SCANet: Securing the Weights with Superparamagnetic-MTJ Crossbar Array Networks

Dinesh Rajasekharan, *Member, IEEE*, Nikhil Rangarajan, *Member, IEEE*, Satwik Patnaik, *Member, IEEE*, Ozgur Sinanoglu, *Senior Member, IEEE*, and Yogesh Singh Chauhan, *Fellow, IEEE*

***Abstract*—Deep neural networks (DNNs) form a critical infrastructure supporting various systems, spanning from the iPhone neural engine to imaging satellites and drones. The design of these neural cores is often proprietary or a military secret. Nevertheless, they remain vulnerable to model replication attacks that seek to reverse engineer the network's synaptic weights. In this article, we propose SCANet, a novel defense mechanism against such model stealing attacks by utilizing the innate stochasticity in superparamagnets. When used as the synapse in DNNs, superparamagnetic magnetic tunnel junctions (s-MTJs) are shown to be significantly more secure than prior memristor-based solutions. The thermally-induced telegraphic switching in the s-MTJs is robust and uncontrollable, thus thwarting the attackers from obtaining sensitive data from the network. Using a mixture of both superparamagnetic and conventional MTJs in the neural network (NN), the designer can optimize the time period between the weight updation and the power consumed by the system. Furthermore, we propose a modified NN architecture that can prevent replication attacks while minimizing power consumption. We investigate the effect of the number of layers in the deep network and the number of neurons in each layer on the sharpness of accuracy degradation when the network is under attack. We also explore the efficacy of *SCANet* in real-time scenarios, using a case study on object detection.**

***Index Terms*—Deep neural network (DNN), hardware security, magnetic tunnel junction (MTJ), model replication attack, superparamagnets.**

1. Introduction

NEURAL networks have become crucial for big data processing and computer vision applications in recent years owing to their computational capabilities, versatility, and fault tolerance [1]. Deep neural networks (DNN), in particular, are now ubiquitous in almost every field, ranging from healthcare, agriculture, commercial, and military electronics [2]–[7]. With the explosion of the Internet-of-Things and the move away from data centers, towards edge computing paradigms, hardware neural networks (NN) are indispensable blocks in the system design flow of the future. Such advanced edge devices require the integration of neural computation at their core to achieve high speeds of operation and reduce the computational load on cloud systems [8]. However, with the pervasive spread of hardware NNs, there is a growing demand to secure them against adversarial entities, which remains largely unexplored.

Intellectual property (IP) protection has been a cause of concern for big multinational corporations as well as defense agencies, with the emergence of potent reverse engineering and piracy techniques [9], [10]. Theft of proprietary, secret military hardware designs and data poses serious repercussions for national security. For instance, a surveillance and reconnaissance drone launched by the military, when shot down and eventually captured by adversaries, could leak sensitive information about the motive of the flight or it's underlying system design.

Typically, the two most prevalent approaches for securing a CMOS integrated circuit (IC) design against IP piracy are logic locking [11] and layout camouflaging [12], [13]. Logic locking involves appending the original circuit with additional key gates, which pass the correct logic only on the application of the secret key. Layout camouflaging, on the other hand, attempts to make the logic gates look indistinguishable at the layout level, so that that the attacker gets confused and has to use more computational resources to decipher the true nature of the gates.

Mainstream NN implementations, as of now, are based on general purpose CPU/GPU software [14], [15] or on dedicated FPGA/ASIC hardware [16], [17]. However, crossbar array-based implementation of NNs are heavily researched, because of their potential area and energy efficiency [18]–[20]. Our work focuses particularly on securing such crossbar array-based hardware NNs constructed with emerging memory technologies.

In such crossbar array-based hardware NNs, the intended functionality is achieved by proper programming of the synaptic weights. Therefore, an attacker can pirate the NN design by reverse engineering these weights that are represented by the programmable conductance of non-volatile memories

Manuscript received November 21, 2020; revised May 2, 2021 and August 28, 2021; accepted November 16, 2021. This work was funded by the Swarnajayanti Fellowship (Grant No.– DST/SJF/ETA-02/2017-18) of the Department of Science and Technology, India. (*Corresponding author: Dinesh Rajasekharan.*)

Dinesh Rajasekharan and Yogesh Singh Chauhan are with the Department of Electrical Engineering, Indian Institute of Technology Kanpur, Kanpur 208016, India (email: dineshr@iitk.ac.in, chauhan@iitk.ac.in).

Nikhil Rangarajan and Ozgur Sinanoglu are with the Division of Engineering, New York University Abu Dhabi, Saadiyat Island 129188, UAE (e-mail: nikhil.rangarajan@nyu.edu, ozgursin@nyu.edu).

Satwik Patnaik was with the Department of Electrical and Computer Engineering, Tandon School of Engineering, New York University, Brooklyn, NY 11201, USA. He is currently with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: satwik.patnaik@tamu.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2021.3130884.

Digital Object Identifier 10.1109/TNNLS.2021.3130884

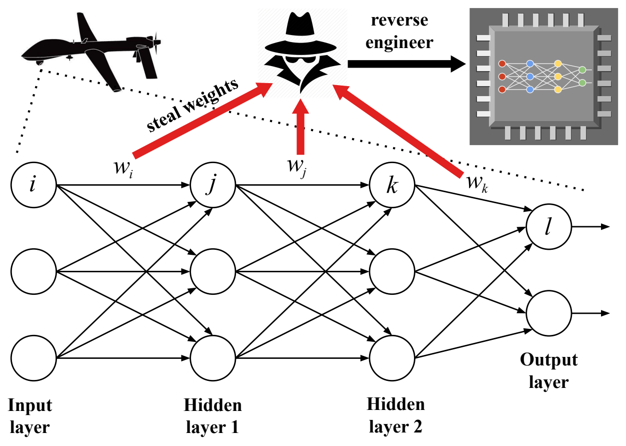


Fig. 1: An attacker can reverse engineer the deep neural network (DNN) on a surveillance drone by identifying the network weights.

by the programmable conductance of non-volatile memories (NVM) such as resistive random access memories (RRAM), phase change memories (PCM) and magnetic tunnel junctions (MTJs) [21]. However, dense packing of these memories within the IC makes reading the conductance of the memory cells intractable, without damaging the memory array [22]. Figure 1 illustrates reverse engineering of an image processing DNN mounted on a drone.

1.1. Prior Work

State-of-the-art reverse engineering attacks on hardware NNs include the *model replication attack* proposed in [23]. In this attack, an attacker gains illicit access to the network and is considered unconstrained concerning the application of inputs. The attacker then proceeds to collect the corresponding output responses, and these input-output (I/O) pairs are then leveraged to infer the weights of the underlying network. Using a sufficiently large dataset of such I/O pairs, the authors have shown that they can train a new network to a degree of likeness sufficiently close to the original network. The authors in [24] introduced a solution to this problem using the obsolescence effect in memristors. They thwart the attacker from collecting an adequate dataset by exploiting the drift in memristance due to the continual application of input voltages and the intrinsic retention property of the device. Hence, only a certain number of I/O pairs can be recorded before the weights get corrupted beyond recovery. However, the pitfall in their defense technique is that the obsolescence effect in memristors can be controlled and diminished by lowering the input voltage magnitude while taking the measurements. Therefore, an attacker might still be able to operate the system for a sufficient time to obtain adequate I/O pairs to replicate the NN.

Other prior works in the field of NN security include [25], where the authors proposed a reverse engineering algorithm to extract the rules from trained NNs, for datasets with mixed-mode attributes. Their algorithm prunes the insignificant input neurons from the trained NNs and reconstructs the classification rules with the remaining significant input neurons. Oh *et al.* [26] demonstrated methods to craft queries for black-box NNs, which seek to infer the internal information of the networks by output prediction. Both these attacks were demonstrated on software NNs. Contrastingly, other attacks on hardware NNs have relied on exploiting the side-channels of the system. In [27], both the structure and weights of encrypted convolutional neural networks (CNN) models are reverse-engineered through memory and timing side-channels by observing the off-chip memory accesses that specific inputs cause. This attack utilizes the distinguishing memory access patterns and read-after-write dependencies to decipher details, such as the number of layers, connections between the layers, and the size of I/O feature maps. Batina *et al.* [28] introduced an electromagnetic side-channel-based attack for multilayer perceptrons and CNNs. Their attack involves measuring the EM radiation while the secret pre-trained weights are loaded from memory for processing, inducing Hamming weight leakage.

1.2. Our Contributions

This paper proposes SCANet, a design-for-trust technique for DNNs against reverse engineering-based adversarial attacks, which leverages the thermal drift in superparamagnetic MTJs (s-MTJs). We demonstrate that designing the synaptic weights of the DNN with s-MTJs is conclusively better than the prior memristor-based approaches.

So far, there has only been one work that describes the prevention of model replication attacks on crossbar array-based NNs, by using the conductance drift in memristors [24]. We present the first attack models to break the security primitives proposed in [24]. We show that, by lowering the input voltage magnitude, it is possible to decrease the rate of conductance drift in memristors, thus, allowing an attacker to easily replicate the NN. However, the thermal conductance drift in s-MTJs is temporal rather than based on the number and magnitude of inputs applied, hence, rendering such attacks ineffective.

The primary contributions of this work are as follows.

1) We design SCANet as a robust defense against model replication attacks by employing the thermally-induced superparamagnetic switching phenomenon, which cannot be controlled externally.
2) We explore three different DNN architectures and study their efficacy concerning metrics like maximum accuracy degradation, non-linearity, number of stochastic weights required, and power consumption for weight update.
3) We benchmark the three architectures in terms of their refresh energy and power, and consider their trade-offs concerning power consumption and security afforded.
4) We highlight practical applications of SCANet in protecting an object detection IP against reverse engineering.
5) We conduct a thorough analysis of various attack vectors against *SCANet*, including temperature-, bias- and reliability-based attacks, transfer learning, logic removal attacks, etc., and suggest suitable counters.

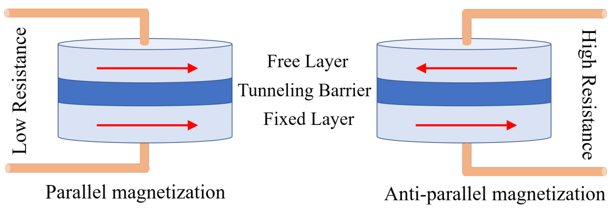


Fig. 2: MTJ device structure and resistance states. The resistance across the MTJ is low (high) when the magnetic moments of the fixed and free layers are parallel (anti-parallel).

The organization of the paper is briefly described next. We explain the working of s-MTJs in Section II and elucidate the different DNN architectures explored for preventing model replication attacks in Section III. Section IV illustrates the impact of s-MTJ synaptic weights on the recognition accuracy of DNNs, trained on MNIST and CIFAR-10 datasets. The advantages of the proposed technique are highlighted in Section V, while Section VI discusses the potential attack methodologies on *SCANet*, followed by concluding remarks.

2. Superparamagnetic MTJs

An MTJ consists of two ferromagnetic layers, composed of materials like Fe, Co, CoFeB, etc., separated by a thin tunneling barrier, which is typically an oxide like MgO or Al_2O_3 [29]. One of the two magnetic layers is a highly coercive hard magnet, and hence, its magnetization direction is fixed. This is known as the *fixed layer* of the MTJ. The magnetization of the other magnet can be switched between two basins of equilibria, using external magnetic fields [30] or spin-polarized currents [31]. This layer is called the *free layer* of the MTJ. The MTJ can be in one of two resistance states, depending on the fixed and free layers' relative orientation. If both the layers have their magnetization pointing in the same direction, the MTJ exhibits low resistance. On the other hand, if their magnetizations are aligned opposite to each other, then the MTJ is in a high resistance state. These states of an MTJ are illustrated in Fig. 2.

2.1. Néel Relaxation

In a conventional MTJ memory, the magnetization of the free magnet is thermally stable and does not switch without any external impetus (field or current). This is because the energy barrier between the two stable basins, $E_\mathrm{b} = 1/2\mu_0 M_\mathrm{S} H_\mathrm{K} V$, is too high to surmount with just thermal fluctuations (see Fig. 3). Here, M_S is the saturation magnetization, H_K is the uniaxial anisotropy field and V is the volume of the magnet. The Néel retention time, or the time for which a magnet retains its state without thermally flipping to the other state is given as [32]

$$\tau_\mathrm{N} = \tau_0 \exp \frac{E_\mathrm{b}}{k_\mathrm{B} T}, \tag{1}$$

where k_B is Boltzmann's constant, T is the temperature and τ_0 is the attempt period of the material. With large E_b of 40× the thermal energy $k_\mathrm{B}T$ [33], the retention time of typical non-volatile memories is 10 years or more.

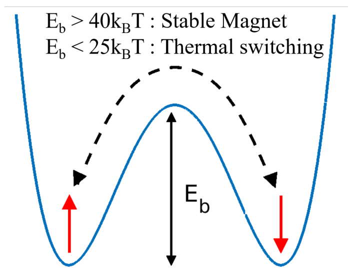

$E_b > 40k_BT$: Stable Magnet
$E_b < 25k_BT$: Thermal switching
E_b

Fig. 3: Energy barrier (E_b) of the free magnet in a two-well model. When E_b is more than 40 $k_\mathrm{B}T$, then the retention time ∼10 years. Here, $k_\mathrm{B}T$ represents the thermal energy. However, if the E_b is less than 25 $k_\mathrm{B}T$, the magnets tend to show stochastic switching due to thermal noise.

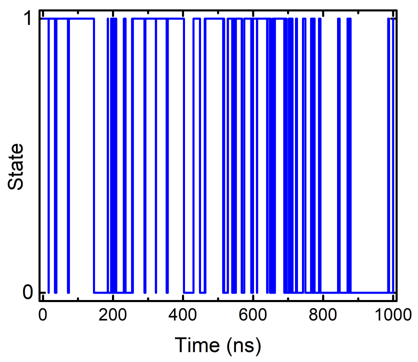
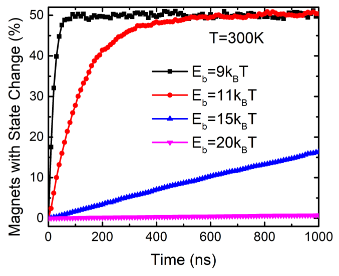


Fig. 4: Stochastic switching of a superparamagnet with an E_b of 9 $k_\mathrm{B}T$, over time.

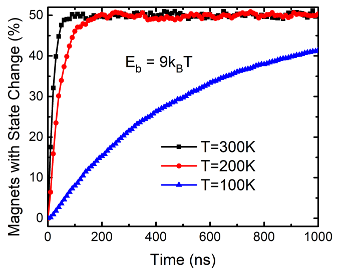
However, the E_b of the free magnet can be reduced, either
However, the E_b of the free magnet can be reduced, either through materials engineering to obtain a small M_S and H_K or by shrinking the volume of the magnet. For some materials like CoFeB and Permalloy, when the dimensions of the magnet are in the order of hundreds of nm, the E_b will only be a few $k_\mathrm{B}T$. This implies that a small enough free magnet can flip stochastically due to the thermal energy, as shown in Fig. 3. Generally, such a stochastic switching phenomenon is observed for E_b less than 25 $k_\mathrm{B}T$ [34]. Note here that, when the free magnet is sufficiently small, it will exist in a single domain state [35], and therefore can carry a net magnetic moment without any applied field. Such nanomagnets, with E_b in the order of $k_\mathrm{B}T$ are known as superparamagnets.

The stochastic evolution of the magnetization state of a superparamagnet (E_b = 9 $k_\mathrm{B}T$) over time is shown in Fig. 4. In the figure, the two states "0" and "1" denote the stable states of the magnet. It can be seen that the magnet, after a short retention time, starts to show stochastic switching between the two states. It is known from prior research that this switching does not follow any predictable pattern, and therefore the MTJs designed with these superparamagnets are also used to construct true random number generators [36], [37].

The impact of E_b and the operating temperature on the retention time of the superparamagnet is shown in Fig. 5. Here, over a time, the percentage of magnets that flipped from their initial state to a different final state is plotted. The time that it takes for approximately 50% of the magnets to flip can be seen as a measure of the retention time of



(a)



(b)

Fig. 5: Plot of percentage of magnets that have flipped from their initial value as a function of time for (a) different E_b values at T = 300 K, and (b) for different temperatures with E_b = 9 $k_\mathrm{B}T$. The sample space for the simulation consisted of 10,000 nanomagnets.

TABLE I:
Simulation parameters used for the superparamagnet

Simulation parameters	Values
Saturation magnetization, M_S	$10\,000$ A/m
Uniaxial energy density (K_U)	7000-12000 Jm^{-3}
Gilbert damping constant (α)	0.01
Volume of magnet	$60\times45\times2\,\mathrm{nm}^3$
Temperature	300 K
Energy barrier (E_b)	$\sim$ 9-15 $k_\mathrm{B}T$

the magnet. From Fig. 5(a), we can infer that the retention time of the nanomagnet is higher when the E_b is higher. Fig. 5(b) highlights the fact that temperature is also a critical factor, with larger retention times achieved at lower temperatures. All the simulations of the superparamagnets shown in this paper are based on the stochastic Landau-Lifshitz-Gilbert-Slonczewski (s-LLGS) equation [38] for monodomain nanomagnets, with simulation parameters listed in Table I. This is a well known mathematical model describing the temporal evolution of nanomagnetic moments and has been found to accurately model s-MTJ devices [39]. The physics-based s-LLGS implementation used in this work [38] has been rigorously tested and benchmarked against results from the NIST-standard micromagnetic tool OOMMF [40], as well as SPICE models. More information on s-MTJs is provided in the section I of the supplementary document.

2.2. s-MTJ synapse structure

The structure of the s-MTJ device used in our analysis, with a superparamagnetic free layer, is depicted in Fig. 6. Here, the state of the superparamagnet also determines the state of the s-MTJ. That is, when the magnetization of the superparamagnet switches stochastically, the resistance of s-MTJ device will also switch stochastically between the high and low states. Note here that weights implemented using s-MTJs have to be refreshed periodically, on a time scale smaller than the retention time, for proper functionality.

A table comparing the properties of different types of memory elements discussed in this work, namely s-MTJs, MTJs and memristors, is shown in Table II.

3. Neural Network Architectures

Crossbar arrays aiding the vector-matrix multiplication (VMM) in hardware NNs can be implemented using several

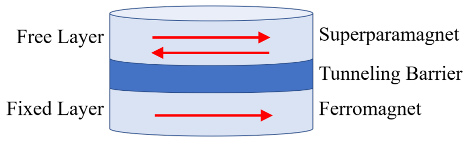


Fig. 6: s-MTJ device structure. Here, the free layer is composed of a superparamagnet. Therefore, the state of the s-MTJ switches stochastically with the superparamagnet.

TABLE II:
Comparison between different memory elements

	Memristors	MTJs	s-MTJs
No of states	High	2	2
Energy consumption per operation	$>$ 10 pJ [41]	$>$ 10 pJ [41]	$\sim$ 10 fJ [42]
Retention time	10 years [41]	10 years [41]	$\mathcal{O}$(ns)
MNIST inference accuracy	$>$ 90%	$>$ 90%	$>$ 90%

architectures. This section describes three such architectures for the *SCANet* framework, which afford varying degrees of protection against replication attacks and have different trade-offs for power dissipation during weight update. These architectures, used in our further analyses, are shown in Fig. 7. A more exhaustive explanation of the implementation details of these architectures is provided as supplementary material, for interested readers.

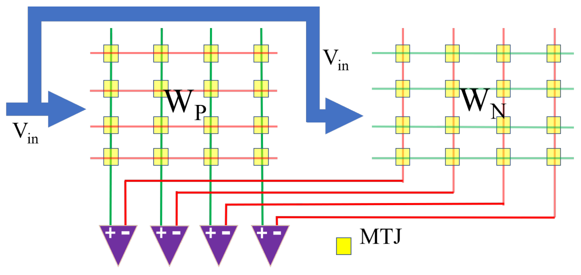
3.1. Architecture 1: Multi-level neural network

Figure 7(a) represents one layer of a conventional NN [43], which can be constructed with emerging memory devices like RRAM, PCM or MTJ. The current through the crossbar array will implement the VMM between the synaptic weights and the input. The inherent problem here is that the weights of a NN can be both positive and negative, whereas the of a NN can be both positive and negative, whereas the conductance of these emerging memories can only be positive. Therefore, to avoid this complication, two crossbar arrays are used. The negative weights are implemented in the crossbar array connected to the negative terminal of the sense amplifier (SA). In contrast, the crossbar array connected to the positive terminal is used to represent the positive weights. Here, the SA is part of the neuron circuit.

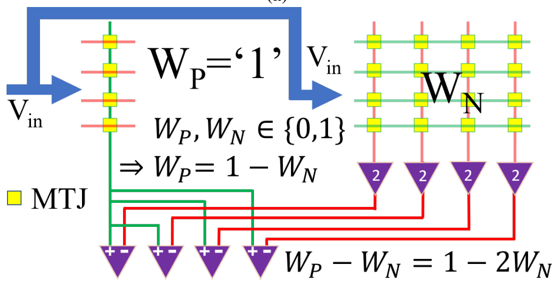
If s-MTJs are used to implement these synaptic weights, after a finite time, the NN will stop providing a correct response, making replication attacks unfeasible. The synaptic weights of NNs with this architecture can be multi-valued. Therefore, this architecture will be referred to as **multi-level NN** architecture, hereafter.

3.2. Architecture 2: Binary neural network

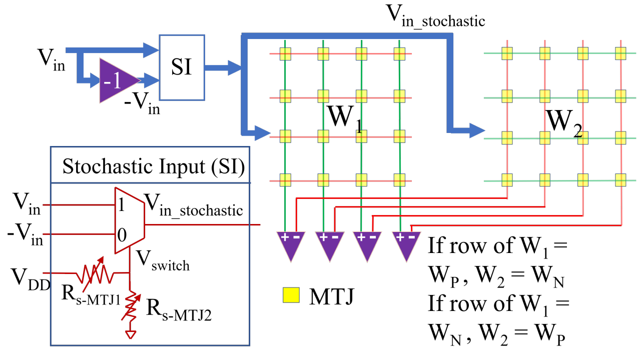
In contrast to RRAM and PCM, which exhibit continuous resistance variation, MTJs display only binary resistance states. For such binary systems, the architecture shown in Fig. 7(b) is better suited [44]. Here, the network is implemented such that the complete information is embedded within the crossbar array W_N connected to the negative terminal of



(a)



(b)



(c)

Fig. 7: Neural network architectures studied for preventing replication attack: (a) multi-level neural network, (b) binary neural network and (c) stochastic input neural network.

the SA. All the MTJs connected to the positive terminal are preset to the low resistance state. NNs with this architecture requires fewer MTJ crossbar junctions and are therefore more area-efficient than multi-level NNs.

Again, if s-MTJs are used in the array connected to the negative terminal, replication attacks can be hindered. However, using s-MTJs in the positive array may not yield many advantages as the attacker might already know that these MTJs are preset to the low resistance state. As this NN architecture can only be used with synaptic weights constrained to +1 or -1, this architecture will be referred to as **binary NN** architecture for the rest of this paper.

3.3. Architecture 3: Stochastic input neural network

The architecture in Fig. 7(c) is explicitly designed to reduce the power consumption during weight refresh while maintaining immunity against replication attacks. Here, the inputs are first randomized through stochastic input (SI) blocks, and the outputs of these blocks are fed to the NN. W_1 and W_2 denote the crossbar arrays connected to the SA's positive and negative terminals, respectively. The SI block consists of a 2:1 multiplexer. The output of the multiplexer can be V_{in} or $-V_{\text{in}}$ depending on the resistances of the s-MTJs, $R_{\text{s-MTJ1}}$ and $R_{\text{s-MTJ2}}$. With the temporal drift in these MTJ resistance states, the NN's input will gradually get corrupted.

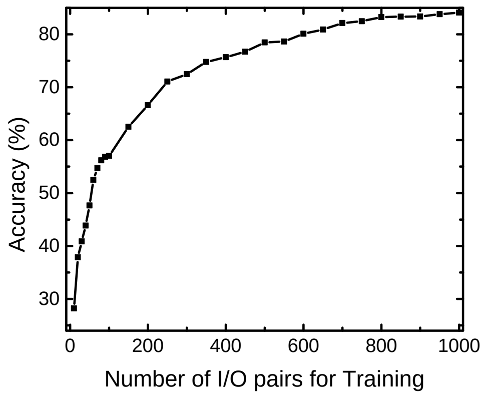


Fig. 8: Recognition accuracy obtained on the MNIST dataset vs. the number of I/O pairs used for training.

During the programming phase of the s-MTJs, the SI blocks can be configured to provide either $-V_{\text{in}}$ or V_{in} as input to the NN. This configuration will be done randomly and hence will be unknown to the attacker. Consider, for instance, that the s-MTJs in the SI block connected to W_1 are programmed to generate -V_{in}. The MTJs in the row of W_1 connected to this SI block will represent negative weights, whereas the MTJs in the row of W_2 connected to this SI block will represent positive weights. The vice-versa applies if the s-MTJs in the SI block is programmed to output V_{in}. Hereafter, Architecture 3 is referred to as **stochastic input NN** architecture.

4. Results

To reiterate, the primary motivation of this work is to protect DNNs from model replication attacks while minimizing the power overheads incurred. Model replication attacks work on the premise that if the attacker has sufficient I/O responses of the DNN, they can use learning algorithms to converge to the correct weights [23]. This has been illustrated in Fig. 8, which shows the relationship between the recognition accuracy obtained on the MNIST dataset and the number of I/O pairs used for learning. It can be seen that, with less than 300 I/O pairs, the attacker can replicate a DNN designed on MNIST dataset with more than 70% recognition accuracy. *Therefore, it is crucial to impede the attacker from repeatedly querying the DNN hardware after gaining access to the system.*

4.1. Multi-level neural network

Implementing all the weights in Fig. 7(a) with s-MTJs will result in stochastic variation, thus degrading the recognition accuracy drastically after a finite amount of time. This is highlighted in the results on the MNIST and CIFAR-10 datasets shown in Fig. 9. From Fig. 9(a), we see that the recognition accuracy (initially high) decreases to $\sim$10% with time, for E_b= 9-12 $k_\text{B}T$. Figure 9(b) shows the number of s-MTJs in the crossbar array that has switched from their initial

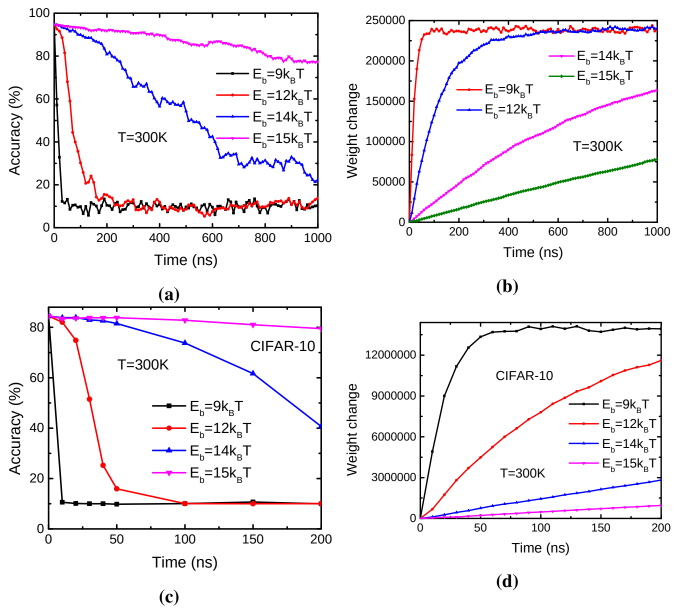


Fig. 9: (a,c) Recognition accuracy and (b,d) s-MTJ state variation over time observed on multi-level DNN, designed to operate on MNIST and CIFAR-10 datasets, respectively. Here, all the weights are implemented using s-MTJs. Number of weights in the DNNs operating on MNIST and CIFAR-10 datasets are 476,400 and 28,044,032, respectively. The analysis is repeated with different E_b values for the s-MTJs.

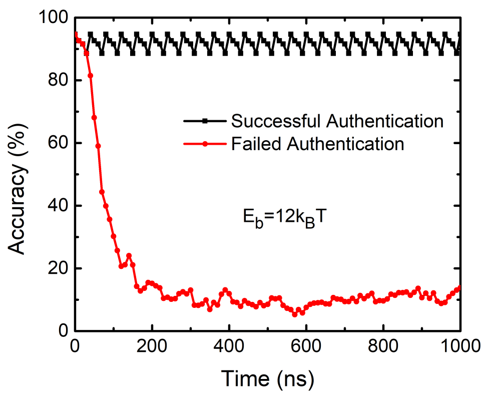


Fig. 10: Accuracy plots of s-MTJ based neural network. Here, we have shown the cases of both successful authentication and failed authentication. When there is no attack on the neural network, the accuracy will remain high.

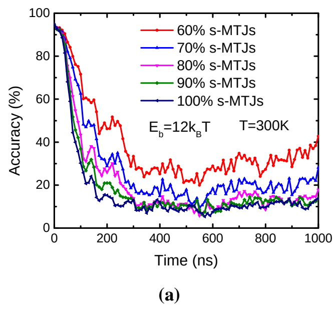
state over time. The total number of s-MTJs in this MINST-trained DNN is 476,400. We note that a superparamagnet has a 50% probability of flipping from its original state after an arbitrary time. Given that all the weights in this network are s-MTJ-based, the steady-state value for the % of total weights that have undergone a change after a long time is ∼50%.

Naturally, the accuracy degradation and thermal switching are more pronounced when the E_b of the s-MTJs is lower. It can be inferred that the recognition accuracy decreases with the increasing number of s-MTJs that have deviated from their initial state. We note that the recognition accuracy comes down to ∼10% when approximately half of the s-MTJs have switched stochastically. From Fig. 9(a,b), it can also be observed that the time taken for the accuracy of the DNN to decrease can be controlled by varying the E_b of the nanomagnets used in the s-MTJs. Larger E_b results in longer retention times for the nanomagnets, which causes a slower rate of accuracy degradation in the DNN. All the features, as mentioned above, are also applicable for the analysis of the CIFAR-10 dataset, shown in Fig. 9(c,d).

Under **normal operation**, when the system is not under attack, it is essential to reprogram the s-MTJs that have flipped attack, it is essential to reprogram the s-MTJs that have flipped from their initial state, to maintain the expected accuracy. Before the accuracy drops below that threshold, the synaptic weights can be updated, if device is in normal operation mode and not under attack. An authentication check is performed to ensure that the system is not under attack, and an all-clear signal is generated if the check is successful. The weight update proceeds only upon the receipt of the all-clear signal. This scheme is showcased in Fig. 10, where a failed authentication results in accuracy degradation, however, a successful authentication maintains a minimum accuracy of ∼90%. The actual hardware implementation of the authentication check can be realized by leveraging cryptographically secure mesh structures [45], or resistance/capacitance sensor arrays [46], [47]. Active shield structures such as those demonstrated in [45] can sense invasive probing and tampering attacks. A mesh shield over the input/ output terminals of the s-MTJ crossbar array can be used to thwart the attacker from querying the DNN. As soon as the attacker attempts to apply an input through external leads, the data bit sequence through the mesh wires will get altered, thus detecting the incursion.

For the MNIST-trained DNN considered in Fig. 9, a large number of s-MTJ weights have to be refreshed periodically, which requires a substantial amount of current to be drawn from the supply. This increases power consumption and can also significantly affect the reliability of the chip due to electromigration and IR-drop effects [48]. So it becomes essential to identify methods for reducing the number of s-MTJs that need to be updated at any given time. Next, we explore some of these techniques along with other factors that may affect the accuracy degradation in the DNN. It will be shown that, s-MTJs only in one layer of the DNN is sufficient to make the output of the stochastic. This allows for the majority of the memory elements to be implemented using conventional non-volatile MTJs that do not require frequent weight update.

4.1.1. Random distribution of s-MTJs: To lower the power dissipation expended during weight refresh, we consider only a fraction of the weights in the DNN to be composed of s-MTJs. We refer to this as the *partial stochastic configuration*, as opposed to the *fully stochastic configuration* before. These s-MTJ-based weights are randomly distributed across the multiple layers of the DNN, and the rest of the weights are implemented with stable MTJs. The DNN is trained on the MNIST dataset. We can see from Fig. 11(a) that, even with just 80% of the weights implemented using s-MTJs, the neural network accuracy drops to 10%, with time. From Fig. 11(a,b), it is also worth noting that, approximately only 4% and 10% of the s-MTJ-based weights (where s-MTJs account for 80% of all weights) have switched from their initial states, before recognition accuracy drops below 90% and 80%, respectively. Hence, during normal operation, only 4%-10% of the weights



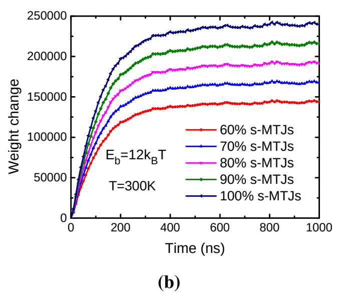
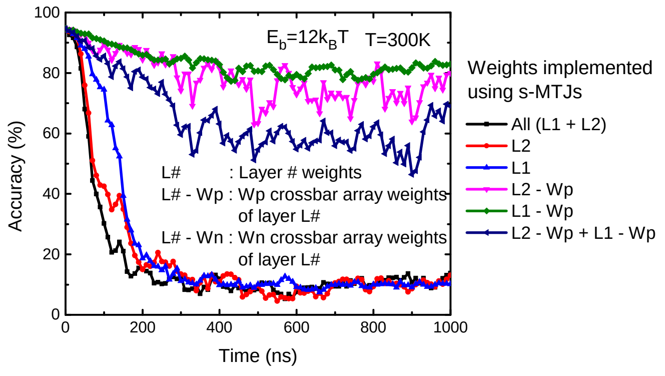


Fig. 11: (a) Recognition accuracy and (b) s-MTJ state variation over time observed on multi-level DNN designed to operate on MNIST dataset. Here, only a fraction of weights are implemented using s-MTJs, and these are randomly distributed throughout the network. The analysis is repeated with varying percentage of total weights implemented using s-MTJs.

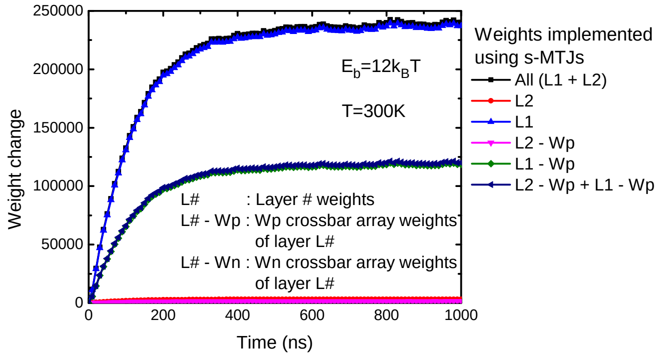
need to be updated periodically, and the rest of the weights need to be programmed only at the beginning of the operation. For this reason, it makes sense to design most of the s-MTJs with larger retention times (higher E_b). In this way, the number of weights that need to be updated during normal operation, at any given time, can be reduced.

4.1.2. Specific placement of s-MTJs: In contrast to the previous sub-section, where s-MTJs were placed randomly throughout the DNN, we could also allocate them to specific layers where they are likely to have a larger impact on the accuracy degradation, and hence, the overall security of the scheme. To identify these specific regions for the placement of s-MTJs, we conduct a study on an MNIST-trained DNN. The results of this study are shown in Fig. 12, where we can see that the accuracy of the network drops to 10% even with s-MTJs used only in the second layer. Placing s-MTJs only in the first layer results in the same accuracy degradation. However, the total number of weights in the second layer is significantly lower than the first layer, implying fewer s-MTJs to be implanted, and therefore lower power consumption. From Fig. 12(a,b), we also observe that using s-MTJs only for the weights in the positive crossbar array does not result in adequate accuracy degradation or weight change. Results of similar analysis on CIFAR-10 dataset, shown in Fig. 13, also substantiate the fact that accuracy degradation can be achieved with s-MTJs used in only one layer of the DNN.

4.1.3. Impact on non-linearity: Although the selective placement of s-MTJs, as described in Sec. 4.1.2, achieves accuracy degradation while lowering the weight refresh cost, the although the selective placement of s-MTJs, as described in Sec. 4.1.2, achieves accuracy degradation while lowering the weight refresh cost achieves accuracy degradation while lowering the weight refresh cost aaaaaaaaaaaaaa aaaaaaaaaa, the rate of accuracy degradation or non-linearity is an important metric to be considered as well. Here, we define the non-linearity in the waveform as the ratio of the slopes of the recognition accuracy during the decrease from 90% to 50% with the initial slope, as shown in Fig. 14. A higher non-linearity implies that the recognition accuracy remains high for a larger amount of time, followed by a sharper descent. Implanting s-MTJs in only one layer of the DNN may not be sufficient to achieve the required non-linearity. Comparing

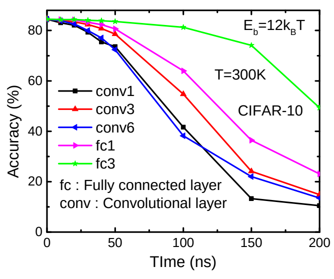


(a)

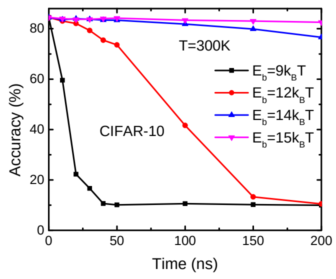


(b)

Fig. 12: (a) Recognition accuracy and (b) s-MTJ state variation over time, observed on the MNIST-trained multi-layer DNN when s-MTJs are used to implement weights only in specific regions of the DNN.



(a)



(b)

Fig. 13: Recognition accuracy variation over time when s-MTJs are used (a) only in a particular layer and (b) only in first layer (*conv1*) of the multi-level DNN, designed to operate on CIFAR-10 dataset.

Fig. 13 and Fig. 9(c), we can see that the non-linearity in the accuracy waveform is higher with s-MTJs in all the layers.

4.1.4. Significance of number of layers: Next, we examine the effect of the number of layers in the DNN on the rate of thermally-induced accuracy degradation in an MNIST-trained network. Here, all the weights are implemented using s-MTJs. As shown in Fig. 15, the recognition accuracy decreases more sharply (higher non-linearity) with a larger number of layers in the DNN. Multiplying two non-linear functions results in a product with even more non-linearity. Similarly, the non-linearity caused by one layer of the DNN, when passed to the next layer, amplifies the non-linearity [24]. Therefore, the larger the number of layers, the larger the non-linearity.

4.1.5. Significance of number of neurons in a layer: Figure 16 shows the variation of recognition accuracy with

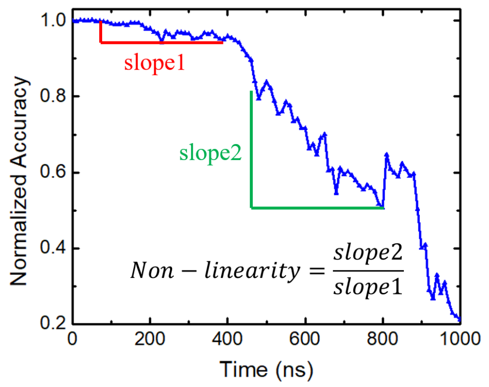


Fig. 14: Method used to quantify non-linearity in the waveform. It is defined as the ratio of slope during the decrease from 90% to 50%, with the initial slope.

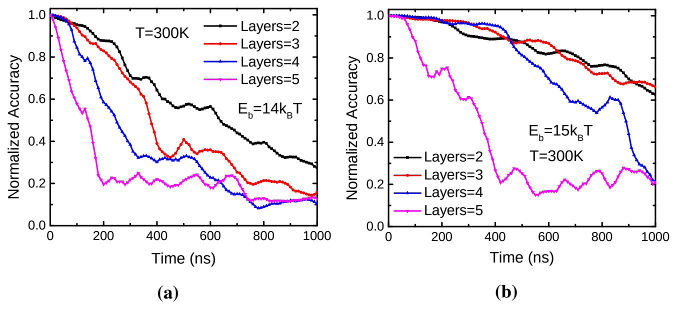


Fig. 15: Recognition accuracy variation observed on multi-level DNN with different number of layers, designed to operate on MNIST dataset. The analysis was done using superparamagnets with (a) $E_b = 14k_BT$ and (b) $E_b = 15k_BT$.

time of two-layer DNNs with a different number of neurons in the intermediate layer. We see that the recognition accuracy on the MNIST dataset starts dropping earlier when only 100 neurons are present in the intermediate layer. With less number of neurons within a layer, the number of synaptic weights will also be low. This means that a change in the state of even one of those weights can have a larger impact on the recognition accuracy of the DNN. This is the reason for the early decrease in recognition accuracy with a lesser number of neurons in a layer. However, the non-linearity in the waveform is larger when there are more number of neurons. Minimum recognition accuracy is found to be achieved by all the networks at approximately the same time.

The variation of the non-linearity metric with the number of layers in the DNN and the number of the neurons in the the variation of the non-linearity metric with the number of layers in the DNN and the number of the neurons in the intermediate layer is tabulated in Table III.

4.1.6. Significance of temperature: With a reduction in temperature, the thermal energy decreases, and therefore the time taken for the nanomagnets to flip their state (retention time) increases. This results in a slower rate of decrease in recognition accuracy with a decrease in temperature. This can also be verified from the analysis shown in Fig. 17, for an MNIST-trained DNN.

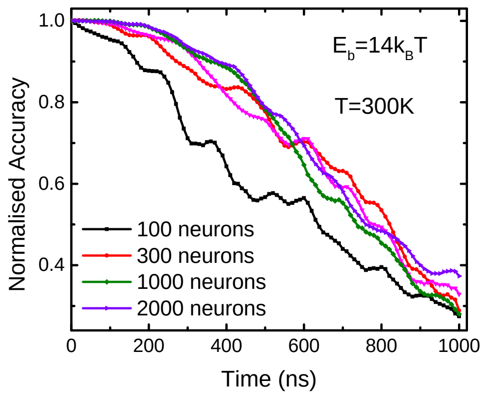


Fig. 16: Recognition accuracy variation over time for a two-layer multi-level DNN designed to operate on the MNIST dataset. The analysis was performed by varying the number of neurons in the intermediate layer.

TABLE III:

Non-linearity in recognition accuracy variation over time for different neural networks

No. of layers	Non-linearity
2	1.28
3	2.05
4	5.61
5	6.48

No. of intermediate neurons	Non-linearity
100	1.67
300	2.23
1000	3.94
2000	4.02

4.2. Binary neural network

The analysis in Sec. 4.1 is repeated for the binary DNN architecture of Fig. 7(b), and the results are presented here. Figure 18 highlights the evolution of recognition accuracy and s-MTJ state variation over time for binary DNNs designed for MNIST and CIFAR-10 datasets. This considers a fully stochastic W_N crossbar array, where all the weights are implemented using s-MTJs. However, since there is only one programmable crossbar array, in this case, the number of stochastic weights that require periodic update is half of that of the multi-level DNN architecture.[1]

For the partial stochastic configuration, where s-MTJs are implanted randomly across W_N, the corresponding results are implanted randomly across W_N, the corresponding results are

[1]Number of s-MTJ-based weights here are 238,200 and 14,022,016 for the MNIST- and CIFAR-10-trained networks, respectively, and ~50% of them switch after arbitrary time, as shown in Fig. 18(c,d).

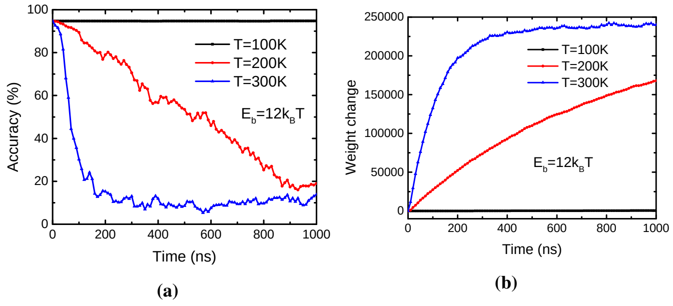


Fig. 17: (a) Recognition accuracy and (b) s-MTJ state variation over time on multi-level DNN for MNIST dataset operated at different temperatures.

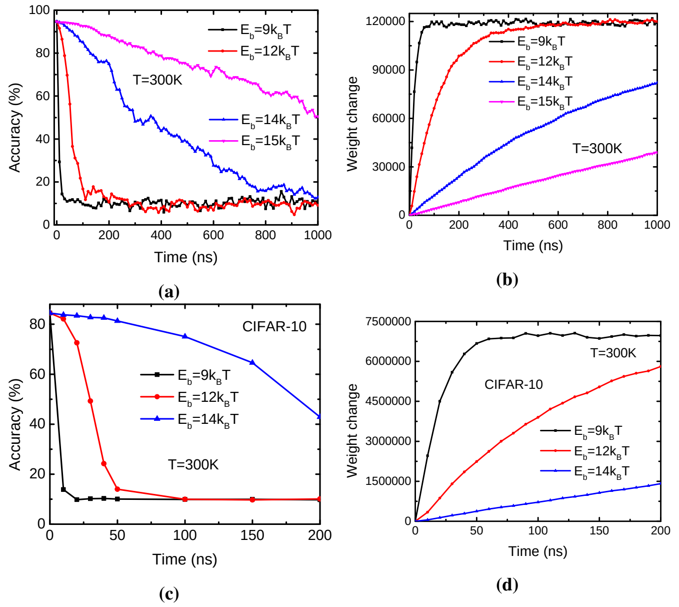


Fig. 18: (a,c) Recognition accuracy and (b,d) s-MTJ state variation over time observed on binary DNN, designed to operate on MNIST and CIFAR-10 dataset, respectively. Here, all the weights are implemented using s-MTJs. The analysis is repeated with different E_b values for the s-MTJs.

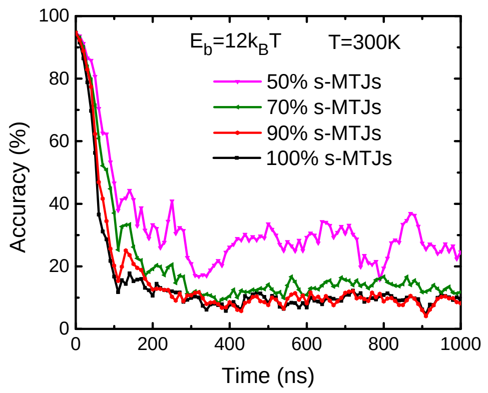


Fig. 19: Recognition accuracy variation over time, observed on binary DNN for MNIST dataset, when only a fraction of weights are implemented using s-MTJs. The s-MTJs are randomly distributed throughout the network. The analysis is repeated by varying the percentage of s-MTJ-based weights.

implanted randomly across W_N, the corresponding results are implanted randomly across W_N, the corresponding results are shown in Fig. 19. It is seen that, for an MNIST-trained binary DNN, implementing just 70% of the weights in W_N using s-MTJs is enough to achieve a minimum recognition accuracy of $\sim$10%. This is better than the case in Fig. 11(a), and therefore, an s-MTJ-based binary DNN will consume less power during weight update than an s-MTJ-based multi-level DNN.

With the multi-level DNN and binary DNN being similar in all aspects except for the number of s-MTJs required, the rest of the results showcased for multi-layer DNNs in Sec. 4.1 are also valid for binary DNNs. For instance, the analysis in Fig. 13, repeated for binary DNNs in Fig. 20, provides the same insights, albeit with different non-linearity. We note here that one drawback of the binary DNN architecture entails higher susceptibility to process variation-induced error. Since

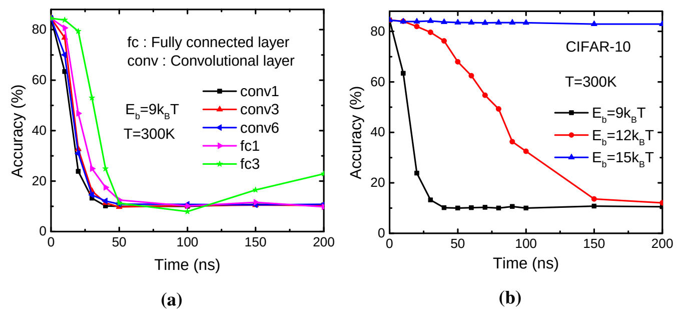


Fig. 20: Recognition accuracy variation over time, when s-MTJs are used (a) only in a particular layer and (b) only in first layer (*conv1*) of the CIFAR-10-trained binary DNN.

it possesses only one programmable crossbar array in a single-ended fashion, process variations might limit the maximum achievable recognition accuracy, in the absence of thermal weight drift [44].

4.3. Stochastic input neural network

Figure 21 presents the time evolution of the recognition accuracy and s-MTJ state variation for a stochastic input DNN, trained on MNIST and CIFAR-10 datasets.[2] The main advantage of this DNN configuration is that the number of s-MTJs that need to be updated periodically is significantly smaller compared to the previous two networks. The number of s-MTJs required here is twice the number of components in the input vector. Therefore, with 28×28-sized MNIST images and 32×32×3-sized CIFAR-10 images, the number of s-MTJs for the MNIST-trained DNN is 28×28×2 = 1568, and that for CIFAR-10-trained DNN is 32×32×3×2 = 6144. From Fig. 21(b), we can see that stochastic input DNNs require about 100× and 200× less number of weight refresh operations than binary DNNs and multi-level DNNs, respectively. However, even with very few s-MTJs to be updated, the network's accuracy still drops to 10% with time. The pitfall of these stochastic input DNNs is that the non-linearity in the recognition accuracy waveform is not easily augmented.

4.4. Benchmarking and Comparison

Here, we benchmark the three DNN architectures explored for the *SCANet* framework in terms of their energy and power consumption for updating the stochastic weights. For the purposes of this benchmarking, we consider DNNs trained on the MNIST dataset only. Recall that when any s-MTJ-based DNN resumes operation after an arbitrarily long time, half of it's s-MTJs will be in incorrect states and have to be reset before the next processing cycle. The energy required for resetting these s-MTJs is denoted as "reset energy."

Under non-attack conditions (regular operation), the s-MTJs that switch to an incorrect state need to be regularly refreshed to maintain the minimum required recognition accuracy. The energy required for this update process is defined as the

[2]The rest of the simulations shown for the previous networks, including partial stochastic configuration and specific placement of s-MTJs in layers, are not applicable for this architecture.

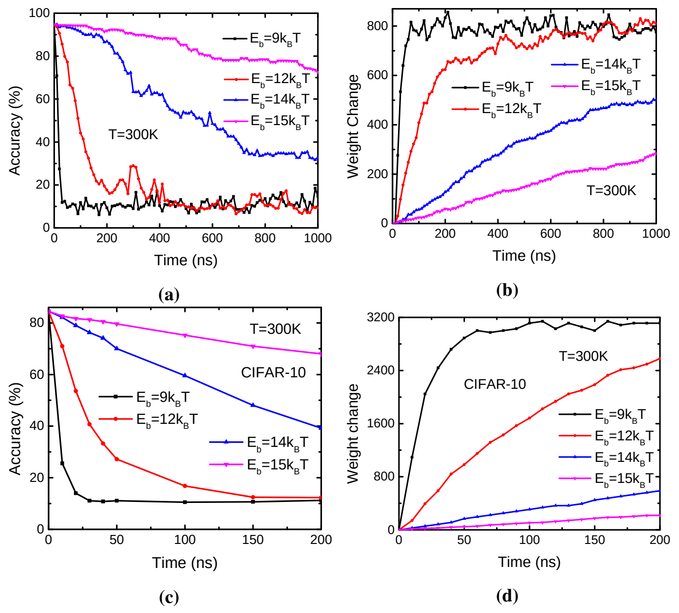


Fig. 21: (a,c) Recognition accuracy and (b,d) s-MTJ state variation over time, observed on stochastic input DNNs designed to operate on MNIST and CIFAR-10 datasets, respectively. The analysis is repeated with different E_b values for the s-MTJs.

"refresh energy." To calculate these metrics, we assume the minimum required recognition accuracy to be 80%, and the energy needed for updating the state of one s-MTJ to be 4.5 fJ [42]. The radar plot in Fig. 22 illustrates the benchmarking of the (i) reset energy, (ii) the refresh energy, and (iii) the power consumed in the refresh operation at three different E_b values, for the three DNN architectures. Depending on the E_b of the s-MTJs used, the time period between each refresh operation will be different, and therefore the average power consumed for the refresh operation will also vary. As expected, the larger the E_b of the magnet, the smaller the average power consumed for a refresh operation.

From the numbers in Fig. 22, we can see that refreshing the s-MTJ weights is much less energetically prohibitive in stochastic input DNNs compared to the other two networks. In Fig. 22, the refresh power calculation assumes that the minimum required recognition accuracy is 80%. In Table V, the energy and power consumption for different accuracy thresholds are tabulated. From the table, it can be seen that the power consumption does not increase much with the increase in accuracy threshold. Keeping a higher accuracy threshold implies that the weights of s-MTJs have to be updated more frequently. However, achieving a higher accuracy threshold also involves reducing the total number s-MTJs (hence less weights will deviate from their initial state) and therefore the energy consumption for a single update cycle is less.

Refresh latency can also have an impact on the overall performance of the DNN when the number of s-MTJs to be updated is large. When more number of weights have to be refreshed, more time has to be allocated for the refresh operation and less time is left for the operation of the DNN itself. This is illustrated with an example in section II of the supplementary section. A detailed comparison of the features

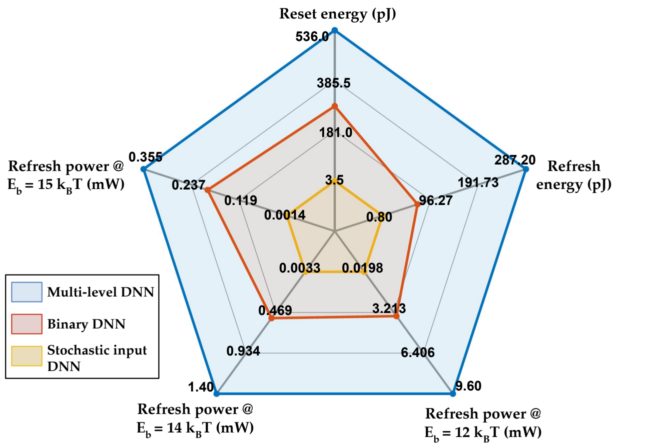


Fig. 22: Benchmarking plot for the weight refresh energy and power consumption of the three DNN architectures. Operating temperature T = 300K.

TABLE IV:
Comparison between the three neural network architectures

	Multi-level DNN	**Binary DNN**	**Stochastic input DNN**
Weights to be updated	High	Medium (∼0.5× multi-level DNN)	Low
Refresh power	High	Medium	Low
Refresh latency	High	Medium	Low
Non-linearity	Augmented by increasing layers of s-MTJs	Augmented by increasing layers of s-MTJs	Cannot be augmented (s-MTJs present only in one layer)
Process variation	Resistant (symmetry in the differential architecture)	Susceptible (information stored only in W_N crossbar) [44]	Resistant (manipulation only in input layer)

TABLE V:

Impact of accuracy threshold on refresh power consumption. Analysis is done on a three-layer DNN with s-MTJs implemented only in the output layer.

Accuracy threshold	$E_b = 12\,k_BT$		$E_b = 14\,k_BT$	
	Energy	**Power**	**Energy**	**Power**
95%	0.68 pJ	32.75 μW	0.39 pJ	5.27 μW
90%	1.26 pJ	28.04 μW	1.01 pJ	5.25 μW
85%	1.46 pJ	27.83 μW	1.28 pJ	4.82 μW
80%	1.46 pJ	27.83 μW	1.38 pJ	4.68 μW
70%	1.99 pJ	27.96 μW	2 pJ	4.38 μW

of the three architectures is elucidated in Table IV.

5. Advantages & Applications

In this section, we demonstrate the key differences and advantages of *SCANet* over prior work, which leverages memristors [24]. We limit the comparison to [24] since it is the only prior work that focuses on resistance drift in emerging NVM-based NNs. We also present a case study highlighting a practical application of our scheme in image processing.

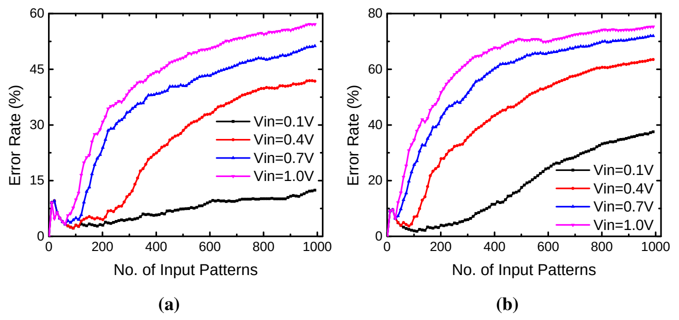


Fig. 23: Error rate variation with the number of input patterns applied on a memristor-based NN implemented using (a) naive architecture and (b) revised architecture. The two architectures studied for preventing replication attack was presented in [24]. The analysis given here was performed with different input voltage range.

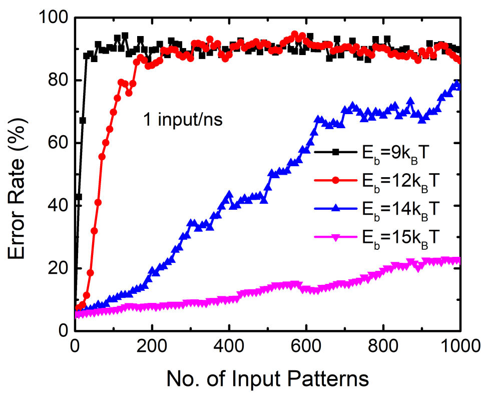


Fig. 24: Error rate versus number of input patterns of s-MTJ based neural network assuming a new input pattern is provided every $1\,\mathrm{ns}$.

5.1. State change without input

One of the significant advantages of using s-MTJs for weights is that the state of an s-MTJ can switch thermally, without any applied input voltage. Hindering this phenomenon might be intractable for the attacker, as will be seen in Sec. 6.1. However, this is not the case with memristor-based implementations, which exploit the obsolescence effect [24]. The resistance drift of memristors due to the obsolescence effect is dependent on the magnitude of the input voltage; the larger the magnitude of input voltage, the larger the resistance
larger the magnitude of input voltage, the larger the resistance
larger the magnitude of input voltage, the larger the resistance
larger the magnitude of input voltage, the larger the resistance
larger the magnitude of input voltage, the larger the resistance
larger the magnitude of input voltage, the larger the resistance
larger the magnitude of input voltage, the larger the resistance
drift. It is this resistance drift that causes the increase in the error rate of the memristor-based DNN. An advanced attacker might be able to gain sufficient I/O pairs by simply scaling down the input voltage applied to the DNN.

The results shown in Fig. 23 illustrate this input scaling attack. Here, the inputs applied to the DNN are gradually scaled down from 1V. The experiment is repeated for both the architectures described in [24]. From Fig. 23, we can conclude that, for a given error rate threshold, the number of I/O pairs obtained can be increased by scaling down the input voltages applied to the DNN.

Therefore, if the applied voltage to the memristors can be kept in the range of $\mathcal{O}$(mV), the input-induced resistance drift can be minimized, and only the negligible Néel retention drift will persist. In contrast, the temporal drift in s-MTJ resistance caused by thermal effects is inevitable and far more pronounced. The error rate of the s-MTJ based multi-level DNN is shown in Fig. 24. In this figure, we have assumed that a new input pattern is processed every $1\,\mathrm{ns}$. In case of s-MTJs, the temporal state change due to thermal energy occurs even if no input is applied. Therefore, if the input pattern is processed slower, only few number of patterns can be processed within a given error rate threshold.

From Fig. 24, we can see that, if the input pattern is applied every $1\,\mathrm{ns}$, with $E_b = 15k_BT$, the error rate decreases gradually. Therefore, the attacker might be able to obtain sufficient number of input patterns for replicating the neural network. However, with magnets of lower energy barrier used, the error rate increases faster and thus the attacker fails to obtain sufficient number of I/O pairs. Hence, the recommended E_b for the s-MTJ considered is $\leq 12k_BT$, which can be controlled by the designer.

5.2. Control over the rate of accuracy degradation

The E_b of superparamagnets is not only material-dependent but also depends on the magnet's volume (see Sec. 2). Therefore, it is possible to modify the E_b of the superparamagnetic weights during fabrication, thereby controlling the rate of accuracy degradation in the DNN.

5.3. Control over the memory elements to be updated

Using a mixture of s-MTJs and conventional stable MTJs for implementing the synaptic weights in the DNN, one can control the number of weights that need to be updated periodically (see Sec. 4.1.1). This not only helps in reducing the power consumption but also curtails the reliability issues due to IR drop and electromigration violations. The obsolescence effect in memristors does not afford this level of control. Hence, memristor-based NNs would require a large number of memory elements to be updated at a given time. This could be a bottleneck while designing DNNs, which can have more than a hundred million synaptic weights.

5.4. Case study: Object detection

We exhibit the efficacy of *SCANet* using an image processing example. Figure 25 shows the impact of thermally-induced weight degradation on a DNN designed for object detection. A single shot multibox detector is used for the object detection application [49], with an image from the PASCAL VOC dataset [50]. The DNN correctly identifies the object within the image as an airplane, in the absence of weight degradation. However, when the same DNN is constructed with s-MTJ-based weights, it cannot identify the object correctly due to weight drift. Note here that we consider the multi-level architecture (with s-MTJs in only one layer) for the detector. The total number of weights is 549,812,984,192, out of which 774,144 are implemented with sMTJs to obtain Fig. 25(b).

6. Attack Vectors & Defenses

In this section, we discuss the possible attack vectors for *SCANet* and the safeguards against them.

Threat Model: We consider a threat model similar to the military drone example in Fig. 1. The design house, foundry, test facility, and end-user are considered trusted entities within this military hardware framework. The attacker is someone outside the supply chain who manages to gain unauthorized access to the DNN via hardware theft. We assume that this attacker's motive is to reverse engineer the DNN in the military hardware through model replication attacks and that she/he possesses complete knowledge of the security measures implemented in the hardware. We also assume that the weight update process in the DNN is completely secure, with the attacker unable to identify the individual state to which each memory cell is programmed.

6.1. Reduction of temperature

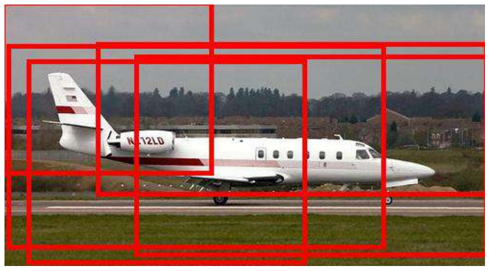
Arguably, the most effective attack against s-MTJ crossbar array-based DNNs would be one that leverages the manipulation of temperature. If the attacker can somehow reduce the operating temperature, then the stochastic weight corruption and the corresponding recognition accuracy degradation can be inhibited. This is evidenced by the results in Fig. 17. At a temperature of 100K, the recognition accuracy decreases only by a small amount. Therefore, the attacker can collect enough I/O pairs to conduct a replication attack.

However, the major challenge for the attacker here would be to cool the system before the states of all the s-MTJs start changing. In the practical scenario of a captured drone, this only affords a few nanoseconds from the moment of capture to the onset of weight corruption. After detecting capture, the drone can stop the weight refresh operation immediately, thereby rendering such a temperature-based attack futile.

6.2. Transfer Learning

Transfer learning is a well-known method used by NN designers when the amount of data available for training is small. It involves using a NN that was already extensively trained with a sufficiently large dataset. In this pre-trained NN, only the last layer's weights are updated based on the new learning algorithm. Using this technique, the designer can develop NNs with reasonable accuracy, even with a limited training dataset.

Figure 26 illustrates the impact of transfer learning. Recognition accuracy variation vs. the number of training images used is shown for two different two-layer DNNs. The first DNN consists of weights that were randomly initialized before training, whereas for the second DNN, weights in the first layer are pre-trained. As seen from the figure, the second DNN with a pre-trained first layer requires fewer images for training. Here, the first layer is pre-trained on the MNIST


(a)

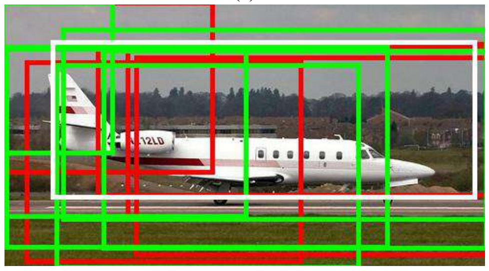

(b)

Fig. 25: Object detection (a) without and (b) with weight degradation. The red, green and white boxes are used to identify aeroplane, bottle and chair, respectively. With weight degradation, the DNN fails to resolve the object as an aeroplane, bottle or boat. The image is from the PASCAL VOC dataset, and the network used is a single shot multibox object detector [49], implemented using multi-level architecture.

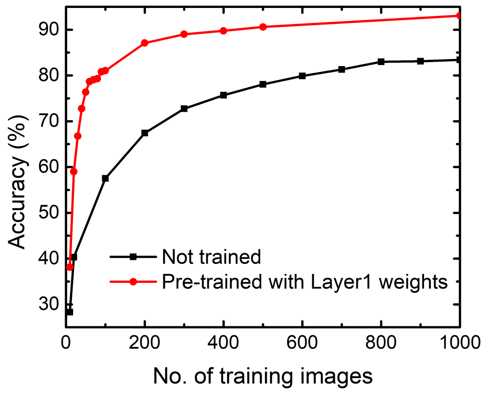


Fig. 26: Plot to explain the impact of transfer learning. Amount of training data required for two-layer neural network with pre-trained initial layer is much less than that for a neural network, with randomly initialized synapse weights.

dataset. However, as per the concept of transfer learning, even if the application is not the same, the weights in the initial layers can be similar, and therefore the number of images required for training can be reduced [51], [52].

If the attacker uses a pre-trained DNN model, she/he will have to query the captured drone very few times. However, we believe that the time interval for the onset of weight corruption, $\mathcal{O}(\sim 10$ ns), will still be too small to gather sufficient I/O pairs for a transfer learning-guided attack.

6.3. Reading the states of all the s-MTJs

Serially probing the states of all the s-MTJs in multi-level and binary DNNs, before the onset of weight degradation is intractable. Also, performing a parallel probing operation on millions of weights is beyond the capabilities of state-of-art probing equipment.

However, in the case of stochastic input DNNs, the synaptic weights are not implemented with s-MTJs. So the attacker can attempt to serially read the states of the entire crossbar array to recover partial information from the DNN. Nonetheless, even if the attacker recovers the states of all the MTJ weights, she/he still has to resolve significant challenges to replicate the DNN. Without the information on the states of the s-MTJs in the SI block, the attacker has to guess the correct sequence of input patterns to be applied to the DNN. We explain this with an example as follows.

Let $V_{\text{in},i}$ be one of the elements of the input vector that is fed to an SI block. Then, the attacker should be able to predict whether the output of this SI block is $V_{\text{in},i}$ or $-V_{\text{in},i}$. Therefore, if there are N such elements in the input vector ($i \in \{1, ..., N\}$), then the probability of attacker arriving at the correct solution for the entire array is 2^{-N}. The MNIST dataset consists of 28×28 sized images, and therefore the number of elements in the input vector is 784. This implies that, even if the attacker can read the states of all the MTJ-based weights in the crossbar, the corruption of s-MTJs in the SI block will afford the attacker a success rate of only 2^{-784}. This probability is very low, and for all practical purposes, can be ignored.

6.4. Logic Removal

Specifically, for the case of stochastic input DNNs, the attacker might attempt to remove the SI blocks altogether and mount an attack on the remaining crossbar arrays. As mentioned in Sec. 6.3, the crossbar arrays contain only half the information, and the attacker has a negligible chance of reverse-engineering the DNN without the knowledge about the randomized inputs from the SI blocks.

6.5. Leveraging bias in the neural network output

An attacker can exploit bias in the DNN output to decipher operational details. Even if the recognition accuracy has dropped due to the state change of s-MTJs, the attacker can identify the internal functionality if the DNN output shows bias towards the correct value. After successfully identifying the network's functionality, the attacker could combine this information with limited I/O response data to train a new NN to acceptable recognition accuracy standards. Therefore, it is essential to ensure that the DNN is free from output bias.

From Figs. 9, 18 and 21, we can see that there is no such bias in our DNN output. A DNN designed to operate on the MNIST dataset has ten possible outputs. Prior to learning, such a DNN with no bias will produce all the possible outputs with equal probability. So the expected ideal recognition accuracy of the system under attack is 10%. Without any state update, the recognition accuracy for all the DNN architectures will settle to ∼10%, as seen in Figs. 9(a), 18(a) and 21(a). We also note here that the s-MTJs themselves are fabricated without any bias, and hence ∼50% of all the stochastic weights will switch randomly after an arbitrary time. Owing to this stochastic switching behavior, the attacker cannot hope to gain any information about the initial/ final states of the s-MTJs.

6.6. Reliability of the superparamagnets

Another possible avenue for an attack is to exploit the potential variation of superparamagnetic properties with time. The Néel relaxation time of the s-MTJ-based weights will change with the deteriorating reliability of the superparamagnets. Over time, if the device properties change enough, causing the E_b of the superparamagnets to increase, the attacker might obtain sufficient I/O pairs for replicating the DNN. However, such an attack hinges on the condition that the drone has been in commission for a time long enough to cause reliability issues.

7. Conclusion

In this work, we show that *SCANet*, a novel design-for-trust technique for hardware DNNs, can be used to counter the reverse engineering of DNNs through the model replication attack. The synaptic weights, implemented using s-MTJs with E_b ∼9-15 $k_\text{B}T$, are shown to undergo uncontrolled thermal switching according to the Néel relaxation theory, resulting in severe accuracy degradation in the absence of periodic weight refresh. Thus, the attacker fails to obtain sufficient input-output responses from the DNN and cannot steal the protected DNN model. We conduct our analysis on three different NN architectures to optimize metrics such as maximum accuracy degradation, non-linearity, and refresh energy. Our optimizations yield non-linearity >5, with refresh energy ∼71 pJ for a binary DNN with 238,200 stochastic weights. The refresh energy can be reduced by order of magnitude to ∼0.8 pJ, by adopting the stochastic input DNN architecture. We perform detailed benchmarking to identify the trade-offs in power and security afforded for the three architectures. Further, we delineate the advantages of the proposed technique against prior work and highlighted use-cases for our protection scheme in object detection applications. We also exhaustively explore the various attack scenarios against *SCANet* and discuss possible countermeasures.

References

[1] Y. LeCun et al. Deep learning. *nature*, 521(7553):436–444, 2015.

[2] T. Brotherton and T. Johnson. Anomaly detection for advanced military aircraft using neural networks. In *2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542)*, volume 6, pp. 3113–3123 vol.6, 2001.

[3] F. Jiang et al. Artificial intelligence in healthcare: past, present and future. *Stroke and vascular neurology*, 2(4):230–243, 2017.

[4] B. Xie et al. Deep convolutional neural network for mapping smallholder agriculture using high spatial resolution satellite image. *Sensors*, 19(10):2398, 2019.

[5] S. Moran et al. Deep learning for medical image segmentation–using the IBM TrueNorth neurosynaptic system. In *Medical Imaging 2018: Imaging Informatics for Healthcare, Research, and Applications*, volume 10579, pp. 1057915. International Society for Optics and Photonics, 2018.

[6] M. Bojarski et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[7] P. Colangelo et al. Application of convolutional neural networks on Intel® Xeon® processor with integrated FPGA. In *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2017.

[8] W. Shi et al. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[9] M. Rostami et al. A primer on hardware security: Models, methods, and metrics. *Proceedings of the IEEE*, 102(8):1283–1295, 2014.

[10] U. Guin et al. Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain. *Proceedings of the IEEE*, 102(8):1207–1228, 2014.

[11] A. Chakraborty et al. Keynote: A disquisition on logic locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):1952–1972, 2020.

[12] H. Gomez et al. Defeating silicon reverse engineering using a layout-level standard cell camouflage. *IEEE Transactions on Consumer Electronics*, 65(1):109–118, 2019.

[13] S. Patnaik et al. Obfuscating the interconnects: Low-cost and resilient full-chip layout camouflaging. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[14] K. W. Cheuk et al. nnAudio: An on-the-fly GPU audio to spectrogram conversion toolbox using 1D convolutional neural networks. *IEEE Access*, 8:161981–162003, 2020.

[15] G. Chen et al. PhoneBit: efficient GPU-accelerated binary neural network inference engine for mobile phones. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 786–791. IEEE, 2020.

[16] J. Han et al. Hardware implementation of spiking neural networks on FPGA. *Tsinghua Science and Technology*, 25(4):479–486, 2020.

[17] Z. Mo et al. FPGA implementation for odor identification with depthwise separable convolutional neural network. *Sensors*, 21(3):832, 2021.

[18] I. Yeo et al. A hardware and energy-efficient online learning neural network with an RRAM crossbar array and stochastic neurons. *IEEE Transactions on Industrial Electronics*, 2020.

[19] Y. Choi et al. Vertical organic synapse expandable to 3D crossbar array. *Nature communications*, 11(1):1–9, 2020.

[20] S. N. Truong. Single crossbar array of memristors with bipolar inputs for neuromorphic image recognition. *IEEE Access*, 8:69327–69332, 2020.

[21] S. Yu. Neuro-inspired computing with emerging nonvolatile memorys. *Proceedings of the IEEE*, 106(2):260–285, 2018.

[22] S. Kannan et al. Security vulnerabilities of emerging nonvolatile main memories and countermeasures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(1):2–15, 2015.

[23] C. Yang et al. Security of neuromorphic computing: thwarting learning attacks using memristor's obsolescence effect. In *Proceedings of the 35th International Conference on Computer-Aided Design*, pp. 1–6, 2016.

[24] C. Yang et al. Thwarting replication attack against memristor-based neuromorphic computing system. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2192–2205, 2020.

[25] M. G. Augasta and T. Kathirvalavakumar. Reverse engineering the neural networks for rule extraction in classification problems. *Neural processing letters*, 35(2):131–150, 2012.

[26] S. J. Oh et al. Towards reverse-engineering black-box neural networks. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 121–144. Springer, 2019.

[27] W. Hua et al. Reverse engineering convolutional neural networks through side-channel information leaks. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2018.

[28] L. Batina et al. CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 515–532, 2019.

[29] A. Chen et al. *Emerging nanoelectronic devices*. John Wiley & Sons, 2014.

[30] B. N. Engel et al. A 4-Mb toggle MRAM based on a novel bit and switching method. *IEEE Transactions on Magnetics*, 41(1):132–136, 2005.

[31] M. Hosomi et al. A novel nonvolatile memory with spin torque transfer magnetization switching: spin-RAM. In *IEEE International Electron Devices Meeting*, pp. 459–462, 2005.

[32] N. Kurti. *Selected works of Louis Néel*. CRC Press, 2019.

[33] R. Faria et al. Low-barrier nanomagnets as p-bits for spin logic. *IEEE Magnetics Letters*, 8:1–5, 2017.

[34] L. Lopez-Diaz et al. Transition from ferromagnetism to superparamagnetism on the nanosecond time scale. *Physical Review B*, 65(22):224406, 2002.

[35] C. Tannous and J. Gieraltowski. The Stoner–Wohlfarth model of ferromagnetism. *European journal of physics*, 29(3):475, 2008.

[36] D. Vodenicarevic et al. Low-energy truly random number generation with superparamagnetic tunnel junctions for unconventional computing. *Physical Review Applied*, 8(5):054045, 2017.

[37] N. Rangarajan et al. A spin-based true random number generator exploiting the stochastic precessional switching of nanomagnets. *Journal of applied physics*, 121(22):223905, 2017.

[38] S. Ament et al. Solving the stochastic Landau-Lifshitz-Gilbert-Slonczewski equation for monodomain nanomagnets: A survey and analysis of numerical techniques. *arXiv preprint arXiv:1607.04596*, 2016.

[39] M. d'Aquino. Nonlinear magnetization dynamics in thin-films and nanoparticles. *http://wpage. unina. it/mdaquino/PhD_thesis/main/main.html*, 2004.

[40] M. J. Donahue and M. Donahue. *OOMMF user's guide, version 1.0*. US Department of Commerce, National Institute of Standards and Technology, 1999.

[41] A. Keshavarzi et al. Ferroelectronics for edge intelligence. *IEEE Micro*, 40(6):33–48, 2020.

[42] A. Sengupta et al. Magnetic tunnel junction mimics stochastic cortical spiking neurons. *Scientific reports*, 6:30039, 2016.

[43] M. Prezioso et al. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature*, 521(7550):61–64, 2015.

[44] Y.-F. Qin et al. Design of high robustness BNN inference accelerator based on binary memristors. *IEEE Transactions on Electron Devices*, 67(8):3435–3441, 2020.

[45] J.-M. Cioranesco et al. Cryptographically secure shields. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 25–31. IEEE, 2014.

[46] P. Tuyls et al. Read-proof hardware from protective coatings. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 369–383. Springer, 2006.

[47] D. J. Boday et al. Implementing carbon nanotube based sensors for cryptographic applications, August 5 2014. US Patent 8,797,059.

[48] Q. K. Zhu. *Power distribution network design for VLSI*. John Wiley & Sons, 2004.

[49] Z. Wang et al. BiDet: An efficient binarized object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2049–2058, 2020.

[50] M. Everingham et al. The Pascal Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[51] C. Tan et al. A survey on deep transfer learning. In *International conference on artificial neural networks*, pp. 270–279. Springer, 2018.

[52] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

Dinesh Rajasekharan received the B.Tech. degree in electronics and communication engineering and the M.Tech. degree in microelectronics and VLSI from the National Institute of Technology Calicut, Calicut, India, in 2013 and 2015, respectively. He is currently working toward the Ph.D. degree in electrical engineering at Indian Institute of Technology Kanpur, Kanpur, India. His research interest includes device and circuit design for neuromorphic computing applications.

Nikhil Rangarajan is a Postdoctoral Associate at the Division of Engineering, New York University Abu Dhabi, United Arab Emirates. He received the Ph.D. and M.S. degrees in Electrical Engineering from New York University, NY, USA, and the B.Tech. degree in Electrical and Electronics Engineering from National Institute of Technology Tiruchirapalli. His research interests include spintronics, nanoelectronics, device physics and hardware security.

Satwik Patnaik received the B.E. degree in electronics and telecommunications from the University of Pune, India, in 2010, the M.Tech. degree in computer science and engineering with a specialization in VLSI design from the Indian Institute of Information Technology and Management, Gwalior, India, in 2013, and the Ph.D. degree in electrical engineering from Tandon School of Engineering, New York University, Brooklyn, NY, USA in September 2020.

He is currently a Postdoctoral researcher with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA. His research delves into IP protection techniques, CAD frameworks for security, leveraging 3D paradigm for enhancing security, exploiting security properties of emerging devices, applied machine learning for hardware security, and side-channel evaluation.

Ozgur Sinanoglu is a professor of electrical and computer engineering at New York University Abu Dhabi. He obtained his Ph.D. in Computer Science and Engineering from University of California San Diego. He has industry experience at TI, IBM, and Qualcomm, and has been with NYU Abu Dhabi since 2010. During his Ph.D. he won the IBM Ph.D. fellowship award twice. He is also the recipient of the best paper awards at IEEE VLSI Test Symposium 2011 and ACM Conference on Computer and Communication Security 2013. Prof. Sinanoglu's research interests include design-for-test, design-for-security and design-for-trust for VLSI circuits, where he has more than 200 conference and journal papers, and 20 issued and pending US Patents. Prof. Sinanoglu is the director of the Center for CyberSecurity at NYU Abu Dhabi. His recent research in hardware security and trust is being funded by US National Science Foundation, US Department of Defense, Semiconductor Research Corporation, Intel Corp and Mubadala Technology.

Yogesh Singh Chauhan is a Professor with Indian Institute of Technology Kanpur (IITK), Kanpur, India. His group is involved in developing industry-standard compact models for GaN HEMT, FinFET, Nanosheet/Gate-All-Around FET, FDSOI transistors, negative capacitance FETs, and 2-D FETs. He is an editor of the IEEE Transactions on Electron Devices and a Distinguished chair of the IEEE Electron Devices Society. He is a member of the IEEE-EDS Compact Modeling Committee and a Fellow of the Indian National Young Academy of Science.