classify 30 input patterns (shown as an inset) into 3 classes (for 5 runs). (b) Evolution of cells' conductance $g(0.5V, w_i)$ during in-situ training of the network upon application of weight updating pulses. States of all 60 devices are measured at the read condition of $V_g = 0.5$ V, $V_d = 0.6$ V and $V_s = 0$ V. The inset shows the histogram of $g(0.5V, w_i)$s before and after the training.

fabricated array was then used to implement a single layer neural network with the top-level (functional) scheme shown in Fig. 3(d) while neuron circuits were emulated in software. The network had ten inputs and three outputs, fully connected with $10 \times 3 \times 2 = 60$ differential weights each implemented with a single floating-gate transistor. Such a network is sufficient for performing, for example, the classification of 3×3-pixel images into three classes with nine network inputs corresponding to the pixel values. We tested the network on a set of 30 patterns, including three stylized letters ('z', 'v' and 'n') and three sets of nine noisy versions of each letter, formed by flipping one of the pixels of the original image (see the inset of Fig. 4(a)). Physically, each input signal was represented by a voltage equal to either 0.5 V or 0 V, corresponding, respectively, to the black or white pixel, while the bias input was equal to 0.5 V.

The network was trained in situ, that is, without using its external computer model, using the Manhattan update rule, which is essentially a coarse-grain, hardware-friendly, batch-mode variation of the usual delta rule of supervised training. After initializing all flash transistors to near fully-erased state, at each iteration ('epoch') of this procedure, patterns from the training set were applied, one by one, to the network's inputs while $V_d = 0.6$ V and $V_s = 0$ V, and then its outputs were measured according to (5). (Note that $\sum_i exp(\alpha(x_i - w_i))$ corresponds to the current measured at virtually grounded source line.) Once all N patterns of the training set had been applied and all gradients were calculated based on (7), the synaptic weights were updated in parallel based on the Manhattan update rule (sign of the gradients) using fixed-amplitude erasure and programming pulses. To update flash transistors in parallel, in each epoch we used the modified version of the algorithm originally proposed for

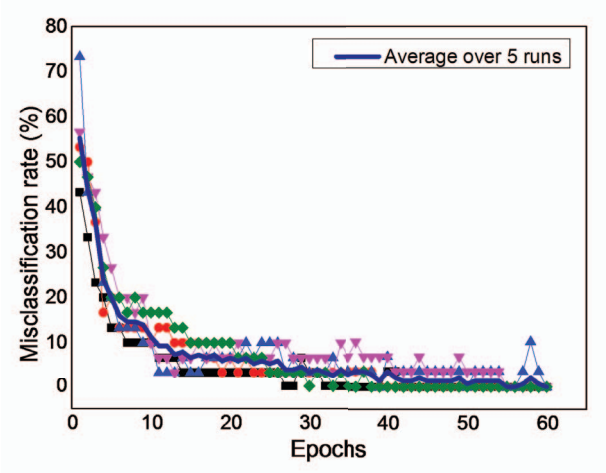

Fig. 5: Results of an experiment similar to the one in Fig. 4, this time assuming a faulty array in which 12 randomly selected devices (20% of all devices) were marked as faulty and turned off during training (completely programmed). The experiment was repeated five times, each time with different initial weights and different configuration of faulty devices.

parallel updating of memristive devices in crossbar structure [12]. The weight stored in each flash transistor was decreased by applying a single programming pulse (6 V, 5 us) to the source line while synapse potentiation was done by applying an erasure pulse (8 V, 2.5 ms) to the control gate (see Ref. [16] for more details).

For this particular classification task, the perfect classification for five different runs was reached, on average, after 13 training epochs (see Fig. 4(a)). Figure 4(b) shows how the synaptic

transmission functions of 60 flash transistors implementing synaptic weights in this experiment had evolved during training (with the histogram of initial and final read currents shown in the inset). For the fully trained network, the average current of each floating-gate transistor was 0.34 nA at the specific read condition of $V_d = 0.6$ V, $V_g = 0.5$ V and $V_s = 0$ V which corresponds to the power consumption of 200 pW/synaptic weight (when the input is at its maximum). Note that in this architecture power consumption of the whole network can be decreased even further by biasing flash transistors in deeper subthreshold (by lowering V_g) with the cost of having smaller signal-to-noise ratio and slower hardware.

One drawback of biasing transistors in subthreshold is that their drain-source current becomes very sensitive to temperature. This means that after training the network, the performance of the network can be completely different at different temperatures. Fortunately, in the proposed architecture this problem can be avoided by using a similar flash transistor to implement the $\ln(\cdot)$ function in (5). In this case, as also shown in the inset of Fig. 2, the performance of the network is stable at wide range of temperature.

Finally, to investigate the sensitivity of the proposed architecture to faulty devices inside the array, the experiment of Fig. 4 was repeated, with 20% of devices were assumed to be faulty. Fortunately, unlike other emerging memory technologies like memristive devices, faulty devices in flash memory arrays only have lower or higher switching voltages compared to the rest of the devices and they do not stuck in any particular resistance state. This means that prior to training, all faulty devices can be completely turned off which have shown to have a minimum effect on the performance of training in faulty hardware [20]. Figure 5 shows that despite all these faulty devices, the hardware still could be successfully trained to classify all patterns.

V. Conclusion

Inspired by the intrinsic nonlinear behavior of emerging nanoelectronic memory devices, in this paper we introduced a new type of neural network with exponential synaptic transmission functions. We modified backpropagation algorithm for the considered transmission function, simulated the performance of medium-scale multilayer exponential-weight perceptron and showed that it compared favorably to that of linear-weight networks. Finally, we successfully verified experimentally the proposed idea by implementing small-scale pattern classifier with integrated NOR-flash memory circuit. We expect that the proposed exponential-weight networks will have an immediate impact for the most advanced analog neuromorphic circuits by broadening the scope of nanoelectronic memory devices that can be utilized to implement synaptic transmission function and by allowing to use such memory devices more efficiently, e.g. without having to confine the operation to a linear regime. Ultimately, we hope that this approach will be an important step towards the development of analog neuromorphic hardware with comparable and even exceeding energy-efficiency and performance to that of the human brain.

Acknowledgment

This work was supported by DARPA under Contract No. HR0011-13-C-0051UPSIDE via BAE Systems. Useful discussions with Konstantin K. Likharev and Michael Klachko are gratefully acknowledged.

References

[1] F. Rosenblatt, *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Buffalo, NY, USA: Cornell aeronautical Lab., Inc., 1961.

[2] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[3] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nat nanotechnol*, vol. 8, no. 1, pp. 13–24, 2013.

[4] F. Merrikh Bayat, X. Guo, M. Klachko, M. Prezioso, K. Likharev, and D. Strukov, "Sub-1-us, sub-20-nj pattern classification in a mixed-signal circuit based on embedded 180-nm floating-gate memory cell arrays," *arXiv preprint arXiv:1610.02091*, 2016.

[5] C. Mead, *Analog VLSI and neural systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[6] C. Schlottmann and P. E. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: The fpaa implementation.," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 1, no. 3, pp. 403–411, 2011.

[7] 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*, vol. 521, no. 7550, pp. 61–64, 2015.

[8] X. Guo, F. Merrikh Bayat, M. Prezioso, Y. Chen, B. Nguyen, N. Do, and D. Strukov, "Temperature-insensitive analog vector-by-matrix multiplier based on 55 nm nor flash memory cells," *arXiv preprint arXiv:1611.03379*, 2016.

[9] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–oxide rram," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[10] M. Milev and M. Hristov, "Analog implementation of ann with inherent quadratic nonlinearity of the synapses," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1187–1200, 2003.

[11] J. B. Lont and W. Guggenbuhl, "Analog cmos implementation of a multilayer perceptron with nonlinear synapses," *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 457–465, 1992.

[12] F. Alibart, E. Zamanidoost, and D. B. Strukov, "Pattern classification by memristive crossbar circuits using ex situ and in situ training," *Nature communications*, vol. 4, 2013.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8609–8613, IEEE, 2013.

[15] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[16] F. Merrikh Bayat, X. Guo, H. Om'mani, N. Do, K. K. Likharev, and D. B. Strukov, "Redesigning commercial floating-gate memory for analog computing applications," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1921–1924, IEEE, 2015.

[17] C. Diorio, P. Hasler, B. A. Minch, and C. A. Mead, "A floating-gate mos learning array with locally computed weight updates," *IEEE Transactions on Electron Devices*, vol. 44, no. 12, pp. 2281–2289, 1997.

[18] F. Merrikh Bayat, B. Hoskins, and D. B. Strukov, "Phenomenological modeling of memristive devices," *Applied Physics A*, vol. 118, no. 3, pp. 779–786, 2015.

[19] hhttp://www.sst.com/technology/sst-superflash-technology.

[20] I. Kataeva, F. Merrikh-Bayat, E. Zamanidoost, and D. Strukov, "Efficient training algorithms for neural networks based on memristive crossbar circuits," in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2015.