Resistive Memory Device Requirements for a Neural Algorithm Accelerator

Sapan Agarwal, Steven J. Plimpton, David R. Hughart, Alexander H. Hsia, Isaac Richter*, Jonathan A Cox, Conrad D. James, Matthew J. Marinella

Sandia National Laboratories, Albuquerque, NM, USA

***Abstract*— Resistive memories enable dramatic energy reductions for neural algorithms. We propose a general purpose neural architecture that can accelerate many different algorithms and determine the device properties that will be needed to run backpropagation on the neural architecture. To maintain high accuracy, the read noise standard deviation should be less than 5% of the weight range. The write noise standard deviation should be less than 0.4% of the weight range and up to 300% of a characteristic update (for the datasets tested). Asymmetric nonlinearities in the change in conductance vs pulse cause weight decay and significantly reduce the accuracy, while moderate symmetric nonlinearities do not have an effect. In order to allow for parallel reads and writes the write current should be less than 100 nA as well.**

Keywords*— *resistive memory, memristor, backpropagation, neural networks, noise, neuromorphic computing

I. Introduction

Resistive memory crossbars can dramatically reduce the energy required to perform computations in neural algorithms by at least six orders of magnitude when compared to a conventional CPU[2]. For data intensive applications, the computational energy is dominated by moving data between the processor, SRAM (static random access memory), and DRAM (dynamic random access memory)[3]. New approaches based on memristor or resistive memory [4-7] crossbars can enable the processing of large amounts of data by significantly reducing data movement, taking advantage of analog operations [8-12], and fitting more memory on a single chip.

Resistive memories are essentially programmable two terminal resistors. If a write voltage is applied to the device, the resistance will increase or decrease based on the sign of the voltage, allowing the resistance to be programmed. At lower voltages, the state does not change. Consequently, these devices can be used to model a neural synapse wherein the resistance acts like a weight that modulates the voltage applied to it. This has resulted in a large interest in developing neuromorphic systems based on such devices [8-11]. Ideally, the resistive memories would have a perfectly linear and controllable response allowing them to be programmed to any arbitrary analog value. Unfortunately, realistic devices have three key non-idealities: 1) read noise which causes the value read from the resistive memory to be different than the true value, 2) write noise which causes the value written to be different from the intended value, and 3) write nonlinearities which means that the change in conductance due to a write pulse will be different depending on the device's current state. This is true for most resistive memory devices, including metal oxide ReRAM, CBRAM, PCM and others. In this paper, we model the resilience of neural algorithms based on machine learning in the presence of device based noise and variability.

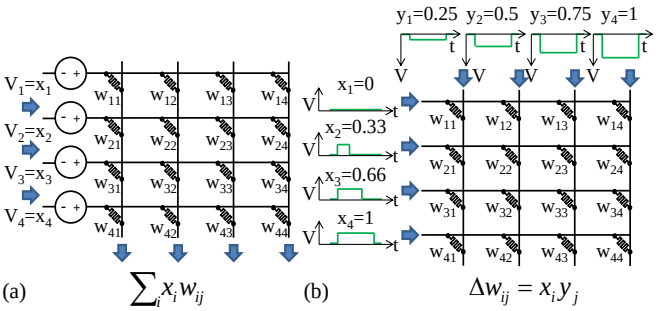

Fig. 1. (a) Analog resistive memories can be used to reduce the energy of a vector-matrix multiply. The conductance of each resistive memory device represents a matrix element or weight. Analog input vector values are represented by the input voltages or input pulse lengths, and output vector values are represented by currents. This allows all the read operations, multiplication operations and sum operations to occur in a single step. A conventional architecture must perform these operations sequentially for each weight resulting in a higher energy and delay. A matrix-vector multiply can also be performed by driving the columns and reading the currents on the rows. (b) A parallel write is illustrated. Weight W_{ij} is updated by $x_i \times y_j$. In order to achieve a multiplicative effect the x_i are encoded in time while the y_j are encoded in the height of a voltage pulse. The resistive memory will only train when x_i is nonzero. The height of y_j determines the strength of training when x_i is nonzero. The column inputs y_j can also be encoded in time as in [1]

The key contribution of this paper is to determine an acceptable range of device operating parameters in order to guide the development of new resistive memory device technologies. Consequently, we only focus on the three aforementioned non-idealities introduced by individual resistive memory devices themselves, excluding other non-idealities that would be present in a full system. Other studies have examined some device non-idealities, but none have systematically analyzed all three effects [13-15]. Analyzing all the non-idealities in a full neuromorphic system will be the subject of future work.

In the next sections, we describe a general purpose neural architecture that can be used to accelerate many neural algorithms. Backpropagation was chosen for this study as it is a computation-intensive algorithm that underlies the

This work was supported by Sandia National Laboratories' Laboratory Directed Research and Development (LDRD) Program under the Hardware Acceleration of Adaptive Neural Algorithms (HAANA) Grand Challenge. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U. S. Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000

*Isaac Richter is presently with the University of Rochester, Rochester, NY

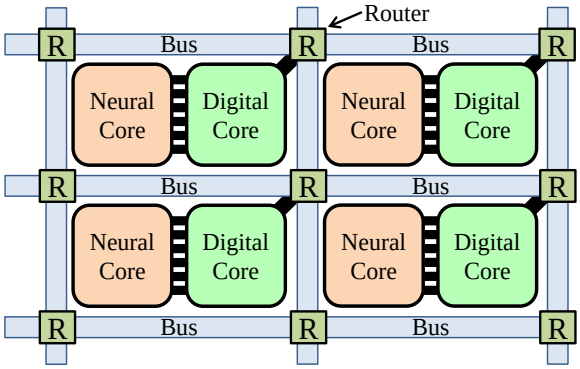

Fig. 2. A general purpose neural architecture is shown. The neural cores only perform parallel vector matrix multiplication, matrix vector multiplication and parallel rank 1 outerproduct updates. For a NxN crossbar the neural core performs $O(N^2)$ operations. The general purpose digital cores process the O(N) inputs/outputs to the neural cores and use the routing network to route data between cores. The flexibility of the digital cores allow for many different algorithms to be implemented, while still taking advantage of the neural cores to accelerate $O(N^2)$ operations. The digital cores can also use digital on-chip resistive memory instruction caches to store slowly changing data while reserving expensive SRAM caches only for the data being processed.

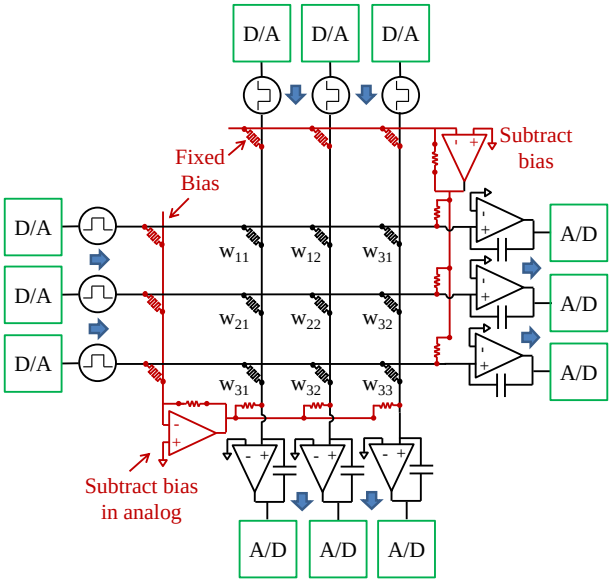

Fig. 3. A neural core is illustrated. A bias row and column are added to the crossbar to allow for negative weights [17]. The rows and columns are driven by either variable length or variable height pulses. The output currents are integrated and then converted to digital using an A/D.

successful application of neural networks in software and hardware, and has been thoroughly benchmarked in previous work[16, 17]. We use a numerical simulation, written in Python, to model how backpropagation performance is impacted by the three hardware non-idealities for different data sets, and determine device properties required to maintain high learning accuracy.

II. General Purpose Neural Architecture

The key design considerations for an effective neural algorithm accelerator is that it should both reduce the computation energy by orders of magnitude, and it should be flexible enough that it can run many different neural algorithms. In [12] it is shown that a resistive memory crossbar can accelerate two key operations: 1) a parallel read, or vector matrix multiply, and 2) a parallel write or rank one

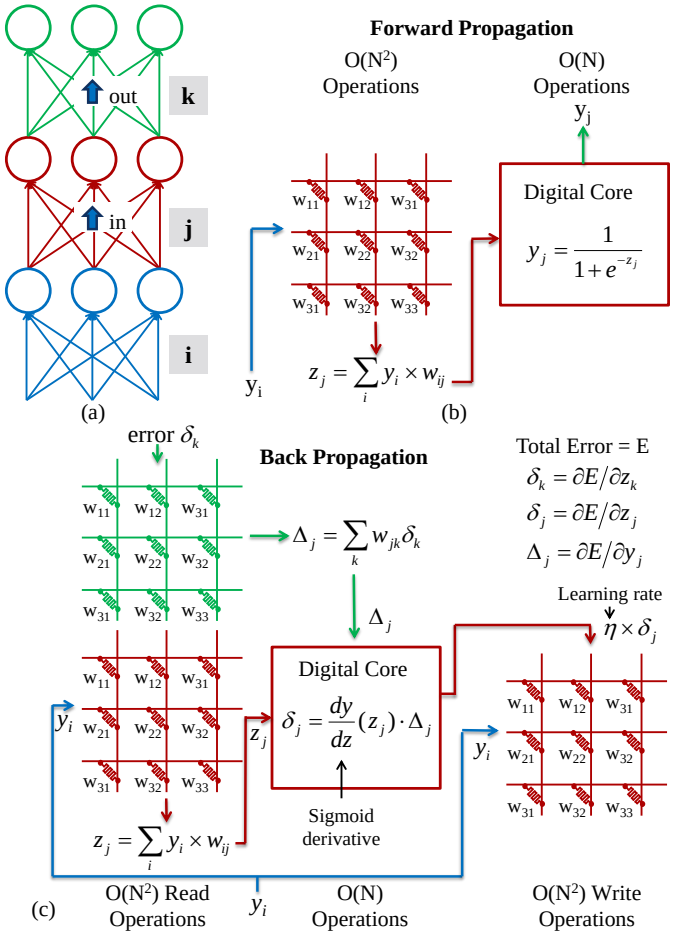

Fig. 4. A mapping of backpropogation to the neural architecture is shown. (a) A simple network (b) In a forward evaluation of a neural network, the vector matrix multiply is done in a neural core and the digital core is used to compute the sigmoid neuron function. (c) The steps required to update the middle layer weights, w_{ij} are shown. The red w_{ij} crossbar is shown twice to illustrate the two operations that need to be performed.

outer product update, as illustrated in Fig. 1. Many neural algorithms such as sparse coding, restricted Boltzmann machines, and backpropagation rely heavily on these two operations. The difference in the implementation of these algorithms is how the inputs and outputs of a crossbar are processed. An NxN crossbar accelerates $O(N^2)$ operations, while it has O(N) inputs or outputs. This means that the energy to process an input or output can cost O(N) times more than the energy to read or write a single resistive memory element without significantly increasing the system energy. This key insight allows us to optimize the tradeoff between energy efficiency and system flexibility. A crossbar based neural core should be used to perform the parallel vector matrix multiply and outer product update, while a more general purpose digital core can be used to process the inputs and outputs of the crossbar. This is illustrated in Fig. 2. The neural core is illustrated in Fig. 3.

To represent both positive and negative matrix values with a resistive device, a reference weight is subtracted in analog following [18]. Alternatively, it is possible to take the difference between two resistive memory elements, but this requires twice as many devices [13]. The inputs and outputs to the neural cores are digital and so analog to digital (A/D) and digital to analog (D/A) converters will be needed. This is energetically expensive, but they are O(N) operations and

TABLE I. DATA SET PROPERTIES

Data set	# Training Examples	# Test Examples	Network Size
UCI Small images [21]	3,823	1,797	64×36×10
File types [22]	4,501	900	256×512×9
MNIST large images [23]	60,000	10,000	784×300×10

TABLE II. WEIGHT CLIPPING LIMITS

Data set	1st Layer Weight σ	2nd Layer Weight σ	Normalized Clipping Range
UCI small images	0.87	1.93	1.5
File types	0.22	0.44	1.5
MNIST large images	0.22	1.05	1.5

can therefore be the same order of magnitude as the energy needed to drive the crossbar [12]. If desired, additional energy efficiency can be traded off against algorithmic flexibility by performing analog neuron operations in the neural core, as in [10]. The implementation of backpropagation on the general purpose neural architecture is illustrated in Fig. 4. The crossbars process $O(N^2)$ operations while the digital cores handle O(N) operations.

Data communication between cores should be built on an address event representation (AER) based spiking communication model [19, 20]. This limits data to only be sent when needed and allows a shared communication bus to be used. Using a routing network also allows arbitrary connections between the cores. We note the brain has an extremely dense connectivity where individual neurons in the cerebral cortex can receive roughly 10,000 input synapses from other neurons[21]. To achieve this, the brain takes full advantage of its 3D structure to minimize connection lengths. Currently, high performance CMOS is 2D or at most 2.5D and so it is impossible to hardwire the same number of connections. Consequently, a shared bus is needed to emulate the same connection density.

Overall, to obtain maximum energy efficiency, this type of a system assumes that a neural network has dense local connections that can be mapped to a crossbar and fewer global connections that need to be sent over the routing network. Computation is localized to the maximum extent possible, which minimizes the amount of high energy cost, longer range communications that are required. A dense local and sparse global connectivity is similar to how the brain is organized. If this is not the case for a given algorithm, a single column of a crossbar can be used in a specific read or write step to allow for maximum flexibility at higher energy cost.

III. IDEAL WEIGHT RANGES AND LEARNING RATE

To efficiently map a backpropagation network to hardware several algorithmic parameters need to be set. For a given dataset, we need the learning rate, number of epochs to train for, random weight initialization, sigmoid slope, and network size. Physical resistive memories also have a min and a max conductance. This means that the network has a min and max weight it can store. Choosing the weight range correctly is important to maximize the usage of the resistive memory's dynamic range.

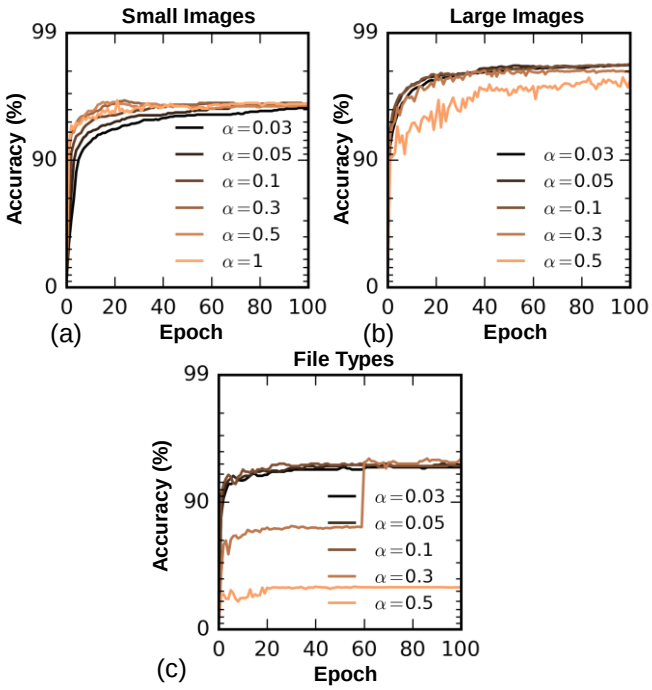

Fig. 5: The error on the test set vs epoch as a function of learning rate, α, is plotted for the different data sets.

In this paper we analyze three different data sets as summarized in Table I. First we consider a small image version (8x8 pixels) of handwritten digits from the "Optical Recognition of Handwritten Digits" dataset from [22]. Next we use MNIST, a large image version (28x28 pixels) of handwritten digits [23]. Finally, we use a Sandia file classification dataset in [24] (256 byte-pair statistical attributes to classify 9 file types). We use a simple two-layer network (one hidden layer) for each dataset with the network size indicated in Table I. For simplicity, we consider a single network configuration; more generally there may be a optimizable tradeoff between network size and noise.

We arbitrarily chose to use sigmoid neurons with a unity slope. Through the learning process the weights will be scaled up or down to match the sigmoid slope. The initial random weight range and learning rate need to be correctly chosen to enable optimal training. Following [25], the weights should be initialized to a uniform distribution of Uniform [$-r$, r] where:

$$r = 4\sqrt{6/(\mathit{fan\ in} + \mathit{fan\ out})} \qquad (1)$$

The fan-in is the number of inputs to a layer and fan-out is the number of outputs (For the first layer of MNIST the fan-in=784 and fan-out=300 from Table I).

This sets an initial scale for the weights. Next we determine the learning rate and number of epochs needed to train. These could be dataset dependent, so we test several different learning rates for each dataset as illustrated in Fig. 5. (All the training plots in this paper are based on accuracy for the test data set, after training on the training data set.) For all three datasets, 100 epochs and a learning rate of 0.1 works well. These values are used in the rest of the paper.

Next, we determine how to best map the weights to the device's conductance state. To do this we need to understand what range the weights ideally take in a noise-free,

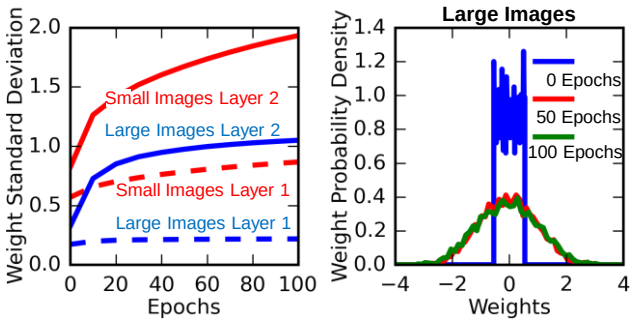

Fig. 6: (a) The standard deviation of the weights vs the training epoch is plotted. After an initial increase, the weights start to saturate. (b) Histograms of the weights in the second layer for large images are plotted vs epoch.

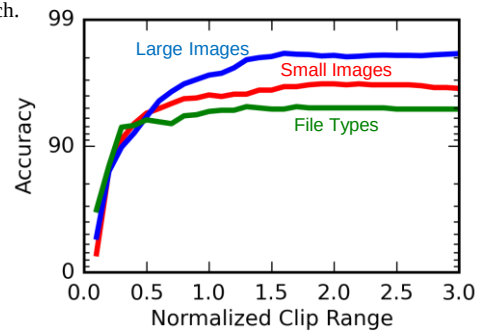

Fig. 7: The final accuracy after 100 epochs vs the normalized clipping range for the three different data sets. The clipping range for each layer is normalized to the standard deviation of the unclipped weights given in Table II.

unbounded model, and what happens if we limit the possible weight range. Fig. 6(a) shows how the standard deviation of the weights grow vs epoch and Fig. 6(b) plots a histogram of the weights for different epochs. The weights rapidly grow in the first few epochs and then start to saturate. The weight range is also different for each layer. The standard deviation of the weights in each layer after training indicates a natural weight range. This is summarized in Table II.

Since the physical devices have a limited conductance range, a limited algorithmic weight range is required. We want the smallest possible range that will not decrease the accuracy. A smaller weight range allows more of the resistive memory's dynamic range to be used, minimizing the impact of noise. The impact of training a neural network with a limited weight range is plotted in Fig. 7. This plot indicates that the weights can be clipped to around 1.5×σ without losing significant final accuracy. As seen in Fig. 8, clipping the weights causes the larger weights to saturate at the limits. The more aggressive clip range of 1.5 allows some weights to saturate, and maximizes the use of the numerical dynamic range. This minimizes the impact of noise caused by a real device and maximizes the information stored in a particular device. When the ideal weight range is not known *a priori* a larger range may be used, at the cost of increased impact from noise.

With a reasonable algorithmic weight range, we can map the ranges from Table II to physical conductance. Consider a normalized conductance scale where the maximum conductance is 1. The minimum normalized conductance will be given by 1/(on-off ratio)=G_{OFF}/G_{ON}. If G_{ON}/G_{OFF}=10, the minimum normalized conductance will be 0.1. This normalized 0.1 to 1 range needs to be mapped to the weight range required by the algorithm. Physically, a fixed bias of

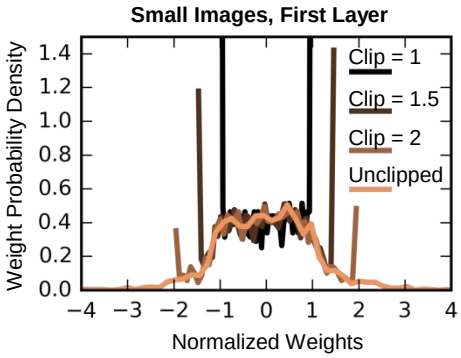

Fig. 8: A histogram of the weight distributions for different normalized clipping ranges is shown for the small image dataset. As clipping range decreases, more weights saturate at the limit.

0.55 is subtracted from each weight as illustrated in Fig. 3. This gives weights in the range of [-0.45, 0.45]. Next, this is scaled up or down to match the algorithmic weight scale. In the digital cores in Fig. 2, the digitized results from the neural core can be multiplied by an appropriate scale factor. In the rest of this paper, when possible, we define the device models relative to the total weight range so that all the results are independent of the scaling. For example, we define noise parameters as a percentage of the total weight range.

IV. Read Noise

When reading a resistive memory there are three kinds of read noise that can change the current: thermal noise, 1/f noise and random telegraph noise (RTN) [26-32]. RTN, the noise from a single trap filling/emptying, is typically the dominant form of noise. It can depend on a few particular traps causing the current to oscillate between two states. Nevertheless, between write cycles, the distribution of the relevant traps will change and the conductance change due to those traps follows a Gaussian[26]. Consequently, we consider two models, 1) a RTN model where the conductance randomly toggles between one of two states, and 2) a simple Gaussian noise distribution. The Gaussian model is a more generic model that can also approximate the effects of thermal noise and 1/f noise. For both models, the impact of the noise, averaged over an entire read cycle, can simply be modelled as a perturbation to the device's conductance during each read. For the RTN model with a standard deviation, σ, we randomly add $+\sigma$ or $-\sigma$ to the conductance where σ is defined relative to the total conductance range:

$$G = G_o \pm \sigma, \quad \sigma = \sigma_{RN} \times G_{range} \tag{2}$$

where G is the conductance after applying read noise, G_o is the actual conductance stored in the device, σ is the standard deviation, and σ_{RN} is the dimensionless standard deviation of the noise normalized to the range of the conductance, G_{range}.

For the Gaussian model, the noise is defined by:

$$G = G_o + N(\sigma), \quad \sigma = \sigma_{RN} \times G_{range} \tag{3}$$

where N is a normal distribution with standard deviation σ.

The classification accuracy of the two models is compared in Fig. 9. The two models give nearly identical results. This is because the noise from many resistive memories is added together during a vector matrix multiply.

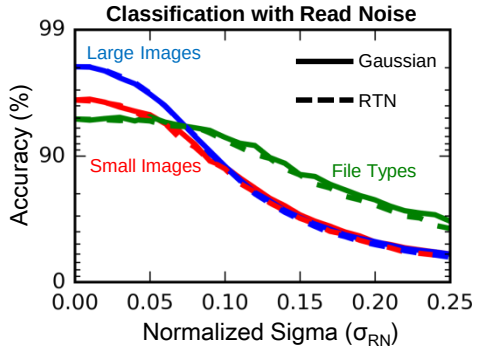

Fig. 9: Classification accuracy vs read noise for the RTN (Eq. 2) and Gaussian (Eq. 3) models give nearly identical results. The weights are pre-trained with no noise and read noise is added during classification of the test set. The noise is applied to both the weights and the fixed biases.

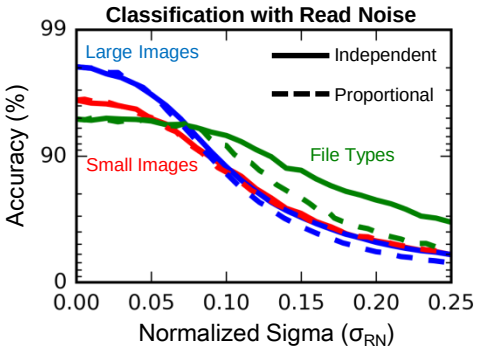

Fig. 10: The classification accuracy vs read noise is shown for the independent Gaussian noise model, Eq. (3), (solid lines) and the proportional Gaussian noise model, Eq. (4) (dotted lines). Using a normalization factor γ = 1.8 gives similar results for the two models. The small differences are due to the fact that the weights do not follow a perfectly uniform distribution. The weights are pre-trained with no noise and noise is only added during classification.

By the central limit theorem, the noise sources will add together to form identical Gaussian distributions so long as the variance of the noise is the same. Consequently, for the following simulations it is sufficient to model RTN with Eq. (3) the Gaussian noise distribution.

In some situations σ can be a function of the device's current state, G_o. For instance in [26], σ is a constant for currents >1μA and then decreases as the current decreases below 1μA. Therefore, we also consider a proportional Gaussian model where σ is also directly proportional to the current state, G_o:

$$G = G_o + N(\sigma), \quad \sigma = \gamma \times \sigma_{RN} \times G_o \tag{4}$$

Here γ is a normalization constant. If we choose γ so that both models have the same variance, they give nearly identical results, as illustrated in Figs. 10 & 11. This means that Eq. (3) is sufficient. In general, we propose Eq. (3) can be used to approximate any read noise distribution if the standard deviation in Eq. (3) is calibrated to give the same variance as a more complicated noise model.

To find γ we can approximate the clipped weight distribution as a uniform distribution over the weight range, and compare the variance between Eq. (3) and (4):

$$\int_{G_{\min}}^{G_{\max}} {\sigma_{RN}}^2 dG_o = \int_{G_{\min}}^{G_{\max}} \left(\gamma \times \sigma_{RN} \times G_o\right)^2 dG_o \tag{5}$$

Solving for γ gives:

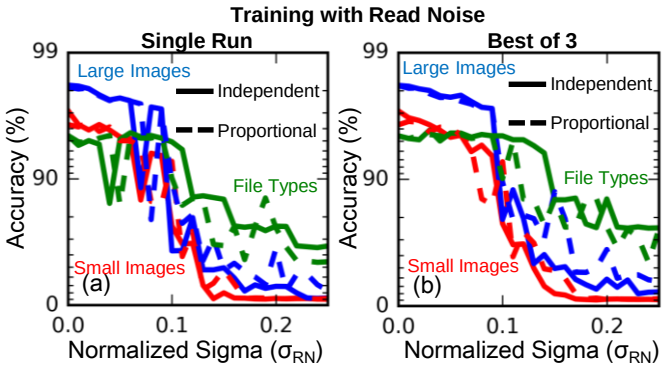

Fig. 11: (a) The classification accuracy on test data vs read noise is shown. The weights are trained with read noise resulting in a more variability between runs due to the noise. (b) The best accuracy out of three runs is shown, eliminating much of the variability. (solid lines = independent noise model Eq. (3), dotted lines = proportional noise model, Eq. (4)).

$$\gamma = \sqrt{\frac{3}{G_{\max} - G_{\min}} \times \frac{1}{G_{\max}^3 - G_{\min}^3}} \tag{6}$$

As seen from Fig. 8, a uniform weight distribution is a reasonable approximation to first order. Using G_{max}=1 and G_{min} =0.1 gives γ = 1.8.

In Fig. 10, we show how the read noise affects classification accuracy using a set of pre-trained weights (noise was not present during training). The accuracy starts to drop off after σ_{RN} = 5% of the weight range. In Fig. 11(a), we show a similar accuracy if we also apply the read noise during training. Now each data point corresponds to the final test set accuracy after a full training run of 100 epochs. In this case, the noise causes a significant variability in the final accuracy between runs which vary only in the initial random number seed. Training a network three times and taking the best result reduces the variability as shown in Fig. 11(b). Using a fixed random number seed for both the weight initialization and the noise also eliminates the variability.

V. Write Noise

Write noise occurs every time the state of a resistive memory changes [13, 33, 34]. The relative magnitude of this write noise is typically greater than that of read noise. Writing a resistive memory involves moving atoms around, which is inherently a stochastic process. The exact nature of the noise will be strongly dependent on the type of resistive memory and whether it is filamentary or non-filamentary. In general, the statistics of this type of noise are not well characterized. One of the most relevant measurements is reported in [13], where a write pulse is applied to a phase change memory at a given conductance and the change in conductance, ΔG, is measured multiple times to obtain a distribution. This work indicates that the noise increases as both ΔG increases and as the initial state, G_o, increases.

Since write noise is not well characterized experimentally, we consider the impact of two different models. First consider a write noise that is independent of the intended state change, ΔG. Since the noise is independent of ΔG, long and short write pulses will have the same noise distribution. Consequently, noise of this type would still have the same distribution after multiple write pulses. This means that the noise only affects the value that is read, and not the

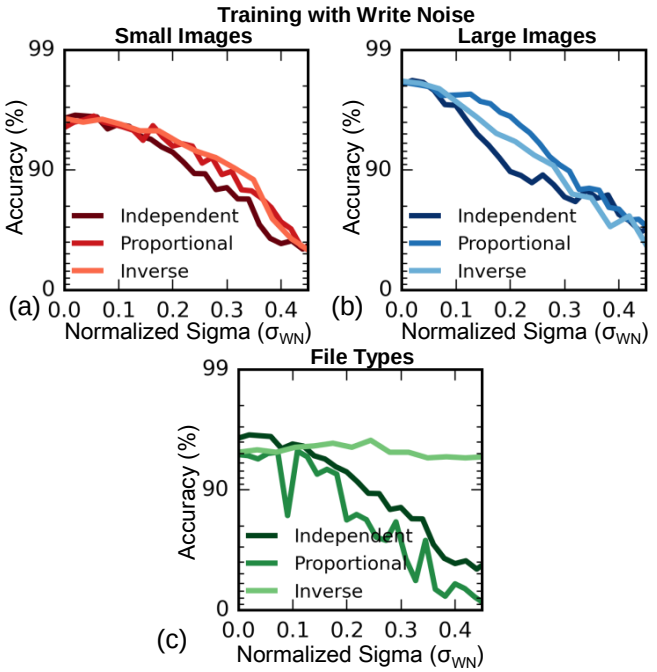

Fig. 12: The accuracy on the test data after training with the three different write noise models is plotted for the different datasets.

internal state. If the internal state has changed, the noise would compound for each step and would depend on ΔG. Effectively, this can be modelled as a read noise and a separate write noise model is not needed. Since backpropagation alternates read and write cycles and the noise is summed over many devices during a vector matrix multiply, the read noise model in the previous section is sufficient to model this type of noise.

Next, consider noise that depends on the size of the update. Here, we make an important simplifying assumption that if we use one pulse or two pulses to change the conductance from one state to another, the noise will be the same as long as the average initial and final states are the same. This implies that one longer or higher voltage pulse will cause the same physical change as two sequential pulses that end at the same average conductance. Implicitly, we are assuming that the resistive memory can be modeled with a single internal state variable so that it does not matter how we end up at a given conductance state, which is typical for these devices [35-37]. Certain devices under particular operating conditions require two internal state variables to be modelled; this is beyond the scope of this work[38].

Since the noise is the same for a given ΔG regardless of the number of pulses required to get a given ΔG, ΔG must be proportional to the variance of the noise, σ^2. After multiple pulses, the variance of the noise in each pulse is additive. Therefore, $\sigma \propto \sqrt{\Delta G}$. We also assume that the write noise follows a Gaussian distribution as it is the result of the collective motion of many atoms so that the state after an update is given by:

$$G = G_o + \Delta G + N(\sigma) \tag{7}$$

Like the read noise, it is possible that σ depends on the initial state, G_o, (and/or the final state. For simplicity, we consider smaller updates so that we only need to consider the initial or only the final state dependence). To understand the

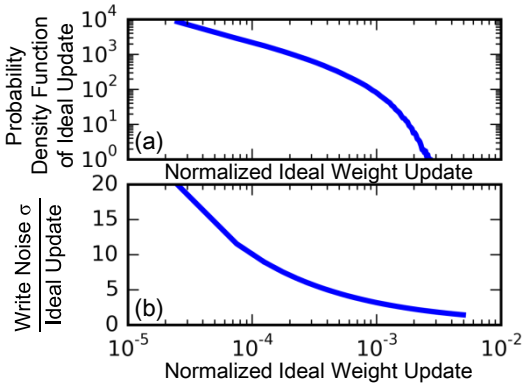

Fig. 13: (a) The probability density of the ideal updates without noise is plotted for the first epoch and second layer of the small image dataset. The update size (x-axis) is normalized to the weight range. (b) The ratio of the standard deviation of the write noise to the update size is plotted. This noise is 2-20X larger than the ideal update itself. This relatively large level of noise does not decrease the training accuracy of backpropogation. The independent guassian noise model, Eq. (8), with a σ_{WN} of 0.1 is used.

impact of the initial state dependence, we consider three possible models. First, σ is independent of G_o:

$$\sigma = \sqrt{\Delta G \times G_{range}} \times \sigma_{WN} \tag{8}$$

Here we add the conductance range, G_{range}, so that a dimensionless standard deviation, σ_{WN}, can be defined. Next we consider a σ that is proportional to G_o.

$$\sigma = \sqrt{\Delta G \times G_{range}} \times \gamma \times \frac{G_o}{G_{range}} \times \sigma_{WN} \tag{9}$$

As with the read noise, γ is chosen so that both models have the same variance. Therefore, γ is given by Eq. 6 and is 1.8 for a uniform weight distribution with G_{max}=1 and G_{min} =0.1. Lastly, we consider σ inversely proportional to G_o:

$$\sigma = \sqrt{\Delta G \times G_{range}} \times \gamma \times \frac{G_{range}}{G_o} \times \sigma_{WN} \tag{10}$$

Once again γ is chosen so that all models have the same variance. For uniform weight distribution with G_{max}=1 and G_{min} =0.1, γ is 0.35.

The effect of the three different models of write noise is plotted in Fig. 12. Again the final accuracy after training for 100 epochs is plotted. We see that the particular G_o dependence does not have a significant effect on the write noise for the small and large images, but matters somewhat for the cyber dataset. To first order, the simplest model independent of G_o, Eq. 8, gives a reasonable intuition of how the system responds when the exact G_o dependence is not known. The key is that the noise model should have the correct noise variance.

In order to compare the magnitude of updates with the corresponding write noise, we plot the probability density of the updates without noise and the corresponding noise sigma in Fig. 13 for small images. We choose a safe noise sigma, σ_{WN}, of 0.1 that does not affect the accuracy as shown in Fig. 12. (Only the second layer of the network is shown for simplicity, but results are nearly identical for the first layer.) The updates are on the order of 0.01% to 0.5% of the weight range. We can define a characteristic update size as a weighted average of the update size, weighted by the update size itself. This captures the fact that it requires ten 0.1%

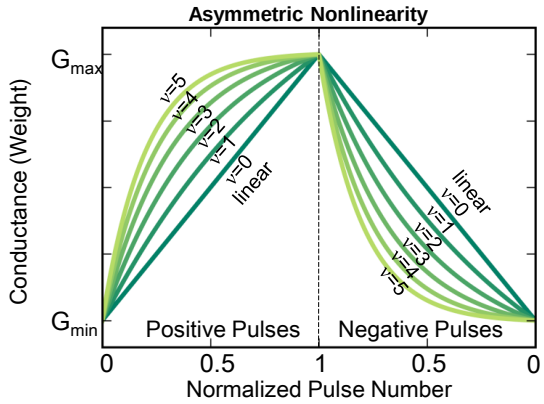

Fig. 14: Asymmetric write nonlinearity is illustrated. As the nonlinearity, α, increases the amount written depends strongly on the current state. The conductance rapidly changes at low conductance and then saturates at higher conductance for positive pulses; the converse occurs for negative pulses. The x axis is normalized by the number of pulses needed to go from the minimum to the maximum conductance.

updates to train as much as a single 1% update. The characteristic update sizes are 0.07%, 0.18%, and 0.17% for the small image, large image and file type datasets respectively. For the characteristic update and $\sigma_{WN} = 0.1$, the noise is 0.3% to 0.4% of the total range. Surprisingly, this safe noise level is 2.4X to 3.8X larger than the characteristic update itself! For smaller updates, the noise can be more than 20X the size of the update, indicating that smaller updates likely do not contribute as much to the overall learning.

VI. Write Nonlinearity

A. Asymmetric Nonlinearity

In addition to write noise between cycles, the physics of resistance change in resistive memories typically causes the conductance change to depend on the resistive memory's current state [14]. Often this nonlinearity is asymmetric with regard to the direction of the pulse. For example, near the maximum conductance a given pulse will not significantly increase the conductance, but it can significantly decrease the conductance. This is particularly true for filamentary devices, due to a thermal-runaway effect. In order to maximize efficiency, a parallel open loop write scheme must be used, and therefore we do not know each individual resistive memory's current state between training examples. This means that the same sized pulse must be applied regardless of the device's state and the nonlinear response thus introduces an additional "error" in the write. Following [14], the conductance, G, as a function of the normalized pulse number, p, for increasing pulses is modeled by:

$$G = G_1\left(1 - e^{-\nu P}\right) + G_{\min} \tag{11}$$

$$\text{where } G_1 = \frac{G_{\max} - G_{\min}}{1 - e^{-\nu}} \tag{12}$$

G_{min} is the minimum conductance, G_{max} is the maximum conductance and ν is a parameter characterizing the nonlinearity. When ν=0, the response is perfectly linear. Experimental devices have been demonstrated with $\nu \approx 2$ - 5 [14]. If we have a target update, ΔG_{target}, using Eq. (11) we can solve for the actual update:

$$\Delta G = \left(G_1 + G_{\min} - G\right)\left(1 - e^{-\nu\delta}\right) \tag{13}$$

where the normalized target update is given by:

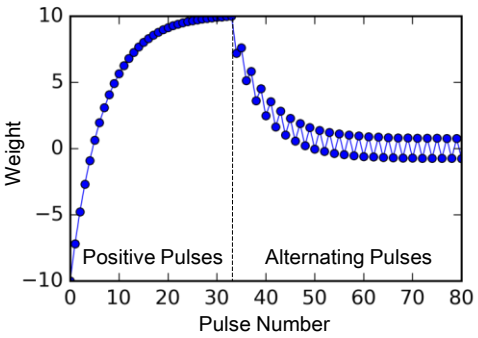

Fig. 15: Applying identical alternating positive and negative pulses causes the weight to decay towards a center value when it should remain constant. When the weight is near the maximum, a positive pulse does not change the weight much, but a negative pulse significantly decreases it. The opposite holds for weights near the minimum weight.

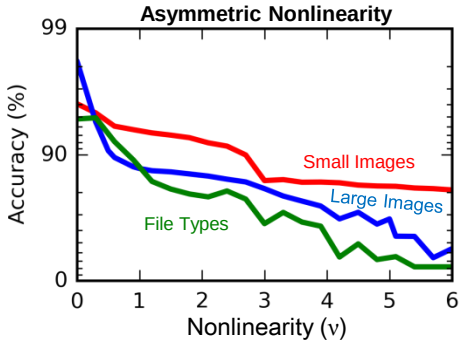

Fig. 16: The impact of the asymmetric write nonlinearity on learning is illustrated for all three datasets.

$$\delta = \Delta G_{\text{target}} / \left(G_{\max} - G_{\min}\right) \tag{14}$$

For decreasing pulses, the conductance is given by:

$$G = G_{\max} - G_1\left(1 - e^{-\nu(1-P)}\right) \tag{15}$$

and the actual update given the target update is:

$$\Delta G = -\left(G + G_1 - G_{\max}\right)\left(1 - e^{\nu\delta}\right) \tag{16}$$

The asymmetric nonlinearity model, Eqs. (11) and (15) are plotted in Fig. 14. A strong asymmetric nonlinearity causes the conductance to decay towards a center value after alternating pulses as illustrated in Fig. 15. A small amount of weight decay can be beneficial to prevent overfitting, but typically the decay will be too large, degrading the ability of devices to "learn" the weights needed for the backpropagation algorithm, reducing its final accuracy. This is illustrated in Fig. 16. The accuracy (after 100 epochs) vs nonlinearity is plotted for the three different data sets.

B. Symmetric Nonlinearity

Some devices exhibit a symmetric nonlinearity. This is demonstrated in [39] for a Ag/GeSe/Pt CBRAM cell, and the data is replotted on a conductance axis in Fig. 17. Resistive memories that have a non-filamentary switching mechanism are also expected to behave with a symmetric switching response, although this has not yet been explicitly demonstrated in the literature at this time. The symmetry is more likely because the conductance modulation is dependent on the motion of many atoms, rather than a few critical atoms in a filament. To understand the impact of a symmetric non-linearity, we consider a simple sigmoid based model illustrated in Fig. 18. We assume that after a sufficient

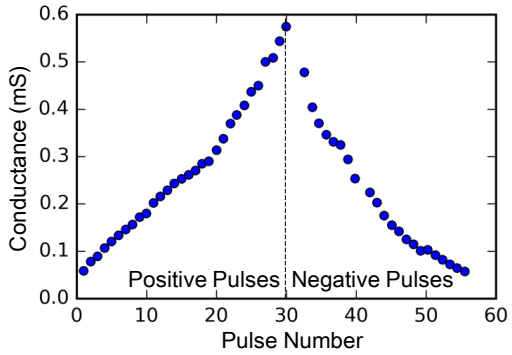

Fig. 17: The conductance vs pulse response for a Ag-GeSe cell if plotted from [39]. This device shows a nearly symmetric response when switching from positive to negative pulses.

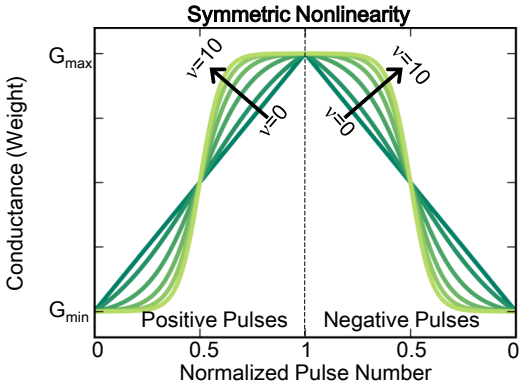

Fig. 18: The symmetric write nonlinearity model is illustrated. The x-axis is normalized by the number of pulses needed to go from the minimum to the maximum weight. In contrast to Fig. 14, the response is reversible for positive versus negative pulses

number of pulses the device will saturate at a maximum or minimum conductance. (The pulsing measurement in Fig. 17 was likely stopped before the conductance started to saturate). The conductance, G, as a function of the normalized pulse number, p is given by:

$$G = A \times \frac{1}{1+e^{-2\nu(p-0.5)}} + B \tag{17}$$

where:

$$A = (G_{\max} - G_{\min}) \times \frac{e^{\nu}+1}{e^{\nu}-1} \tag{18}$$

$$B = G_{\min} - \frac{(G_{\max} - G_{\min})}{e^{\nu}-1} \tag{19}$$

G_{min} is the minimum conductance, G_{max} is the maximum conductance and ν is a parameter characterizing the nonlinearity. ν is defined such that the symmetric and asymmetric models have the same slope at the center conductance: (G_{min}+ G_{max})/2. the actual update given the target update is:

$$\Delta G = \left(A \Big/ \left[1 + e^{-2\nu\delta} \left(\frac{A}{w-B} - 1 \right) \right] \right) - w + B \tag{20}$$

where δ is defined by Eq. (14). A symmetric nonlinearity model does not suffer from the same weight decay problem as the asymmetric nonlinearity. Consequently, a much larger nonlinearity can be tolerated without decreasing the accuracy as illustrated in Fig. 19 as compared to Fig. 16.

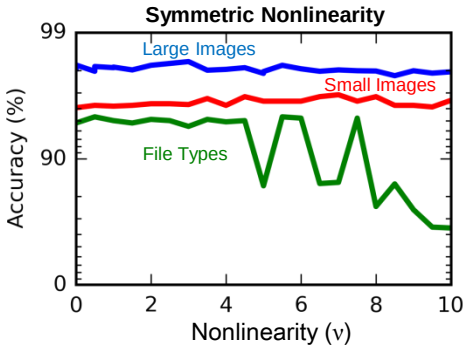

Fig. 19: The accuracy does not significantly degrade with moderate symmetric nonlinearity.

VII. Combined Non-Idealities

Finally, we compare the impact of all the non-idealities operating at the same time. In Fig. 20 we show the effect of read and write noise with different nonlinearities for the small images. Figs. 21 and 22 show the same for large images and file types respectively. Each colored "pixel" in each sub-figure represents a final accuracy after training for 100 epochs. The largest data set (MNIST) required 2-3 days of CPU time on a single core to train with all three non-idealities enabled for a single set of parameters. Because pixels represent independent runs, we used up to 1024 cores of a parallel cluster to scan multiple parameters and produce the data for Figs 20-22.

For the read noise we used a Gaussian noise model with a fixed sigma, Eq. (3). For the write noise, we used the simplest model, with noise independent of the current state, Eq. (8). As seen from the figures, adding an asymmetric nonlinearity response rapidly reduces the overall accuracy. Moderate symmetric nonlinearities do not impact the accuracy. For small images, Fig. 20d, the symmetric non-linearity actually increases the accuracy at higher levels of noise. We believe this is because the weights are nudged towards the max or min values, reducing the impact of noise.

VIII. Device Resistance

The last key device requirement to consider is the resistance required for use in a crossbar. Scaled wires at a 10nm half pitch can only handle 10 μA before electromigration becomes an issue[40]. Higher currents also cause unacceptable parasitic voltage drops [41]. In order to support a 1000x1000 crossbar with a fully parallel read or write, each device can have no more than a maximum switching current of 10nA. If we only read/write a smaller 100x100 crossbar in parallel, each device can have a switching current of 100nA. At 1V that corresponds to a resistance of 10 MΩ.

High resistance devices have been demonstrated [42, 43], but devices have not yet been demonstrated with both a high resistance and low variability, symmetric analog switching characteristics. The need for high on-state resistance and good analog characteristics means that filamentary resistive memories may not work as well as non-filamentary devices. A resistance higher than a quantum of conductance, 13 kΩ, requires current to tunnel through barrier. This presents a fundamental problem for a filamentary device: a single atom

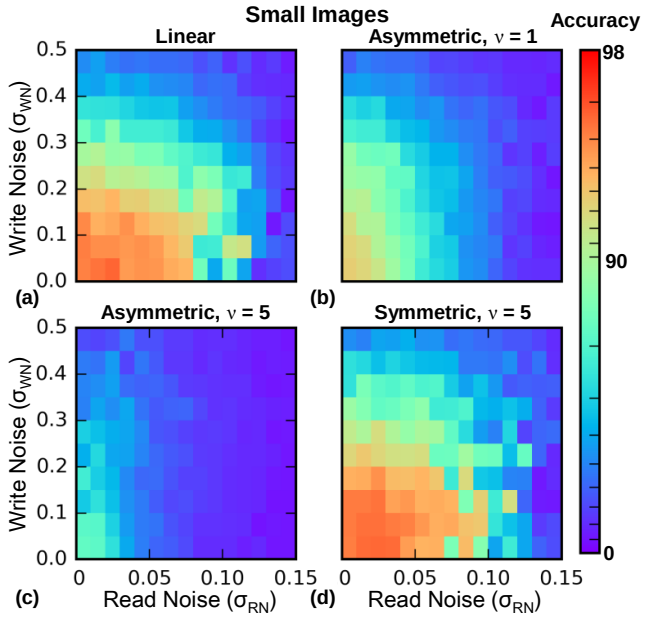

Fig. 20: The impact of read noise, write noise and nonlinearity is illustrated for small images

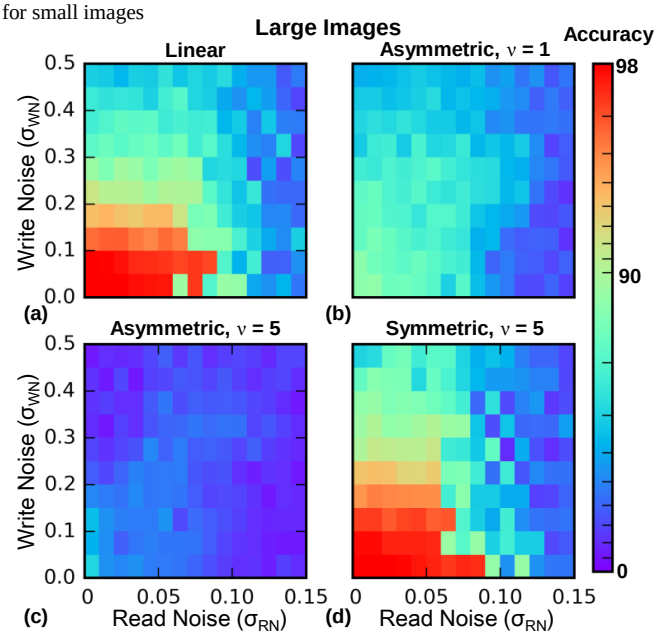

Fig. 21: Same as Fig. 20 for large images

can halve that tunneling barrier, resulting in huge variability and poor analog characteristics.

IX. Conclusion

We have introduced a general purpose neural architecture that can solve many different problems. This architecture can be used effectively to implement backpropagation with resistive memory crossbars that have the properties summarized in Table III. Our numerical modeling of 2D crossbars (matrices) of devices on three different datasets has shown that training or classifying with resistive memories with a read noise sigma up to 5% of the total conductance range does not significantly degrade the accuracy (~1%). Neural networks are also robust to write noise that is up to 0.4% of the total range and 300% of a characteristic update. This will vary slightly depending on the dataset and the

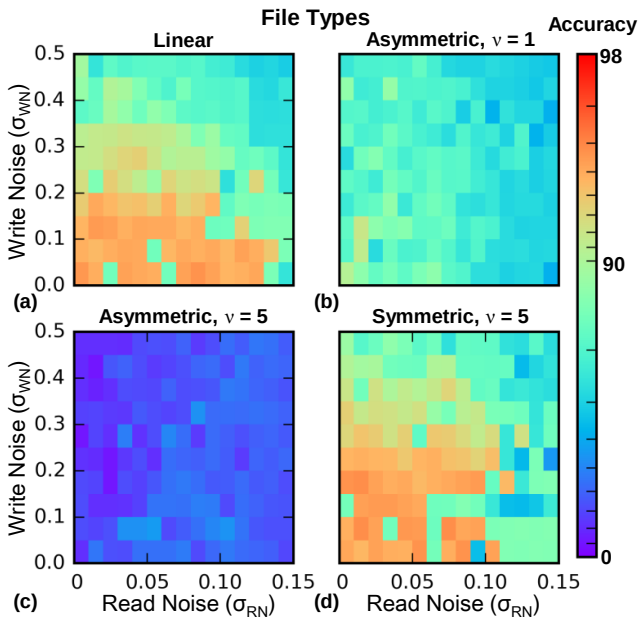

Fig. 22: Same as Fig. 20 for file types.

TABLE III. Required Device Properties

	Small Images	Large Images	File Types
Read Noise σ (% Range)	3%	5%	9%
Write Noise σ (% Range)	0.3%	0.4%	0.4%
Asymmetric Nonlinearity (ν)	0.1	0.1	0.1
Symmetric Nonlinearity (ν)	>20	5	5
Maximum Current	160 nA	13 nA	40 nA

neural network architecture. Both read and write noise can be modelled with reasonable accuracy using a simple Gaussian noise model. The simpler noise models generally match more complex models so long as the models have the same noise variance averaged over the weight distribution. The read and write noise models are physically inspired, but refined to empirically fit the available data. Asymmetric nonlinearities with ν>0.1 degrades the classification accuracy as it causes weight decay, while moderate symmetric nonlinearities with ν up to 5 do not harm the classification accuracy. The asymmetric nonlinearity model is empirically derived from device data, while the symmetric nonlinearity model is more speculative. To work in an energy efficient crossbar, resistive memories must also have a high on-state resistance of 10 MΩ or higher. Promising devices have been demonstrated experimentally, but more resistive memory development is needed to create a device that meets all of these requirements simultaneously.

X. Acknowledgements

The authors would like to thank Geoff Burr, Kevin R. Dixon, Tarek Taha, Dhireesha Kudithipudi, Shimeng Yu, Hugh Barnaby, and David J. Mountain for helpful discussions.

References

[1] D. Kadetotad, X. Zihan, A. Mohanty, *et al.*, "Parallel Architecture With Resistive Crosspoint Array for Dictionary Learning Acceleration," *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 5, pp. 194-204, 2015.

[2] T. M. Taha, R. Hasan, C. Yakopcic, *et al.*, "Exploring the design space of specialized multicore neural processors," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, 2013, pp. 1-8.

[3] P. Kogge, K. Bergman, S. Borkar, *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," 2008.

[4] D. B. Strukov, G. S. Snider, D. R. Stewart, *et al.*, "The missing memristor found," *Nature*, vol. 453, pp. 80-83, 2008.

[5] L. O. Chua, "Memristor-The missing circuit element," *Circuit Theory, IEEE Transactions on*, vol. 18, pp. 507-519, 1971.

[6] R. Waser and M. Aono, "Nanoionics-based resistive switching memories," *Nat Mater*, vol. 6, pp. 833-840, 2007.

[7] K.-H. Kim, S. Gaba, D. Wheeler, *et al.*, "A Functional Hybrid Memristor Crossbar-Array/CMOS System for Data Storage and Neuromorphic Applications," *Nano Letters*, vol. 12, pp. 389-395, 2012/01/11 2012.

[8] S. H. Jo, T. Chang, I. Ebong, *et al.*, "Nanoscale Memristor Device as Synapse in Neuromorphic Systems," *Nano Letters*, vol. 10, pp. 1297-1301, 2010/04/14 2010.

[9] C. Ting, Y. Yuchao, and L. Wei, "Building Neuromorphic Circuits with Memristive Devices," *Circuits and Systems Magazine, IEEE*, vol. 13, pp. 56-73, 2013.

[10] R. Hasan and T. M. Taha, "Enabling back propagation training of memristor crossbar neuromorphic processors," in *Neural Networks (IJCNN), 2014 International Joint Conference on*, 2014, pp. 21-28.

[11] Y. Kim, Y. Zhang, and P. Li, "A reconfigurable digital neuromorphic processor with memristive synaptic crossbar for cognitive computing," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 11, p. 38, 2015.

[12] S. Agarwal, T.-T. Quach, O. Parekh, *et al.*, "Energy Scaling Advantages of Resistive Memory Crossbar Based Computation and its Application to Sparse Coding," *Frontiers in Neuroscience*, vol. 9, 2016.

[13] G. W. Burr, R. M. Shelby, S. Sidler, *et al.*, "Experimental Demonstration and Tolerancing of a Large-Scale Neural Network (165 000 Synapses) Using Phase-Change Memory as the Synaptic Weight Element," *Electron Devices, IEEE Transactions on*, vol. 62, pp. 3498-3507, 2015.

[14] P.-Y. Chen, B. Lin, I.-T. Wang, *et al.*, "Mitigating Effects of Non-ideal Synaptic Device Characteristics for On-chip Learning," presented at the Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, Austin, TX, USA, 2015.

[15] H. Miao, L. Hai, C. Yiran, *et al.*, "Memristor Crossbar-Based Neuromorphic Computing System: A Case Study," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, pp. 1864-1878, 2014.

[16] M. Ueda, Y. Nishitani, Y. Kaneko, *et al.*, "Back-Propagation Operation for Analog Neural Network Hardware with Synapse Components Having Hysteresis Characteristics," *PLoS ONE*, vol. 9, p. e112659, 2014.

[17] S. K. Esser, R. Appuswamy, P. Merolla, *et al.*, "Backpropagation for Energy-Efficient Neuromorphic Computing," in *Advances in Neural Information Processing Systems*, 2015, pp. 1117-1125.

[18] S. N. Truong and K.-S. Min, "New memristor-based crossbar array architecture with 50-% area reduction and 48-% power saving for matrix-vector multiplication of analog neuromorphic computing," *Journal of semiconductor technology and science*, vol. 14, pp. 356-363, 2014.

[19] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, pp. 668-673, August 8, 2014 2014.

[20] P. Jongkil, T. Yu, C. Maier, *et al.*, "Live demonstration: Hierarchical Address-Event Routing architecture for reconfigurable large scale neuromorphic systems," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, 2012, pp. 707-711.

[21] A. Schüz and G. Palm, "Density of neurons and synapses in the cerebral cortex of the mouse," *The Journal of Comparative Neurology*, vol. 286, pp. 442-455, 1989.

[22] K. Bache and M. Lichman. UCI machine learning repository [Online]. Available: http://archive.ics.uci.edu/ml/

[23] Y. LeCun, C. Cortes, and C. J. Burges. The MNIST database of handwritten digits [Online]. Available: http://yann.lecun.com/exdb/mnist

[24] J. A. Cox, C. D. James, and J. B. Aimone, "A Signal Processing Approach for Cyber Data Classification with Deep Neural Networks," *Procedia Computer Science*, vol. 61, pp. 349-354, // 2015.

[25] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*, ed: Springer, 2012, pp. 437-478.

[26] D. Veksler, G. Bersuker, L. Vandelli, *et al.*, "Random telegraph noise (RTN) in scaled RRAM devices," in *Reliability Physics Symposium (IRPS), 2013 IEEE International*, 2013, pp. MY.10.1-MY.10.4.

[27] S. Ambrogio, S. Balatti, V. McCaffrey, *et al.*, "Noise-Induced Resistance Broadening in Resistive Switching Memory Part II: Array Statistics," *Electron Devices, IEEE Transactions on*, vol. 62, pp. 3812-3819, 2015.

[28] S. Ambrogio, S. Balatti, V. McCaffrey, *et al.*, "Noise-Induced Resistance Broadening in Resistive Switching Memory Part I: Intrinsic Cell Behavior," *Electron Devices, IEEE Transactions on*, vol. 62, pp. 3805-3811, 2015.

[29] Y. Shimeng, R. Jeyasingh, W. Yi, *et al.*, "Understanding the conduction and switching mechanism of metal oxide RRAM through low frequency noise and AC conductance measurement and analysis," in *Electron Devices Meeting (IEDM), 2011 IEEE International*, 2011, pp. 12.1.1-12.1.4.

[30] Z. Fang, H. Y. Yu, W. J. Fan, *et al.*, "Current Conduction Model for Oxide-Based Resistive Random Access Memory Verified by Low-Frequency Noise Analysis," *Electron Devices, IEEE Transactions on*, vol. 60, pp. 1272-1275, 2013.

[31] N. Raghavan, R. Degraeve, A. Fantini, *et al.*, "Microscopic origin of random telegraph noise fluctuations in aggressively scaled RRAM and its impact on read disturb variability," in *Reliability Physics Symposium (IRPS), 2013 IEEE International*, 2013, pp. 5E.3.1-5E.3.7.

[32] Y. H. Tseng, S. Wen Chao, H. Chia-En, *et al.*, "Electron trapping effect on the switching behavior of contact RRAM devices through random telegraph noise analysis," in *Electron Devices Meeting (IEDM), 2010 IEEE International*, 2010, pp. 28.5.1-28.5.4.

[33] X. Guan, Y. Shimeng, and H. S. P. Wong, "On the Switching Parameter Variation of Metal-Oxide RRAM Part I: Physical Modeling and Simulation Methodology," *Electron Devices, IEEE Transactions on*, vol. 59, pp. 1172-1182, 2012.

[34] S. Kim, S. Choi, J. Lee, *et al.*, "Tuning Resistive Switching Characteristics of Tantalum Oxide Memristors through Si Doping," *ACS Nano*, vol. 8, pp. 10262-10269, 2014/10/28 2014.

[35] C. Yakopcic, T. M. Taha, G. Subramanyam, *et al.*, "Generalized Memristive Device SPICE Model and its Application in Circuit Design," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, pp. 1201-1214, 2013.

[36] M. D. Pickett, D. B. Strukov, J. L. Borghetti, *et al.*, "Switching dynamics in titanium dioxide memristive devices," *Journal of Applied Physics*, vol. 106, p. 074508, 2009.

[37] S. Kvatinsky, E. G. Friedman, A. Kolodny, *et al.*, "TEAM: ThrEshold Adaptive Memristor Model," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 60, pp. 211-221, 2013.

[38] P. R. Mickel, A. J. Lohn, C. D. James, *et al.*, "Isothermal Switching and Detailed Filament Evolution in Memristive Systems," *Advanced Materials*, vol. 26, pp. 4486-4490, 2014.

[39] D. Mahalanabis, H. J. Barnaby, Y. Gonzalez-Velo, *et al.*, "Incremental resistance programming of programmable metallization cells for use as electronic synapses," *Solid-State Electronics*, vol. 100, pp. 39-44, 10// 2014.

[40] *International Technology Roadmap for Semiconductors 2013 Edition*. Available: http://www.itrs2.net

[41] P.-Y. Chen, D. Kadetotad, Z. Xu, *et al.*, "Technology-design co-optimization of resistive cross-point array for accelerating learning algorithms on chip," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, 2015, pp. 854-859.

[42] S. Gaba, C. Fuxi, Z. Jiantao, *et al.*, "Ultralow Sub-1-nA Operating Current Resistive Memory With Intrinsic Non-Linear Characteristics," *Electron Device Letters, IEEE*, vol. 35, pp. 1239-1241, 2014.

[43] C. H. Cheng, C. Y. Tsai, A. Chin, *et al.*, "High performance ultra-low energy RRAM with good retention and endurance," in *Electron Devices Meeting (IEDM), 2010 IEEE International*, 2010, pp. 19.4.1-19.4.4.