Security of Neuromorphic Systems: Challenges and Solutions

Beiye Liu, Chaofei Yang, Hai Li, Yiran Chen
Department of Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA, USA, 15261
{bel34, chy61, hal66, yic52}@pitt.edu

Qing Wu, Mark Barnell
Air Force Research Laboratory
Rome, NY, 13441-4505
{qing.wu, mark.barnell}@us.af.mil

***Abstract*—With the rapid growth of big-data applications, advanced data processing technologies, e.g., machine learning, are widely adopted in many industry fields. Although these technologies demonstrate powerful data analyzing and processing capability, there exist some security concerns that may potentially expose the user/owner of the services to information safety risk. In particular, the adoption of neuromorphic computing systems that implement neural network and machine learning algorithms on hardware generates the need for protecting the data security in such systems. In this paper, we introduce the security concerns in the learning-based applications and propose a secured neuromorphic system design that can prevent potential attackers from replicating the learning model. Our results show that the computation accuracy of the designed neuromorphic computing system will quickly degrade when no proper authorization is given, by leveraging the drifting effect of the memristor device.**

Keywords—Memristor; Neuromorphic; Security

I. Introduction

Machine learning are widely used in data processing to help the user better understand the underlying property of the data [13]. For example, one popular setup of machine learning application to deliver a prediction/inference based on a given set of features. In such applications, two phase of operations need to be conducted: 1) "training", which often requires a large set of data samples. Each sample consists of a set of variables (i.e., feature vector) and a label. Based on the training data, a specific type of machine learning model is selected and trained to analyze the new data; 2) "testing", where the trained model will make predictions based on the given testing sample. The testing sample may or may not contain new feature vectors that appear in the training samples. In general, the accuracy of the trained model is measured by its prediction accuracy.

The security of a data processing system can be critical in certain scenarios. For instance, the data processing model is often confidential and valuable to the system owner, putting the model under security risk if the system is unconstrainedly accessed by the public. As a practical example, assuming that a military unmanned aerial vehicles (UAV) is carry a signal processing system [10]. The task of the system is to detect some target patterns from raw data, e.g., images captured by the UAV in real-time, based on a trained classifier: when an image containing certain types of wanted targets is sent to the processing system, the model will make a positive prediction. Due to the bandwidth limitation of wireless communication and the large volume of the data, the classification/detection task has to be completed locally, say, by the UAV. Since the UAV can be captured by the unauthorized party, the classifier used on the UAV may be replicated by simply obtaining the I/O data, as we shall show in Section III. In other words, the attacker can replicate the classifier even without knowing any information about the model selection or the original training data. We refer to the security concern of the learning-based systems as "model privacy challenge".

Model privacy challenge is a very general problem in the application of learning-based system but hard to be overcame because of its unique problem setup. In the rest of this paper, we will show that the similarity between the replicated model and the original model severely depends on the prediction accuracy of the original model. Hence, there exists a tradeoff for the original model between the prediction accuracy and the risk of being replicated. Based on this observation, one possible approach to combat the model privacy challenge is to limit the number of the valid I/O pairs that the attacker can obtain from the original model. In fact, this approach established the foundation of our proposed secured neuromorphic computing system (NCS) design.

NCS recently emerged as a hot research area in circuit and system society [1][2]. NCS is a new hardware computing diagram that is specifically designed to efficiently realize machine learning algorithms. In this work, we mainly focus on a NCS receiving significant attentions is the one implementing large scale neural networks on top of memristor crossbars [3][7]. To solve the model privacy challenge of memristor-based NCS, in this work, we propose to utilize the drifting property of the memristor to build a NCS that has "forgetting effect". Even the NCS is captured by unauthorized party, the functionality of the system will quickly decays after very limited number of tests. Experiments show that our proposed NCS can effectively solve the model privacy challenge by "forgetting" the computing model before attacker accomplishes a successful replication.

II. Preliminaries

A. Memristor Basics

Predicted by Prof. Leon Chua in 1971 [8], memristor is the 4th fundamental circuit element whose resistance is determined by the historical profile of the electrical excitations through the

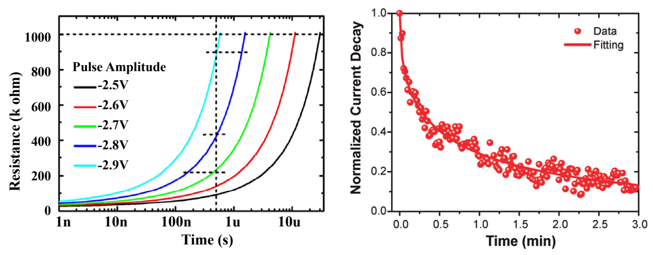


Figure 1: (a) Memristor device [6]. (b) Memristor forgetting effect [11].

device. In other words, the resistance state of a memristor can be programmed by applying current or voltage. During *reset* process, the memristor switches from low resistance state (*LRS*) to high resistance state (*HRS*). As shown in Figure 1 (a), the resistance of a memristor can be programmed to an arbitrary state in between the *LRS* and the *HRS* by controlling the programming voltage amplitude and duration. During *set* process, the memristor switches from *HRS* to *LRS* [6].

B. Drifting Effect

The drifting effect of memristor comes from two sources. The first one is the intrinsic retention property of the device. And the second one is sensing-induced drifting. The retention property of memristor device is discussed in [11]. It shows that after the device is programmed to certain resistance state, the resistance will change over time. To demonstrate this property, a memristor device was stimulated with short voltage pulses and the retention data was recorded by reading the current with low voltage pulses every second for total 3 minutes (Figure 1 (b)). The intrinsic retention property is hard to control since it is related to the material switching mechanism. However, the forgetting speed can be controlled by choosing the sensing voltage amplitudes and durations. As shown in Figure 1 (a), the resistance change of a memristor is a continuous procedure that can be described by:

$$\Delta R = f(v,t). \tag{1}$$

Here, v and t are the programming voltages and duration that are applied on the device, respectively. Thus, the resistance can be slowly changed by small sensing pulses, and the changing speed can be controlled by carefully choosing v and t.

C. Memristor Crossbar Based NCS

Figure 2 (a) depicts a conceptual overview of a neural network that can be directly mapped onto one core of the NCS. Two layers of neurons are fully connected by one layer of synapses. The output neurons collect the information from the input neurons through the network of synaptic connections and process them with a transformation function. The synapses multiply the signal transferring on them with different synaptic weights. In general, the relationship between the value of the

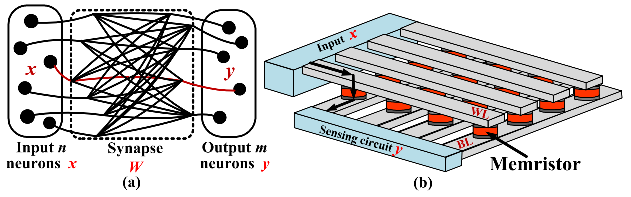


Figure 2: (a) Conceptual overview of a neural network [3]. (b) Circuit architecture of a memristor crossbar-based NCS [12].

input vector x and the output vector $\boldsymbol{y}$ can be described as [1]:

$$\boldsymbol{y_n} = f(\,\boldsymbol{W_{n\times m}} \cdot \boldsymbol{x_m}\,). \tag{2}$$

Here the connection weight matrix $\boldsymbol{W_{n\times m}}$ denotes the synaptic strengths between the two layers of neurons, e,g., $\boldsymbol{x}$ and $\boldsymbol{y}$. The matrix-vector multiplication shown in Equation (2) is one of the fundamental operation of a neural network. Because of the structural similarity, a memristor crossbar can serve as an efficient platform for executing matrix-vector multiplications . In multi-layer neural network design, multiple memristor crossbar are connected to match the network topology.

Testing: The operation defined by Equation (2) is exactly the feedforward "testing" operation of a traditional neural network model. As shown in Figure 2(b), during the testing process of a memristor-based NCS, $\boldsymbol{x}$ is represented as a vector of voltage signals applied to the word-lines (WLs) of the memristor crossbar while the bit-lines (BLs) are grounded. The current sensed from the bottom of each bit-line will be converted to the output voltage vector $\boldsymbol{y}$ by a specially designed sensing circuit. The sensing circuit can be a CMOS analog module [7] or a domain wall emerging device [2] based on the transformation function requirement.

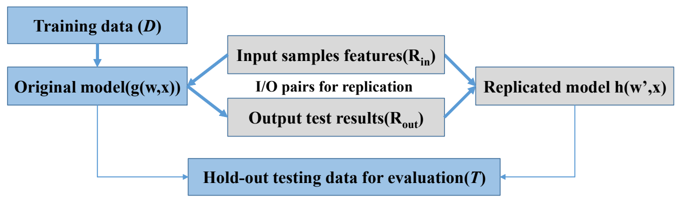
Training: Another basic operation of NCS is "training". Unlike the definition of "training" in traditional machine learning domain, the training operation of NCS is composed of 1) Calculating the value of each connection in $\boldsymbol{W_{n\times m}}$ and 2) Programming the memristors in the crossbar to the resistance states representing the weight matrix $\boldsymbol{W_{n\times m}}$. In programming a memristor, the voltages of the WL and BL connected to the target memristor are set to $+V_{bias}$ and GND, respectively, while the other WLs and BLs are connected to half select voltage, i.e., $+V_{bias}/2$. Hence, only the target memristor applied with the full V_{bias} above the threshold voltage will be successfully programmed [6]. Note that the programming voltage amplitude V_{bias} and duration must be carefully selected to ensure the resistance of each memristor $\boldsymbol{M_{ij}}$ satisfies $1/\boldsymbol{M_{ij}} = \boldsymbol{W_{ij}}$ [5].

III. Model Privacy Challenge

As aforementioned, the NCS models may need to maintain confidential to the unauthorized party. However, as pointed out in [14], there exist some simple but efficient approaches to replicate such models, causing potentially severe security issues in such applications.

A. Model Replication

Assume that the function of the original model, i.e., $g(w,x)$, is a classification model whose details are not accessible to the public. As shown in Figure 3, all the information represented by the blue blocks are confidential. Without loss of generality, we assume the function of the model can be described by:



Training data (D)
Original model(g(w,x))
Input samples features(R_in)
I/O pairs for replication
Output test results(R_out)
Replicated model h(w',x)
Hold-out testing data for evaluation(T)

Figure 3: Training and replication of the confidential model.

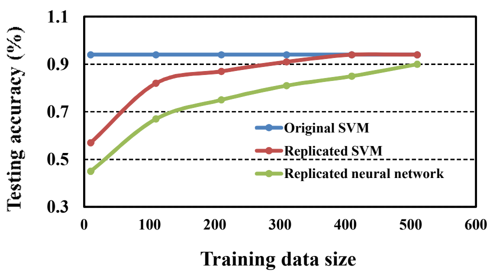


Figure 4: Replication quality (test accuracy) vs. number of I/O pairs.

$$g(\boldsymbol{w,x})=p(y=y_i \mid \boldsymbol{w},x),\ i=1\cdots n. \quad (3)$$

Here, $\boldsymbol{w}$ is a vector of the parameters of the original model. $\boldsymbol{x}$ is the input vector of the features. In Equation (3), y_i is the *i-th* class that a sample can be assigned to. The probability functions $p(y=y_i \mid \boldsymbol{w},x)$ are defined by the structure of the original model, e.g., a feedforward neural network or a naïve Bayesian model. The training data "***D***" is also hold only by the original system owner. After the training, the original model "$g(\boldsymbol{w,x})$" becomes available to process the unknown data.

We assume there is an attacker tries to attack the privacy of the original model, i.e., replicating the confidential original model through function approximation. Without knowing any information about the original model, a possible attacking procedure that the attacker may take consists of the following steps, as summarized in Figure 3:

Step 1 – Construction of "I/O pairs": The training data for model replication can be constructed by repeatedly testing the original trained model as a normal user. For example, the attacker can simply submit *m* test samples "$\boldsymbol{R_{in}}$" to the original trained model and collect the corresponding test results "$\boldsymbol{R_{out}}$". The paired features/results data "[$\boldsymbol{R_{in}}$, $\boldsymbol{R_{out}}$]" can be then used as the training set for the model replication.

Step 2 – model selection: The attacker needs to select a processing model to replicate the original model. Surprisingly, we found that selecting a model different from the original one does not necessarily hurt the result of model replication. After the I/O pairs are constructed, the training of the replicated model can be conducted routinely.

B. Case Study

The example we use here is a hand write digits classification problem from scikit-learn [13]. Each data sample contains a feature vector of 64 values corresponding to an image of 8×8 and also a label from 0-9 (10 different digits). The data samples are divided to three groups, e.g., 500 training samples "***D***" for the original model, 500 samples "$\boldsymbol{R_{in}}$" for the replication, and 300 testing samples "***T***". For fairness, we the same testing samples to for both original and replicated models.

The original model we use here is a support vector machine (SVM). The testing accuracy is defined as the percentage of correctly classifying the testing samples "***T***". Trained with the sample set "***D***", the original model achieves a testing accuracy of 0.94. We then send $\boldsymbol{R_{in}}$ to this well trained original model to generate $\boldsymbol{R_{out}}$. By combining [$\mathbf{R_{in}}$, $\mathbf{R_{out}}$], we construct a group of I/O pairs that consists of 500 samples for replication. Figures 4 shows the relation between replication quality

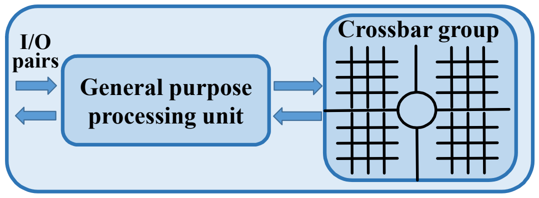


Figure 5: Memristor crossbar based secured neuromorphic platform.

and the number of I/O pairs that are used for replicated model training. Here the replication quality is measured by comparing the test accuracy of the replicated model and the original model on the test samples. Two different models – SVM and neural network, are selected for model replication. As depicted in Figure 4, the replicated SVM model reaches an accuracy very close to the original model when all 500 I/O pairs are used for training. The reason that neural network does not reach the similar accuracy could be the limited number of replication samples obtained from the dataset. Nonetheless, this example shows that replication can be done without knowing details of the original model or training data.

IV. SECURED NCS DESIGN

A. Overview

Figure 5 depicts the conceptual diagram of our proposed memristor-based secured NCS. The basic idea is to limit the number of the valid I/O pairs that unauthorized party can obtain from the hardware system. Take the aforementioned UAV application as an example, the weighted connections of a neural network classifier is implemented on memristor crossbars. Due to the "drifting effect" of the memristor, the accuracy of the classifier will degrade with the number of testing. For security purpose, we define two modes of UAV functioning as "safe mode" and "unsecured mode".

Safe mode: The UAV is in "safe mode" when the control center can track and maintain trustable connection with it. We can embed some validation data set in the hardware so that the functionality of the classifier can be monitored by the validation accuracy. Once the accuracy drops below certain threshold, the memristors are refreshed or calibrated based on the standard weights sent by the remote control center.

Unsecured Mode: The UAV can be potentially captured by the unauthorized party [10], which is considered as attackers. If the UAV is not in "safe mode", the memristors will no longer be calibrated and the NCS quickly loses its accuracy after certain number of testing cycles.

B. NCS Design Based on Forgetting Effect

By leveraging the memristor's forgetting effect, we may design a secured "volatile" NCS. For example, when input signals are sent to NCS, the voltages applied on the word lines are shown on left side of Figure 6. Positive voltages are applied to the 1st, 3rd and 4th rows and resistances of the corresponding memristors in the same rows are shifting to positive direction. Negative voltage is applied to the 2nd row and the resistance is shifting to negative direction. The resistance change across the array can be found on right side of Figure 6.

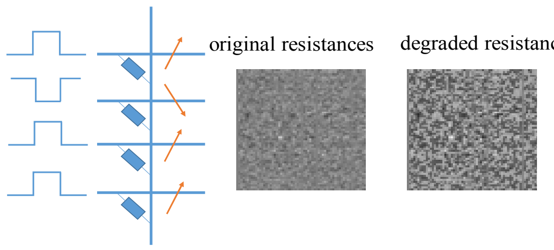


Figure 6: Memristor resistance change.

The resistance shift leads to the current change in bit lines, which further affects the computation of next layer and directly deteriorate the final classification results. The NCS starts to forget what it memorizes.

V. EXPERIMENT

A. *Testing Accuracy Degrades with the Testing Number*

We build a multilayer feedforward neural network with one hidden layer in MATLAB. The whole structure can be found in Figure 7.

The transfer function after the hidden layer is tan*sig*:

$$\mathrm{f}(net) = \tan sig(net) = \frac{2}{1+e^{-2*net}} - 1 \cdot \quad (4)$$

Our training set contains 500 samples with a feature size of 64 and 500 labels with a class number of 10, respectively. Our testing set contains 300 samples and labels. We assume the degradation of weights matrix is linear and take into account the effects from random inputs, which is shown as follows:

$$\mathrm{w} = w + sign(w) * rate * input \cdot \quad (5)$$

Here the variable *rate* is a manually chosen parameter indicating the degradation speed and the input is generated by a random number which cover most testing input sample cases. To evaluate the system, we use error rate and mean square error here as our criteria:

$$\mathrm{mse} = \frac{1}{n}\sum_{i=1}^{n} | y_i - t_i |^2 \cdot \quad (6)$$

Our network is simulated with *rate* = 0.001 and ran 500 times. The result is plotted in Figure 8. With an initial MSE of 0.0280, it increases to 0.6093 after 500 simulations. The initial error rate is 11.00% and it quickly increases to an unacceptable level of 45.67% after 500 cycles. If we set 20% as the threshold to determine if the system is functional, then the system becomes malfunctioned being tested for 250 times.

In conclusion, a secured memristor-based NCS may be built by benefitting from the forgetting effect of memristors.

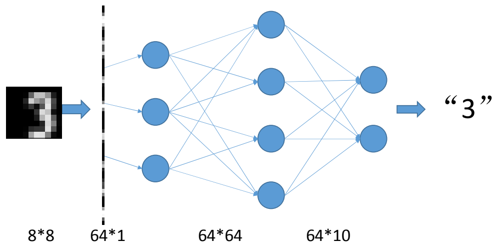


Figure 7: Neural network structure.

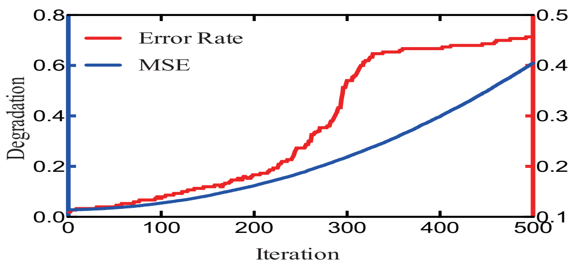


Figure 8: Error rate and MSE degradation.

Without authorized calibrations, the system cannot make accurate decisions after being used for certain times and thus, the model of the system is protected from being stolen by attackers.

VI. CONCLUSION

In this paper, we introduce the security concern, i.e., model privacy challenge, in machine learning based data processing system. In some scenarios, this concern severely limit the deployment of these powerful technologies. We then propose a solution to protect the original model from being replication by leveraging the forgetting effect of memristors. Our initial results demonstrates the feasibility of the proposed technique.

VII. ACKNOWLEDGEMENT

We thank for the support from grants NSF CNS-1253424, NSF XPS-1337198, and AFRL FA8750-15-2-0048.

REFERENCES

[1] B. Liu, et al., "The Circuit Realization of a Neuromorphic Computing System with Memristor-Based Synapse Design," Neural Information Processing, 2012.

[2] M. Sharad, et al., "Ultra Low Power Associative Computing with Spin Neurons and Resistive Crossbar Memory", DAC, 2013.

[3] M. Hu, et al., "Hardware realization of BSB recall function using memristor crossbar arrays", DAC, 2012.

[4] J. Liang, et al., "Cross-Point Memory Array Without Cell Selectors—Device Characteristics and Data Storage Pattern Dependencies", IEEE Trans. on Electron Devices, 2010.

[5] RE. Pino, et al., "Statistical memristor modeling and case study in neuromorphic computing," DAC, 2012.

[6] S. Yu, "Investigating the switching dynamics and multilevel capability of bipolar metal oxide resistive switching memory", APL, 2011.

[7] B. Liu, "Digital-assisted noise-eliminating training for memristor crossbar-based analog neuromorphic computing engine," DAC, 2013.

[8] L. Chua, "Memristor-the missing circuit element", IEEE Trans. On Circuit Theory, 1971.

[9] S. R. Lee, et al.,"Multi-level Switching of Triple-layered TaOx RRAM with Excellent Reliability for Storage Class Memory," Symposium on VLSI Technology, 2012.

[10] Brad Lendon, "Iran says it built copy of captured U.S. drone," CNN, 2014.

[11] Ting Chang, Sung-Hyun Jo, and Wei Lu, "Short-Term Memory to Long-Term Memory Transition in a Nanoscale Memristor," ACS Nano 5 7669, 2011.

[12] B. Liu, et al., "Reduction and IR-drop compensations techniques for reliable neuromorphic computing systems," ICCAD, 2014.

[13] Fabian Pedregosa, "Scikit-learn: Machine Learning in Python," The Journal of Machine Learning Research, 2011.

[14] B. Liu, "Cloning your mind: security challenges in cognitive system designs and their solutions," DAC, 2015.