Tolerating Retention Failures in Neuromorphic Fabric based on Emerging Resistive Memories

Christopher Münch
Department of Computer Science
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
Tel: +49 721 608 47586
Fax: +49 721 608 43962
e-mail: christopher.muench@kit.edu

Rajendra Bishnoi
Computer Engineering Lab
Delft University of Technology
Delft, The Netherlands
Tel: +31 15 27 86196
Fax: +31 15 27 86196
e-mail: r.k.bishnoi@tudelft.nl

Mehdi B. Tahoori
Department of Computer Science
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
+49 721 608 47778
+49 721 608 43962
e-mail: mehdi.tahoori@kit.edu

***Abstract*—In recent years, computation is shifting from conventional high performance servers to Internet of Things (IoT) edge devices, most of which require the processing of cognitive tasks. Hence, a great effort is put in the realization of neural network (NN) edge devices and their efficiency in inferring a pre-trained Neural Network. In this paper, we evaluate the retention issues of emerging resistive memories used as non-volatile weight storage for embedded NN. We exploit the asymmetric retention behavior of Spintronic based Magnetic Tunneling Junctions (MTJs), which is also present in other resistive memories like Phase-Change memory (PCM) and ReRAM, to optimize the retention of the NN accuracy over time. We propose mixed retention cell arrays and an adapted training scheme to achieve a trade-off between array size and the reliable long-term accuracy of NNs. The results of our proposed method save up to 24% of inference accuracy of an MNIST trained Multi-Layer-Perceptron on MTJ-based crossbars.**

***Index Terms*—resistive memory, neural networks, retention**

I. Introduction

In our day-to-day life, computation devices play an increasing role. The miniaturization of circuits over the last few decades allows for small chips, which can be used nearly everywhere to bring "intelligence" to conventional electronic devices. By connecting these smart devices and using them to control one another, the Internet of Things (IoT) is created. Their ability to perform in an energy efficient way is one of the most crucial design targets of IoT edge devices. Because many of the potential applications involve cognitive tasks, embedded neuromorphic processing is a must for many IoT systems.

Neural networks allow the processing of large amounts of input data, to perform a (complex) pre-trained cognitive task in an efficient way. In embedded neuromorphic IoT systems, the training is done once and except for limited in-field updates, the trained neural network, i.e., the weights, should remain unchanged. The drawback of this approach is the need of extra non-volatile memory to store the neural network on the chip. Emerging resistive memories such as *Magnetic Tunneling Junction* (MTJ) based *Spin Transfer Torque Magnetic* RAM (STT-MRAM) [1, 2], *Phase-Change* RAM (PCRAM) [3, 4] or *Resistive* RAM (ReRAM) [5, 6] are able to provide this large quantity of storage in a dense and non-volatile way.

Nevertheless, the retention and consequently the non-volatility of these memory technologies is subject to fabrication process and design time decisions. By changing the corresponding fabrication and design parameters, there will be a trade-off between retention time from one side and the energy, latency and area from the other side [7]. Additionally, these resistive memories show an asymmetric behavior in their retention induced switching characteristic. For example, it is more likely for an MTJ in the high resistance state to flip to the low resistance state due to a retention failure than the other way around [8, 9]. These retention failures consequently have an impact on the overall accuracy of the implemented neural network and need to be considered and dealt with.

In this work, we improve the long term accuracy of binary neural networks which are mapped to non-volatile resistive crossbars. Therefore, we train and optimize the parameters of the neural network and map them to an MTJ-based crossbar for evaluation. To mitigate retention failures, our approach consists of a mixed-retention crossbar array as well as the corresponding training methodology for mixed retention MTJ crossbars. This way, we are able to optimize the in-memory computation crossbar for neuromorphic computing in terms of area and power consumption while maintaining the long term accuracy of the mapped neural network.

Our contributions in this paper are as follows:

- We have developed a model to evaluate the neural network inference degradation over time due to asymmetric retention faults of resistive memories.
- We propose the use of mixed retention resistive arrays to improve the area efficiency, while maintaining the target inference accuracy.
- We propose an adapted neural network training scheme to improve long-term reliability of binary neural networks and show its mapping to the mixed retention array.

The rest of the paper is organized as follows. In Section II we cover the needed background for our ideas and the later presented evaluation methodologies. Section III describes our retention failure simulation framework and provides architecture-level and training solutions to mitigate retention failures. Finally, Section IV concludes the paper.

II. Background

A. Binary Neural Network Training

A neural network is composed of multiple layers of neurons, connected by synapses. The first layer is used to receive the input data, which is then propagated through the hidden layers to the output layer, where the result of the network operation is provided. Depending of the complexity of the task, multiple

978-1-7281-4123-7/20/$31.00 © 2020 IEEE

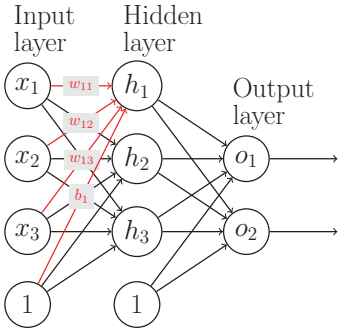


Fig. 1: Topology of a small two-layer Neural Network.

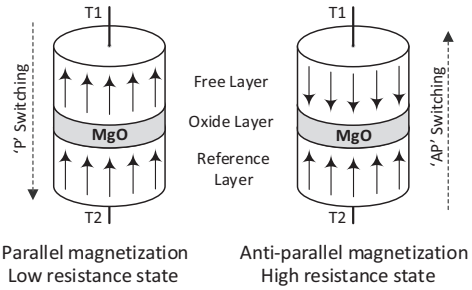


Fig. 2: Magnetic Tunnel Junction (MTJ) structure and switching current directions.

hidden layers are used to train on and infer the input data. Each neuron sums up its weighted inputs and applies an activation function to the sum to calculate its own output for the next layer. A neural network is therefore not only defined by its layers, but also by the connection between the respective neurons. In this paper, we focus on *fully connected layers*, in which each neuron of the layer is connected to all of the neurons from the previous layer (see Fig. 1).

$$J(W,b,x,y) = \frac{1}{2}\|h_{W,b}(x) - y\|^2 \tag{1}$$

During a supervised training process, the neural network is fed with training samples x which *should* produce the expected output y. The actual output of the *forward propagation* through the neural network $\hat{y} = h_{W,b}(x)$ is then used to adjust the weights during the following *back propagation*. In the case of binary neural networks, non-binary weights are used during the back propagation to allow the training. This is necessary, because the typically used *Gradient Descent Training* [10] can only be performed on proper differentiable loss-functions. These are used to evaluate the quality of a neural network. By minimizing the loss (Eq. 1), the overall accuracy of the network is improved. Therefore, to enable optimizers the use of gradient based approaches to minimize the loss, it is important to keep the weights in a non-binary form during training. Only during the forward propagation phase in the training the weights are binarized to infer the result. After the network is fully trained it is binarized [11] and mapped to resistive memory.

B. Magnetic Tunnel Junctions

An MTJ is a multi-layer structure, composed of two ferromagnetic layers separated by an insulation layer (see Fig. 2). One of the ferromagnetic layers has a fixed magnetic orientation, the reference layer (RL), the other one is free, the free layer (FL). A bidirectional write current above the device dependent critical current I_c is used to switch the magnetic orientation of FL to either the same magnetic orientation as RL, the parallel state (P-state) or to the opposite magnetization state, the anti-parallel state (AP-state). The MTJ has a low resistance when it is in the P-state and a high resistance when it is in the AP-state. This resistance can be sensed with the help of a unidirectional small read current I_r, with $I_r < I_c$. Due to thermal fluctuations, the state of the MTJ can switch randomly over time. This switching is significantly more likely to happen, if the MTJ is in the AP-state [8] as the energy barrier for AP to P switching is lower than the energy barrier for P to AP switching [9]. These retention faults are highly dependent on the thermal stability factor Δ of the MTJ.

Changing Δ also has impact on other characteristics of the MTJ like I_c. The switching probability of an MTJ after the time span t as well as the influence of the thermal stability on the probability can be modeled as

$$P(t)_{sw} = 1 - e^{-\frac{t}{\tau_0} * e^{-\Delta}} \tag{2}$$

where $\tau_0 = 1ns$ is the attempt time and Δ the thermal stability of the cell.

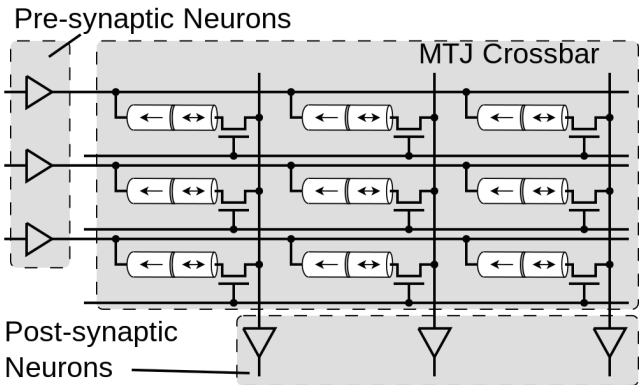


Fig. 3: Conceptual synaptic array using MTJ devices for neuromorphic computing.

C. Neuromorphic computing using resistive memories

Neuromorphic architectures typically demand a high amount of storage. Emerging Resistive memories offer several features, such as non-volatile storage, high density cells and low power consumption. The low memory footprint allows for the needed integration density. Furthermore, one of the largest use cases of these architectures is in the Internet of Things and general hand-held devices. These applications require embedded neuromorphic accelerators to store and compute pre-trained neural networks with limited in-field updates. Therefore, the non-volatility of these embedded memories results in lower footprint as no separate non-volatile storage is require to store weights. A low power consumption is therefore critical and the non-volatility further increases the usability for that matter. Additionally, the resistive nature enables new unique ways to neuromorphic computing compared to the traditional von-Neumann computing paradigm. This makes re-

sistive memories an excellent technology choice for embedded neuromorphic computing.

By using the resistive memory cells in a crossbar (see Fig. 3) an effective placement for fully connected layers can be achieved. Each row is representing a pre-synaptic neuron and each column a post-synaptic neuron. In the case of the first layer of a network, the pre-synaptic neuron behavior is dictated by the input values and for the last layer of a network the post-synaptic neurons represent the result of the neural inference. Each pre-synaptic neuron is responsible to generate the source current which is summed up on the column-line. The resistive state of each synaptic memory cell is controlling the contributing current to the total current sum sensed by the post-synaptic neuron of the column. The post-synaptic neuron on the other hand is used to evaluate the information on its column.

There are various schemes to calculate the neuron output depending on the neuromorphic concept used. The serial row access enables one row after another and sums up the result in a digital manner in the neuron. This would result in a conventional von-Neumann computing operation in line with the concepts of [11] and [12]. The main computation is therefore done in the neuron. The parallel row access enables multiple, ideally all, rows at once and evaluates on the current sum created on the column line. Here, the calculation is done in the memory itself. Such in-memory computing operation needs additional circuitry to count the activated HRS cells for evaluation of the overall result [13]. Another possibility is to use the crossbar for Spiking Neural Networks [14, 15]. A Neuron in these architectures is additionally monitoring the current flow in the column over time and generating output spikes instead of a discrete result. This can be seen as a mixture of the previous two, using in-memory multiplication and near-memory (in the neuron) summation.

D. Related Work

A thorough evaluation on inference accuracy is done in [16] for a compound RRAM synapse. They use multi-level RRAM cells to store the weights of the neural network. Due to conductance state degradation over time, it is possible that some of the neighboring conduction levels are not distinguishable. They suggest to use multiple RRAM devices per cell, so that each cell can hold additional states and declare the overlapping states as unused. This way, the sense margins are improved and distinguishable after the deterioration. An architecture-level solution by using different Error Correction Codes (ECC) is proposed in [17]. These are implemented in the memory controller and can be efficiently implemented. Additionally, the usage of ECC is not depending on the underlying memory technology. Using spintronic based binary neural networks was evaluated more generically in [18]. In this paper, the mitigation of MTJ defects was handled by increasing the redundant network cells in the resistive memory crossbar and by deactivating columns of the crossbar with additional logic based on post-manufacturing test information.

Our approach to improve the long term accuracy of a neural network stored in a resistive memory is different to these solutions, as we propose a mixture of an adapted training off-chip and a modified cell design, while keeping the rest of the system unaltered.

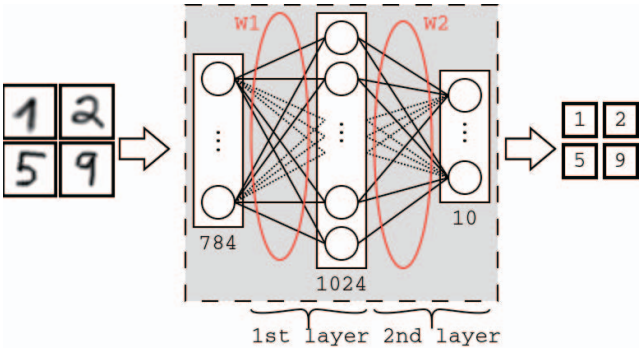


Fig. 4: Architecture and labeling of the neural network under evaluation.

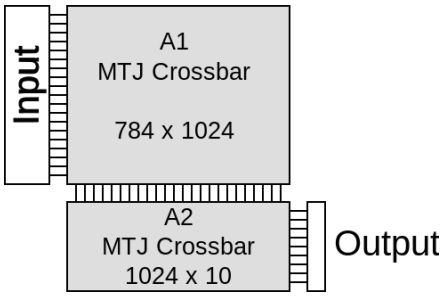


Fig. 5: Crossbar layout of the evaluated neural network.

III. Multi-Retention Arrays with adaptive Training

A. Experimental Setup

Our general design and evaluation flow is shown in Fig. 6. In order to show the main concept of this work and evaluate the results, we use the following classification task. A visualization of our neural network is shown in Fig 4.

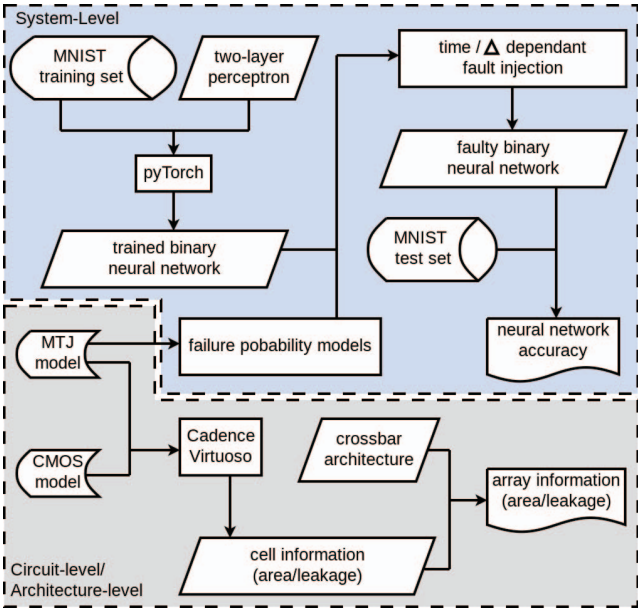


Fig. 6: Overview of our design and evaluation flow.

TABLE I: MTJ parameter setup

Δ	Parameter	Value
40	MTJ radius	29 nm
	Free/Oxide layer thickness	2.21/1.48 nm
	'AP'/'P' resistance	5.1 kΩ/2.3 kΩ
	Critical Write Current I_c	13.6 uA
60	MTJ radius	27 nm
	Free/Oxide layer thickness	2.16/1.48 nm
	'AP'/'P' resistance	5.9 kΩ/2.6 kΩ
	Critical Write Current I_c	19.7 uA

We trained a two-layer Perceptron using pytorch [19] with 1024 hidden neurons as described in [11] on the MNIST dataset [10] to recognize hand-written digits from zero to nine for the evaluation of our work. The data from the dataset is composed of 60000 28×28 pixel black and white images. The input vector size is therefore $28 \times 28 = 784$. As the dataset is built from digits, the possible output is a character from 0 to 9, so the output vector size is 10. Each neuron of the first (hidden) layer is connected to each input neuron. The same is done for each second layer (output) neuron and all neurons from the first layer. The first and the second layers are therefore *fully connected layers*. The full network topology is $[784 \times 1024 \times 10]$, resulting in $|W_1| = 784 \times 1024 = 802,816$ number of weights in the first layer and $|W_2| = 1024 \times 10 = 10,240$ number of weights in the second layer. The resulting binary network is mapped to two different MTJ crossbars, W_1 is mapped to the MTJ array A_1 and W_2 is mapped to the MTJ array A_2 (see Fig. 5).

For this mapping, each cell is used to calculate the XNOR operation result of its input and its state. By enabling the cell according to the input neuron and sensing the state to get the result, a sense mechanism in the output neuron can be used to evaluate the network operation either serial or in parallel. For proper parallel calculation the sensing can be done in two phases. First, all cells with 1 as an input are activated and the enabled HRS cells are counted in the neuron. In the second phase, all cells with -1 as an input are activated and the cells in LRS are counted and added to the counter from phase one. This way, the binary neural network can be directly mapped on the crossbar and used correctly.

We used the TSMC 40nm low power library and the MTJ model as described in [20] in Cadence Virtuoso. Our reference design is an MTJ crossbar with a thermal stability factor of 60. We reduced the majority of the MTJs in this crossbar with MTJs with a thermal stability factor of 40. For our area estimation we assumed a write current of twice the critical write current shown in Table I for the two different MTJ models, both with a Resistance-Area product of 6.12 $\Omega\mu m^2$ and a TMR of 123%. The supply voltage was 1.1V and the assumed temperature was 27°C.

B. Retention-induced Inference Degeneration

First, we want to evaluate the implications of retention failures over time on the inference accuracy of the trained network. We evaluated the AP $\rightarrow$ P switching in A_1 depending on the thermal stability factor Δ of the MTJs. To evaluate the impact of retention faults, we model the switching probability as described in Section II and simulate the switching stepwise. Each step is corresponding to 10% of the expected operational of 10 years. By varying this expected operational time, the needed thermal stability and consequently the results that are presented later are shifted accordingly. A lower thermal stability leads to a higher switching probability and consequently to a higher observed switching rate in the simulation. For a fixed thermal stability factor, more cells in HRS increase the probability of a switching to happen as each MTJ switching is independent from one another [8].

We initially train our two-layer perceptron on the MNIST database and calculate the inference accuracy based on the test set. Then the array is iteratively injected with random faults in the first layer depending on the switching probability for one year according to the thermal stability factor. This modified network is then again used to calculate the inference accuracy. We repeat the above until we cover the full expected operational time. This is done for multiple thermal stability factor values of MTJs.

As shown in Fig. 7 the initial training was able to achieve an inference accuracy of 92% on the test set with about half of the MTJs in A_1 set to the HRS. Please note that this can be further improved by selecting a different network topology or increasing the number of neurons used in the network. For conventional memory operation a thermal stability factor of 60 is statistically considered sufficient to reliably store data over a ten year period [21]. Therefore, we are interested in the impact on lower thermal stability factor and show the results of the simulation around the interesting turning point. We can see, that the accuracy drops rapidly over time for Δ below 40 and will finally be stuck at inferring a single output all the time, as all weights switched to the LRS. For $\Delta = 40$, the inference accuracy is only moderately decreasing and above 40 there is only minimal impact on the accuracy, even with a small number of HRS MTJs switched.

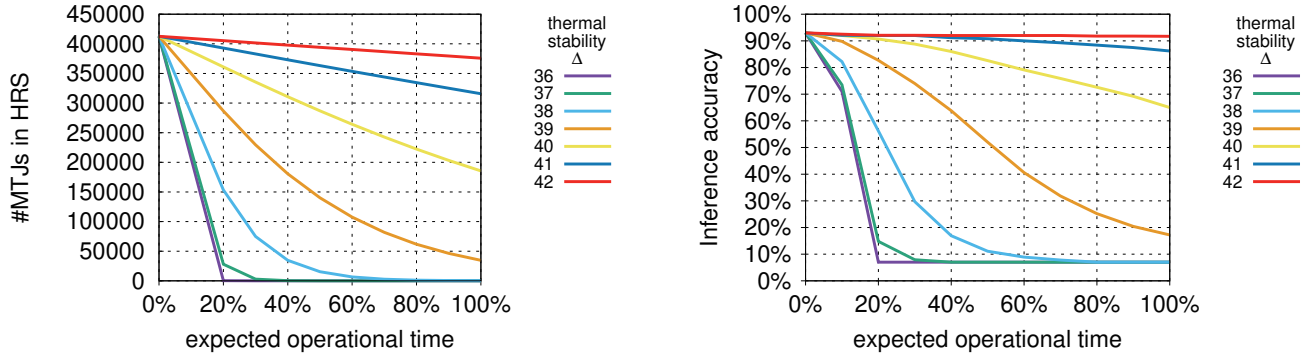
C. Minimizing Switching Probability due to Retention Failures

As discussed before, the retention failures are asymmetric and are more likely to occur only in one direction. This means that in MTJs, only the cells in HRS are subject to retention failures. Therefore, decreasing the number of MTJs in the array which are in HRS should reduce the total number of retention failures in the crossbar. By minimizing the number of HRS MTJs, our objective is to reduce the number of retention faults and to minimize the inference accuracy loss We use the modified cost function

$$J'(W, b, x, y) = J(W, b, x, y) + \alpha * \sum_{w \in W} w \qquad (3)$$

for the training, which uses an additional α-weighted regularization term in form of the sum of all weights in the layer. Typically, regularization is done with a norm like the L1-Norm (the sum of absolute weights) or the L2-Norm (the root of summed squared weights). Both of those norms are used to minimize the absolute value of weights, so the overall tendency of a weight is towards zero. In the binarization step of the training and the inference the weight is sampled to either -1 or +1.

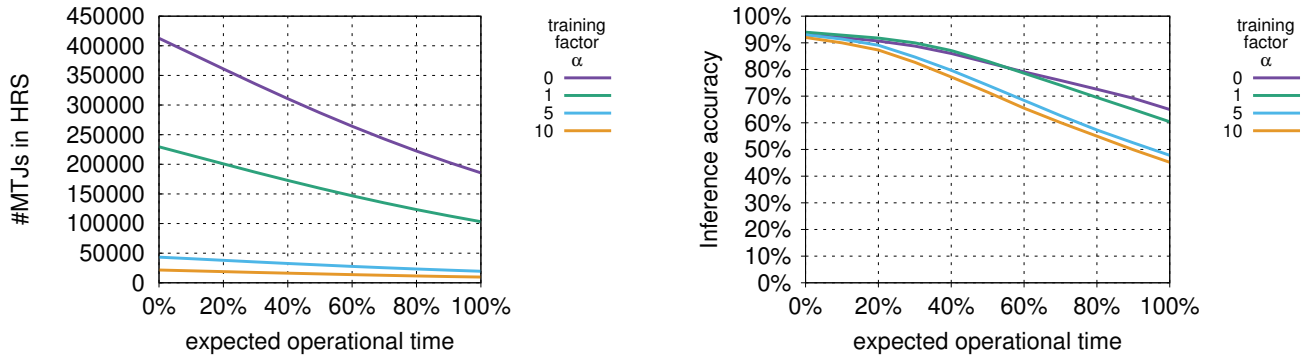
By minimizing the sum of the weights, they are pulled towards -1. For our following results we trained the neural networks as described in the last section with the adapted



(a) HRS cell degeneration.

(b) Inference accuracy degeneration.

Fig. 7: The impact of retention failures due to different thermal stability factors of MTJs on the inference accuracy of neural network trained with **conventional** [11] training algorithm



(a) HRS cell degeneration.

(b) Inference accuracy degeneration.

Fig. 8: Influence of α on minimizing the number of MTJs in HRS evaluated for $\Delta = 40$.

cost function and $\alpha = 10$. The influence of α on the training accuracy is shown in Fig. 8. A larger value increases contribution of the regularization term. Therefore, we chose $\alpha = 10$ as a larger value would not imply a significant improvement in the number of HRS MTJ (Fig. 8a), but still worsen the overall accuracy loss after the expected operational time (Fig. 8b).

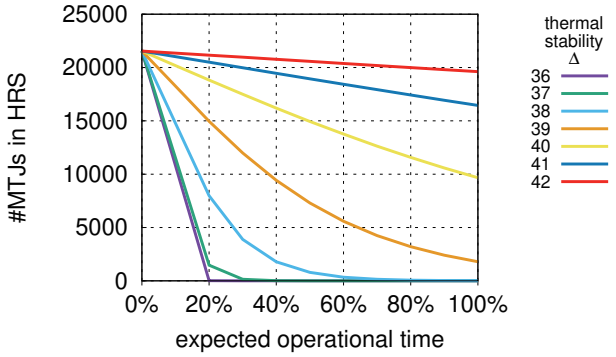
The result of this adapted training is shown in Fig. 9. Please note the difference of the scale in the y-axis in Fig. 7a and Fig. 9a. We can see that the training, as shown in time step zero, was able to reduce the number of initial HRS MTJs from about half of all the MTJs in A_1 (400k/800k) to about 2.5% (20k/800k) without sacrificing the initial inference accuracy, which is still 92%. Considering the degeneration over time, this approach is inferior to the conventional one presented earlier. The number of retention faults was reduced in absolute numbers, because there were less cells in HRS, but now the underlying information of the network is stored much denser in far fewer MTJs. Consequently, a fault in one of the remaining HRS MTJs is much more severe than previously and the overall inference accuracy of the neural network is more severely impacted by retention failures. As shown in Fig. 9b and compared with Fig. 7b the overall inference accuracy of the network deteriorates sharper over time.

For comparison, Fig. 10 shows the distribution of HRS MTJ over A_1 for the conventional training and the adapted training with $\alpha = 10$. The adapted training results in a significantly sparser HRS MTJ matrix. When the HRS MTJ are sorted by the total number per column and accordingly rearranged, most of them can be put close together, as shown in Fig. 11. With the conventional training, half of the columns had about 550 HRS MTJs and the other half of them around 250 HRS MTJs (Fig. 12a). In contrast, with the adapted training most of the columns are not used and only a few of them are above 100 HRS MTJs (Fig. 12b). This motivates our next step in which we use a mixed-retention crossbar array.

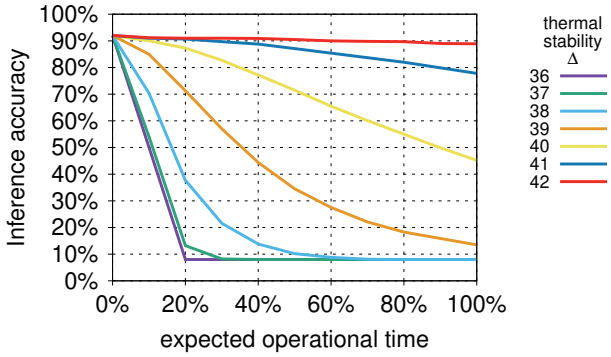
D. Mixed Retention Crossbar Array

The conventional approach is to use a high retention array for the entire crossbar array. However, the support of the high Δ MTJs has a large area and power requirement. These cells need a high current to be written and consequently larger access transistors, which suffer from a high leakage and use more area. Therefore, we propose the usage of a mixed retention crossbar as shown in Fig. 13. By reducing the thermal stability, and consequently the retention, of a sub-array within the crossbar, it is possible to reduce the size of the access transistors and the write circuits for the corresponding columns.

After minimizing the HRS weights the significant information of the neural network is mainly stored to only a fraction of the MTJs. Since the trained network becomes more sensitive to the faults in those critical cells, it motivates the design of a

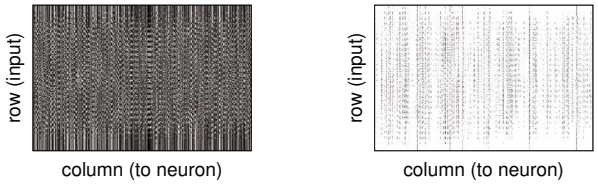


(a) HRS cell degeneration.



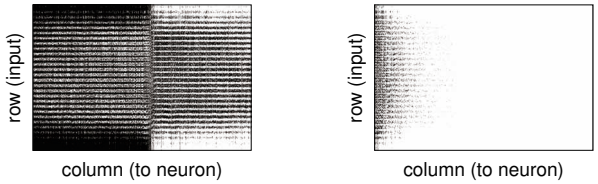
(b) Inference accuracy degeneration.

Fig. 9: The impact of retention failures due to different thermal stability factors of MTJs on the inference accuracy of neural network trained with our **proposed adapted** training algorithm



(a) Conventional training. (b) Adapted training.

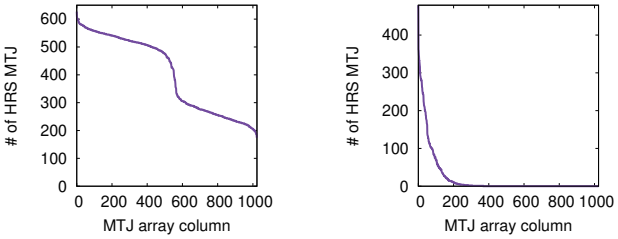
Fig. 10: Distribution of MTJs in HRS in the MTJ crossbar after training. Black indicates a HRS MTJ, white a LRS MTJ.



(a) Conventional training. (b) Adapted training.

Fig. 11: Distribution of MTJs in HRS in the MTJ crossbar after sorting by the number of HRS per column. Black indicates a HRS MTJ, white a LRS MTJ.

hybrid crossbar array to mitigate retention failures. The main idea is that we use high retention cells, which are mostly immune to retention failures, only for a very small subset of the array which stores HRS weights. The larger subset (majority) of the array is designed with low retention cells



(a) Conventional training ($\alpha = 0$). (b) Adapted training.

Fig. 12: Number of HRS MTJs per column after sorting.

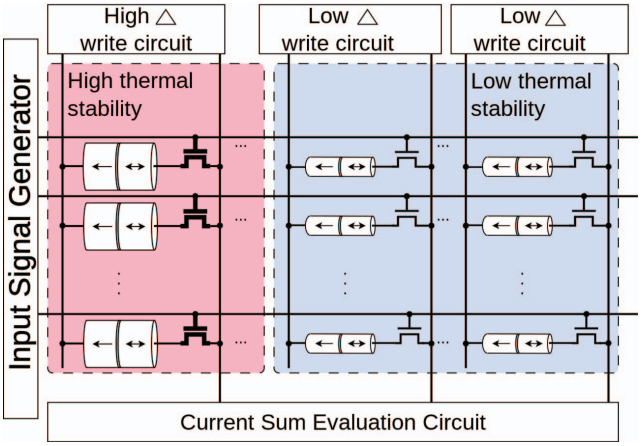
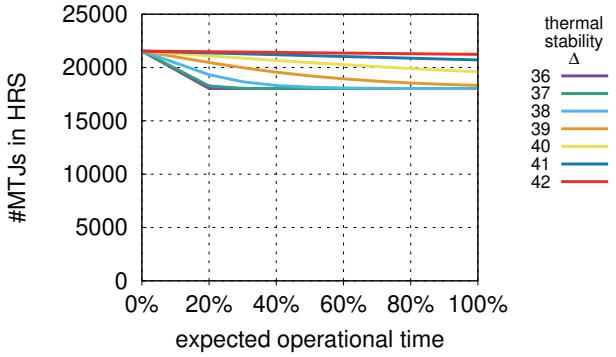


Fig. 13: A multi-retention MTJ crossbar with two different thermal stabilities.

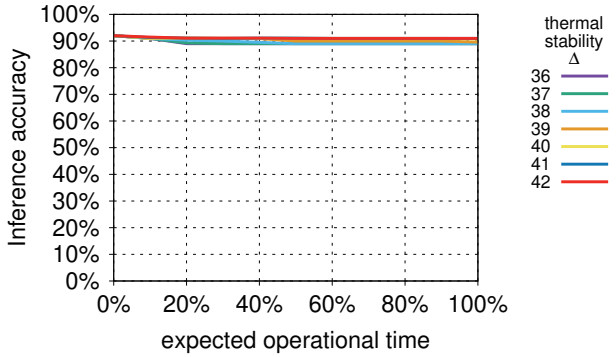
to store LRS weights. By paying the price (area and write energy/latency) for a small subset, we are able to achieve high retention reliability of the entire neural network and avoid inference accuracy loss over time.

By using a mixed retention array, we can map the important neurons, the ones with a significant amount of +1 weights, to the columns with a high stability. The low thermal stability MTJs on the other hand allow for an adjusted access transistor, which does not need to provide a current level as high as the current for high Δ MTJs. Hence, the access transistors in the low Δ section of the array can be smaller, which directly impacts the area of the whole array, as they are the main area consideration for the cells. Additionally, the smaller access transistors reduce the leakage power of the array.

Until now, we discussed a reduced retention on designs with vertical input and horizontal output, meaning that the inputs to a crossbar was row-wise, whereas its output was column-wise. For more complex neural network routings, it is possible to design an accelerator with multiple crossbars in a way, that they can be used like that and additionally in a transposed manner with switched row/column behavior to allow for efficient arbitrary routings. This is for example the case, when directly routing the output of the first layer to the second layer. If the overall architecture is capable of arbitrary



(a) HRS cell degeneration.



(b) Inference accuracy degeneration.

Fig. 14: Adapted training on mixed arrays with varying Δ for the majority of the array and 10% MTJs with $\Delta = 60$

routing layers, it would also improve the general usage of the mixed retention array by using a mixture of high retention rows and high retention columns.

For our evaluation, we additionally prepared the area savings of our mixed retention array for the previously described two-layer perceptron. $\Delta = 40$ was picked as it was a reasonable trade-off between fast degeneration and long term stability (compare Fig. 7b and 9b). The switching behavior of this crossbar is shown in Fig. 14a. We can see that the general trend for the retention faults is comparable to the previously described cases, meaning that the faults still happen depending on the remaining cells in HRS. However, as most of them are now put in the high retention sub-array, the fault rate drops significantly, after all the low retention MTJs were switched to LRS. The remaining fault free cells are the cells that are protected by the higher retention MTJs. This directly reflects on the inference accuracy shown in Fig. 14b, which only drops insignificantly over the simulated period. The decrease of the write current from 39 uA to 27 uA saves 37% in area for a cell using the $\Delta = 40$ MTJs instead of the $\Delta = 60$ MTJs. Additionally, the leakage power for activated cells is decreased by 43 % per cell. Consequently, the mix of 90% MTJs with $\Delta = 40$ and 10% with $\Delta = 60$ is able to save 33% in array area due to the reduction of the access transistors and reduces the possible leakage power of the evaluated array by 39%.

IV. Conclusion

In this paper, we showed the influence of asymmetric retention faults on the inference accuracy of neural networks mapped into non-volatile resistive memories. We proposed a mixed retention crossbar array plus an adapted neural network training to minimize the number of cells in a high resistance state, as they are more prone to retention faults. While our evaluations were performed on Magnetic Tunnel Junctions, the main concept is applicable to other resistive memory technologies. This saves the inference accuracy over time by 24%.

References

[1] M.-T. Chang, P. Rosenfeld, S.-L. Lu, and B. Jacob, "Technology comparison for large last-level caches (L-3 Cs): Low-leakage SRAM, low write-energy STT-RAM, and refresh-optimized eDRAM," in *HPCA*, 2013, pp. 143–154.

[2] S. A. Wolf, J. Lu, M. R. Stan, E. Chen, and D. M. Treger, "The promise of nanomagnetics and spintronics for future logic and universal memory," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2155–2168, 2010.

[3] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y. . Chen, R. M. Shelby, M. Salinga, D. Krebs, S. . Chen, H. . Lung, and C. H. Lam, "Phase-change random access memory: A scalable technology," *IBM Journal of Research and Development*, vol. 52, no. 4.5, pp. 465–479, July 2008.

[4] G. W. Burr, M. J. Brightsky, A. Sebastian, H. Cheng, J. Wu, S. Kim, N. E. Sosa, N. Papandreou, H. Lung, H. Pozidis, E. Eleftheriou, and C. H. Lam, "Recent progress in phase-change memory technology," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 2, pp. 146–162, June 2016.

[5] I. G. Baek, M. S. Lee, S. Seo, M. J. Lee, D. H. Seo, D. . Suh, J. C. Park, S. O. Park, H. S. Kim, I. K. Yoo, U. . Chung, and J. T. Moon, "Highly scalable nonvolatile resistive memory using simple binary oxide driven by asymmetric unipolar voltage pulses," in *IEDM Technical Digest. IEEE International Electron Devices Meeting*, Dec 2004, pp. 587–590.

[6] R. Waser, R. Dittmann, G. Staikov, and K. Szot, "Redox-based resistive switching memories nanoionic mechanisms, prospects, and challenges," *Advanced Materials*, vol. 21, no. 2526, pp. 2632–2663, 2009.

[7] Y. Chen, X. Wang, H. Li, H. Xi, Y. Yan, and W. Zhu, "Design margin exploration of spin-transfer torque ram (stt-ram) in scaled technologies," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 18, no. 12, pp. 1724–1734, 2009.

[8] K. Hofmann, K. Knobloch, C. Peters, and R. Allinger, "Comprehensive statistical investigation of stt-mram thermal stability," in *Symposium on VLSI Technology: Digest of Technical Papers*, June 2014, pp. 1–2.

[9] K. Tsunoda, M. Aoki, H. Noshiro, Y. Iba, S. Fukuda, C. Yoshida, Y. Yamazaki, A. Takahashi, A. Hatada, M. Nakabayashi, Y. Tsuzaki, and T. Sugii, "Area dependence of thermal stability factor in perpendicular stt-mram analyzed by bi-directional data flipping model," in *IEEE International Electron Devices Meeting*, Dec 2014, pp. 19.3.1–19.3.4.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, Nov 1998.

[11] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[12] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *arXiv e-prints*, p. arXiv:1603.05279, Mar 2016.

[13] T. Chiba, M. Natsui, and T. Hanyu, "Design of a current-mode linear-sum-based bitcounting circuit with an mtj-based compensator for binarized neural networks," in *IEEE International Symposium on Multiple-Valued Logic (ISMVL)*. IEEE, 2019, pp. 91–96.

[14] A. F. Vincent, J. Larroque, N. Locatelli, N. B. Romdhane, O. Bichler, C. Gamrat, W. S. Zhao, J. O. Klein, S. Galdin-Retailleau, and D. Querlioz, "Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 2, pp. 166–174, Apr. 2015.

[15] E. I. Vatajelu and L. Anghel, "Fully-connected single-layer stt-mtj-based spiking neural network under process variability," in *Proc. IEEE/ACM Int. Symp. Nanoscale Architectures (NANOARCH)*, Jul. 2017, pp. 21–26.

[16] Y. Xiang, P. Huang, Y. Zhao, M. Zhao, B. Gao, H. Wu, H. Qian, X. Liu, and J. Kang, "Impacts of state instability and retention failure of filamentary analog rram on the performance of deep neural network," *IEEE Transactions on Electron Devices*, pp. 1–6, 2019.

[17] Y. Deguchi, K. Maeda, S. Suzuki, T. Nakamura, and K. Takeuchi, "Error-reduction controller techniques of taox-based reram for deep neural networks to extend data-retention lifetime by over 1700x," in *IEEE International Memory Workshop (IMW)*, May 2018, pp. 1–4.

[18] C. Münch, R. Bishnoi, and M. B. Tahoori, "Reliable in-memory neuromorphic computing using spintronics," in *Proceedings of the Asia and South Pacific Design Automation Conference*. ACM, 2019.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[20] A. Mejdoubi, G. Prenat, and B. Dieny, "A compact model of precessional spin-transfer switching for MTJ with a perpendicular polarizer," in *MIEL*, 2012.

[21] E. Chen, D. Apalkov, Z. Diao, A. Driskill-Smith, D. Druist, D. Lottis, V. Nikitin, X. Tang, S. Watts, S. Wang, S. A. Wolf, A. W. Ghosh, J. W. Lu, S. J. Poon, M. Stan, W. H. Butler, S. Gupta, C. K. A. Mewes, T. Mewes, and P. B. Visscher, "Advances and future prospects of spin-transfer torque random access memory," *IEEE Transactions on Magnetics*, vol. 46, no. 6, pp. 1873–1878, June 2010.