An Efficient Approach to Model Strong PUF with Multi-Layer Perceptron using Transfer Learning

Amir Ali Pour, David Hely, Vincent Beroulle, Giorgio Di Natale

An Efficient Approach to Model Strong PUF with Multi-Layer Perceptron using Transfer Learning

Amir Ali-pour*, David Hely†, Vincent Beroulle‡ and Giorgio Di Natale§
(*,†, ‡) Univ. Grenoble Alpes, Grenoble INP, LCIS, 26000 Valence, France
§ Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMA, 38000 Grenoble, France,
(*,†, ‡, §) Email: firstname.lastname@univ-grenoble-alpes.fr

***Abstract*—The study to increase the capability to model strong Physically Unclonable Functions (PUFs) has been a trend recently in the field of Cryptography and Hardware Security. The race between the increasing complexity of strong PUF structures and the increasing capability of modeling strong PUFs with fewer resources for training is still ongoing. In this work, we evaluate a new technique to use Transfer Learning to model strong delay-based PUF using Multi-Layer Perceptron (MLP) as the probabilistic model. Transfer Learning has been already proposed for modeling strong PUF with Convolutional Neural Networks (CNNs). Here we propose Transfer Learning for MLP, since MLP models are relatively less complex and can be trained potentially with less training data. We exploit the reusability of weight values in the hidden dense layers of an MLP model in an existing domain, to further decrease the required resources in training an MLP in another domain. Here a domain represents the the CRP space of a given strong PUF instance. We support our proposed Transfer Learning method with simulated data of some variants of XOR Arbiter PUF. We show that our proposed method can reduce the required number of CRPs by approximately 50% compared to modeling the same MLP with random initialization.**

***Index Terms*—Transfer Learning, Artificial Neural Network, Machine Learning, Multi Layer Perceptron, Physically Unclonable Functions, XOR Arbiter PUF**

I. Introduction

The study of Physically Unclonable Function (PUF) through the past 2 decades has been pervasively expanding. The fundamental idea of PUF is to implement a hardware infrastructure that leverages the intrinsic manufacturing variations (e.g., threshold voltage, critical dimensions) to extract device specific fingerprints. PUFs are potential hardware primitives for light-weight device authentication and encryption key generation [1]–[4].

A PUF circuit generates a response to a given challenge, the so-called Challenge-Response-Pair (CRP). A Strong PUF is a PUF variation that can generate a very large amount of CRPs. For instance, for a PUF circuit with 128-bit challenge size, it is possible to generate 2^{128} CRPs.

The strong PUF variants, especially the delay-based PUF which are based on linear concatenation of delay effects, are likely to be modeled via machine learning methods [5]–[8]. The common practice in this context is to train a probabilistic model to estimate the CRP characteristic of a given PUF circuit. Consequently, the estimated model has the same CRP manifestation with a high similarity to that of the PUF circuit.

The success in modeling strong PUF depends on the number of CRPs given to train the estimated model. It is proven that the minimum number of CRPs required to yield an accurate model is proportional to the level of complexity in the structure of the PUF [9]. Thus modeling a strong PUF can be very data demanding for PUF which are largely complex [9]. For instance, modeling k-XOR Arbiter PUF with $k >= 5$ need above 1 million CRPs to yield estimation accuracy above 90%.

While the increased complexity in the structure of strong PUF seems a workable solution against model-building attacks, new researches such as the ones found in [7], [10] try to tackle PUF complexity by proposing novel probabilistic modeling based on deep learning methods which are structurally tuned to converge faster and require lesser data for training. Moreover, a recent work suggests that a transferring trained data from one PUF to model a new PUF can yield accurate model with reduced number of CRPs required for training [11]. This work proposes using Transfer Learning on CNN models which are relatively more complex models than MLPs as used in [10].

In this work we show that there is also a potential to model strong PUF with MLPs using Transfer Learning to reduce the training CRPs. Our starting point is to use the latest MLP proposed by Mursi et al in [10]. We experimentally prove that it is possible to initialize some of the dense layer weights of a given MLP model with pre-trained values, and consequently train the model accurately with a reduced CRP dataset. We demonstrate that our modeling technique can model strong PUF with less resources compared to its predecessor techniques [10] and [11]. Accordingly, we deliver the following contributions:

- Schematic of a modeling procedure of strong PUF using MLP and Transfer Learning.
- Evaluation with simulated noise-free data of several variants of strong PUF models to show that reusing dense layer weight values can further decrease the required number of CRPs for training compared to [10].
- Experimental assessment with simulated noisy data to show the resilience of modeling scheme based on Transfer Learning to PUF instability.
- A comparison between our proposed Transfer Learning method and wang's Transfer Learning on CNNs in [11]

This material is based upon the work supported by the French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-15-IDEX-02).

in modeling some variants of strong PUF.

The rest of the paper is as follows. In section II, we discuss the preliminary information. In section III, we elaborate on our proposed method. Section IV elaborates on our evaluation setup at the begining and then we discuss the experimental results and comparisons. Section V contains the conclusion of the work and our perspective for the future steps.

II. Preliminaries

A. Modeling PUF with ANN

One of the commonly discussed use-cases of modeling strong PUF is in impersonating attacks which aim at cloning the PUF circuit of a target device [10], [12], [13]. Various mediums are used to collect CRP for an adversary, such as eavesdropping on a communication channel where CRP is being transmitted.

In modeling PUF, the first step is to collect a dataset of CRPs, we denote as C_{PUF}, from the PUF circuit of the target device. To model an strong PUF as we denote as f_{PUF}, the attempt is to train an estimated model h_{PUF} which comprises g_{PUF} and a set of internal values θ. Model h_{PUF} goes through a training process which comprises several steps. The primary step is the initialization where θ is initialized with some primary values. After that the optimization process is started. Let us consider c_i as a challenge value from the dataset C_{PUF}. The optimization process iteratively changes the values of θ until the following verification is true:

$$f_{PUF}(c_i) = r_i \approx r'_i = g_{PUF}(c_i, \theta) = h_{PUF}(c_i) \quad (1)$$

Where r_i is the PUF circuit's response to the challenge c_i and r'_i is the estimated model's prediction of r_i for c_i. The learning algorithm of the optimization process updates the internal values θ with respect to C_{PUF} and the function g_{PUF}. To know if h_{PUF} is accurate, it is tested with a subset of C_{PUF} which is not used during the training. If the model could predict correctly the majority of the responses of the CRPs, the model is then considered optimally trained. Noting that the training process is empirical and needs to be performed in variable iterations until the desired prediction accuracy is achieved. We assume here that the estimated model is an ANN. A typical ANN comprises of several layers $\{L_i | i = (1, 2, ..., n)\}$ of cumulative activations. Each layer L_i includes weighted inputs from the preceding layer, and several nodes which comprise a bias value, and an activation function of the weighted inputs. The activation function itself is a linear regression of the sum of its inputs. Here the internal parameter θ for an ANN is the weight and the bias values from all the layers.

B. Transfer Learning

Depending on how the internal values of an ANN are initialized, the training process can lead faster or slower to a point of convergence. Several techniques for ANN weight and bias initialization have been proposed . A selection of these techniques are discussed in [14]. In contrast to these solutions, another solution called Transfer Learning exists which is based on reusing the values of internal parameters of the trained estimated model [15]. It is proven that Transfer Learning is a potential technique to mitigate the large data dependency issue in deep learning. This addresses the demand on very large number of samples in training in order to yield an optimal prediction accuracy, which also appears in modeling variants of strong PUFs.

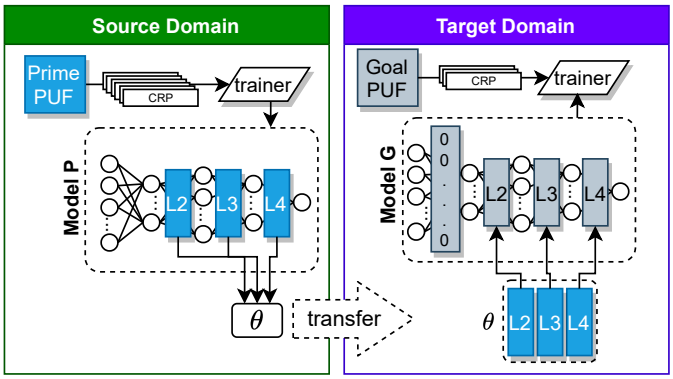


Fig. 1: Our proposed Transfer Learning Plan, based on reusing the hidden layers of Mursi's [10] MLP model.

In the context of strong PUF modeling, the main goal is to reduce the required CRP in building an accurate estimation model for a PUF circuit *PUF GOAL*. Here Transfer Learning suggests extracting θ from an already trained model, we refer to as *MODEL P* which is an estimation of a PUF circuit we refer to as *Prime PUF*. A Transfer Learning Plan then decides which part of θ can be reused. Based on the Transfer Learning Plan, θ' is generated from θ. The modified θ' is then assigned accordingly to the internal parameters of a new model *MODEL G*. *MODEL G* is then passed to the training process for estimating *PUF GOAL*. In the following we elaborate on our proposed Transfer Learning Plan which can be applied to MLPs in modeling strong PUF.

III. Proposed Method

MLPs have been a common selection as a fit probabilistic model structure to estimate large and complex strong PUFs [7], [10], [16]. Although other neural network models exist such CNN that have the potential to model strong PUFs with large complexity. However, CNN's computation overhead exceeds that of an MLP with the same capability. Therefore, it is interesting to see if the potential of Transfer Learning can emerge with MLPs in modeling strong PUFs. This has not been practiced before. At the time, the latest practice of Transfer Learning for modeling strong PUF we found is in [11] which uses CNN models. We continue in the following to elaborate on how we can successfully use Transfer Learning on MLP models in modeling strong PUF to reduce the number of CRPs required for training. In this work we take the following assumptions:

- A target PUF exists which is the goal to model. We call this the Goal PUF.
- Access to the Goal PUF is limited. Thus only a few CRPs can be collected.

- The structure of the Goal PUF is known to the adversary.
- A PUF circuit is available with full control. We call this the Prime PUF.
- Access to the Prime PUF is unlimited. Therefore as many CRPs as required can be collected.
- The structure of the target PUF is known and is similar to the structure of the Prime PUF.

In common Transfer Learning practices, it is suggested to transfer the weight values from the initial layers of an ANN model. It is assumed that the primary features of the target domain are learned in the primary layers of an ANN. In this work, we observed in a preliminary experiment that if we transfer the first layer's weight and bias values, the training performance is negatively affected. Meaning that it will take more CRPs and time of training until convergence. On the other hand, we observed that transferring the weight and bias values of the hidden layers have an opposite effect. Meaning that the training improved in terms of number of CRPs required for training, as well as the success rate of the training and the time of training until convergence.

In using the lateral approach, we measured the average distance between each corresponding layer's weight and bias values of Model P and Model G. We observed that the distance between corresponding weight values on Model P and Model G on the first layers, are considerably higher than that in the subsequent hidden layers. Accordingly, we speculated that the primitive features learned in the first dense layer of Model P are device-specific. Therefore transferring these values to Model G can lead to the possibility of requiring more CRPs and time of training. On the other hand, the weight values in the hidden layers and the output layer of a trained model in general, could correspond to learned features in higher levels of abstraction regarding the characteristic of the target PUF. Therefore, we speculate that the weight value in the hidden layers have potential to be reused in modeling strong PUF with the same structural complexity.

Accordingly, we make a Transfer Learning plan to reuse the values of all hidden layers weights and biases, excluding the weights between the input layer and the first hidden layer. A schematic of our Transfer Learning plan is shown in Fig. 1. We take the following steps to perform our modeling method:

- **Step 1)** Initialize an MLP (Model P) to model the prime PUF circuit.
- **Step 2)** Train Model P with the CRPs captured from the prime PUF instance.
- **Step 3)** Initialize another MLP model (Model G) with the similar structure to the prime model. All weights of Model G are initialized with zeros.
- **Step 4)** Overwrite the weight and bias values of layer 2 to layer l of Model G with the weight and bias values of layer 2 to layer l of Model P. Here l is number of layers in the model.
- **Step 5)** Train Model G with a CRP set captured from the PUF goal circuit.

In the following section we analyze our proposed Transfer Learning plan.

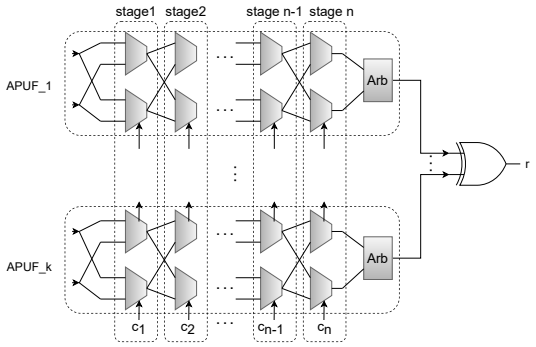


Fig. 2: The structure of an n-stage k-XOR Arbiter PUF.

IV. Evaluation setup and experimental results

Our evaluation in this work is based on modeling variants of XOR Arbiter PUFs using simulated data. Here we first elaborate on the structure of XOR Arbiter PUF, and then the simulation code which we used to generate our CRP dataset for modeling. We will then elaborate on our model training setup and discuss our experimental results.

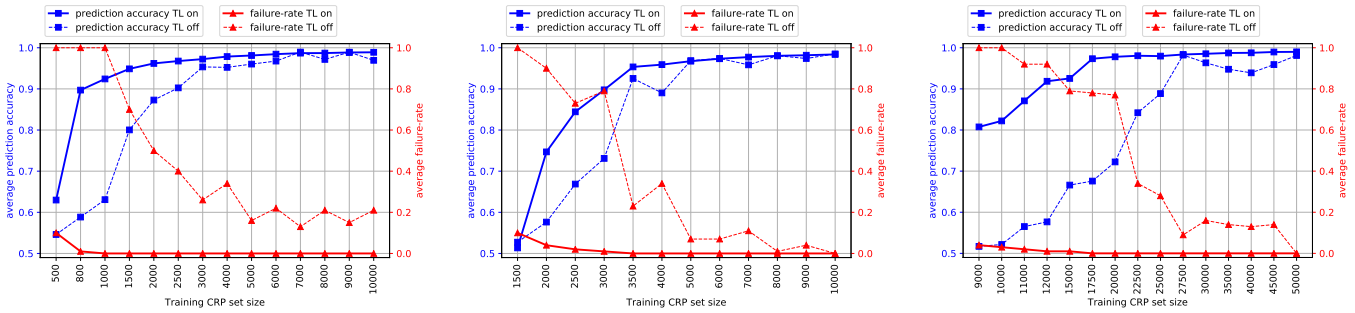
A. Experimental Setup

XOR Arbiter PUF is a variant of the Arbiter PUF family introduced first in [17]. Arbiter PUF is based on the delay difference between two racing signal paths which are structurally similar. Due to minor process variations, the signals have a timing difference which allows one path to reach the terminal point (an Arbiter) faster than the other. This timing difference also inherently varies between different PUF circuits. Which then gives each PUF circuit a unique characteristic. An XOR Arbiter PUF is created with 2 or more Arbiter PUFs which are set in parallel, and their outputs connected to an XOR block. The challenge input to all Arbiters is the same, and the output of the XOR is the output of the XOR Arbiter PUF. Fig. 2 shows the structure of an n-stage k-XOR XOR Arbiter PUF.

We reused the Python-based XOR Arbiter PUF simulator developed by Ruhrmair and described in [13]. The code of this simulator is available in [18]. In this simulator the timing

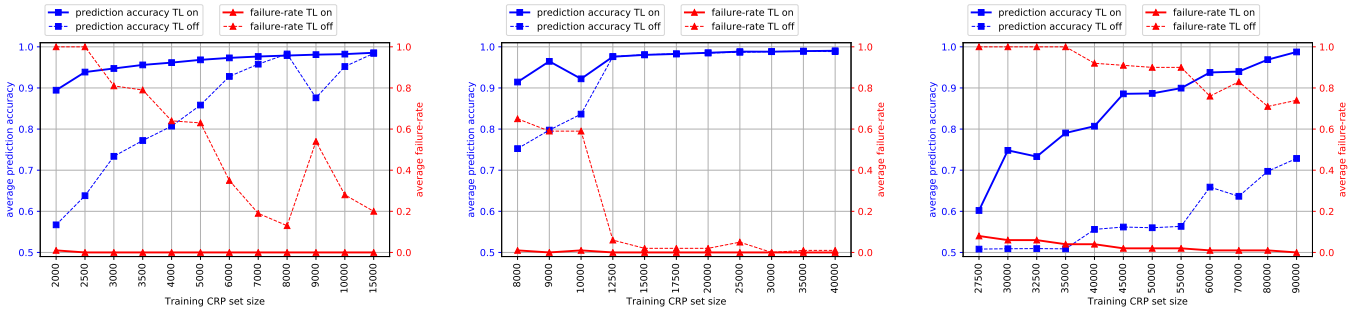
TABLE I: Training specifications & Hyper-parameters

Parameter	**Value**
Optimization function	Adam
Loss function	BCEloss
Learning rate	0.001, 0.0001
Weight initializer	Kaiming Uniform
Bias initializer	Unifrom
Maximum epoch	2000
Maximum re-training attempts for Transfer Learning disabled	10
Maximum re-training attempts for Transfer Learning enabled	2
Training batch size	= Training set size
Optimal Test Accuracy	90%



(a) 2-XOR (b) 3-XOR (c) 4-XOR

Fig. 3: Modeling variants of 64-bit XOR Arbiter PUF with and without Transfer Learning.



(a) 2-XOR (b) 3-XOR (c) 4-XOR

Fig. 4: Modeling variants of 128-bit XOR Arbiter PUF with and without Transfer Learning.

difference in the two racing paths are modeled as the sum of the delays in each stage. In simulating XOR Arbiter PUF, the simulation employs k individual n-stage Arbiter PUF in parallel. The same challenge input is applied to all parallel Arbiter PUFs, and their individual outputs are XORed to yield the final response. The delay parameter values in the simulator are generated randomly with a standard normal distribution, with mean 0 and standard deviation 1.

Using the python PUF simulator, we generated 10 instances of $64, 128$-stage $2, 3, 4$-XOR Arbiter PUF variants. Noting that this simulation does not inherently include the PUF instability noise. To simulate noisy PUF, we randomly selected CRPs from each dataset and flipped the response. Accordingly we generated noisy CRP datasets for each PUF variant, with 2%, 5% and 10% noisy CRPs in each dataset.

On the modeling part, we recreated Mursi's ANN in [10]. This model is an MLP with 3 hidden layers. The number of neurons in each layer are $2^{(k-1)}$, $2^{(k)}$, $2^{(k-1)}$, for the first, second and third hidden layer, respectively. Here k associates with the number of XORs in a n-stage k-XOR Arbiter PUF. The activation function used for all layers except the output layer is hyperbolic tangent function ($Tanh$). The output layer uses the Sigmoid activation function.

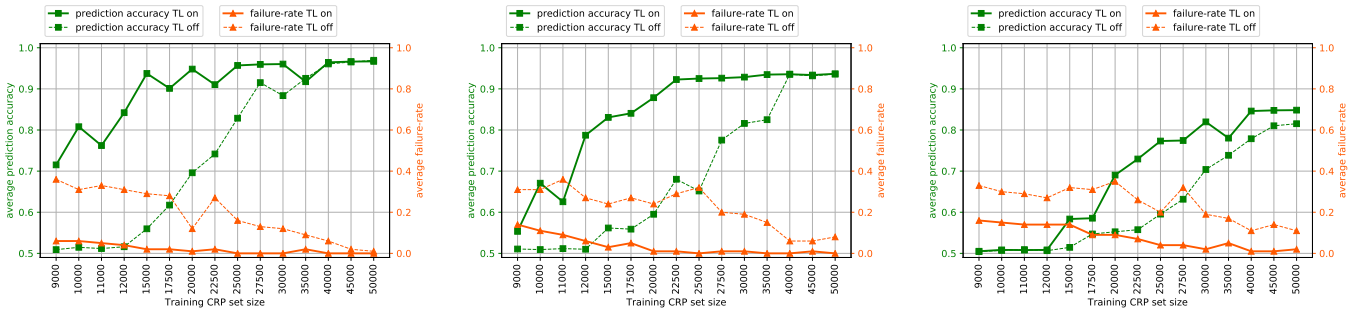
In terms of the training specifications and hyper parameters, our entire parametric consideration for the training are given in Table I. We implemented the model creator and the trainer code with Pytorch on Python 3.8 with Anaconda IDE and the Spyder editor on Windows 10. The system we used for training has 2 Nvidia Quadro RTX 5000 graphics cards. Each card has 8 GBs of dedicated memory, 3072 cuda cores with 448 GB/s memory bandwidth. The system has also 2 intel Xeon silver CPUs with 2.2 GHz speed, 128 GBs of RAM, and 10 TB of storage. It is worth noting that during our experiments, only one of the Quadro RTX 5000 was enough. Also since the dedicated GPU memory was large, we considered the training batch size to be the size of the training set, and also increased the number of epochs accordingly. We clarify that the maximum epoch indicated in Table. I should not lead to and overfitting problem, since the fitting iteration (updating the trainable parameters θ) is performed only once per epoch (due to the batch size being the same as training set size). This is similar to having batch size $= \frac{training\ set\ size}{10}$, and maximum epoch $= 200$. However, the larger batch-size allowed us to get a better training performance on GPU.

B. Experimental results

In modeling each variant of XOR PUFs for various training set sizes, we measure the following:

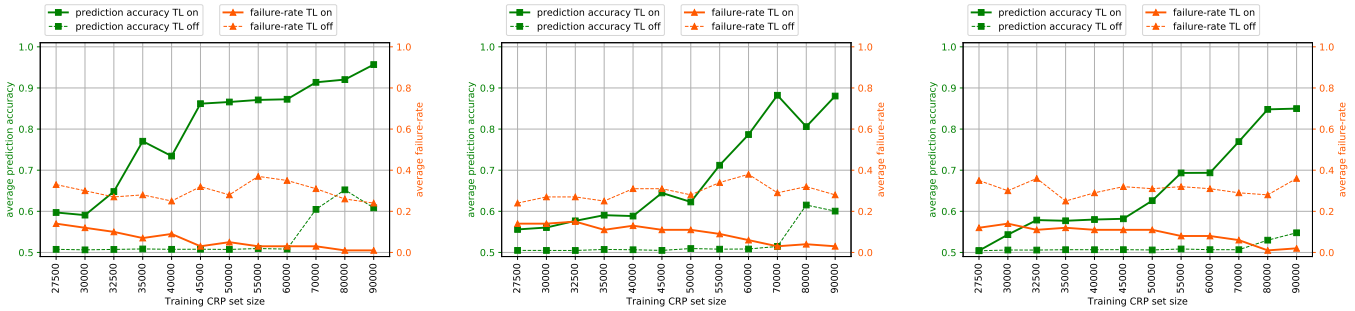
- Averaged prediction accuracy $= \frac{\Sigma_{i=1}^{10} \epsilon_i}{10}$
- Averaged failure-rate $= \frac{\Sigma_{i=1}^{10} NTr_i}{MAX\_NTr}$

Where ϵ_i and NTr_a, indicates the trained model's prediction accuracy and number of retraining attempts, respectively., of



(a) 2% noise (b) 5% noise (c) 10% noise

Fig. 5: Modeling 64-bit 4-XOR Arbiter PUF with and without Transfer Learning with presence of noisy CRP.



(a) 2% noise (b) 5% noise (c) 10% noise

Fig. 6: Modeling 128-bit 4-XOR Arbiter PUF with and without Transfer Learning with presence of noisy CRP.

modeling the ith PUF instance of a given PUF variant and $i \in \{1, 2, \ ... \ , 10\}$. MAX_NTr also indicates the maximum re-training attempts as given in Table. I.

We first compare our method to Mursi's MLP with random initializer. Fig. 3 shows the training results for the variants of 64-bit XOR PUFs. The first observation is on the average prediction accuracy. If we consider 90% as the target accuracy, we see that training with our Transfer Learning Plan leads to the target accuracy with fewer CRPs in all cases. The average failure-rate also is almost zero using our method. We measured the same for variants of 128-bit XOR Arbiter PUF as shown in Fig. 4. The same results as that in modeling 64-XOR PUF variants can be seen here. Where the target accuracy above 90% can be achieved with fewer CRPs using our method for initialization. The average failure-rate also is always less than modeling with with random initializer.

Our Transfer Learning plan also shows to have a lead against modeling with random initializer where the training data includes some noisy CRPs. Fig. 5 shows the results of training the 64-bit 4-XOR Arbiter PUF variant in different levels of noise. It is apparent that the difference between the deterioration rate in modeling with random initializer and our proposed method is significant. For instance in the case of 10% noise and for the maximum number of CRPs given for training, the baseline's prediction error has increased by almost 20%, whereas the case with Transfer Learning, it has dropped for 15%. Although it is apparent also that the failure-rate has increased for both cases. Yet the difference of the average failure-rate between the case of modeling with random initailizer and our proposed method is significant. Same conclusions can be drawn in modeling a variant with larger challenge space. Fig. 6 shows the results of training the 128-bit 4-XOR Arbiter PUF with different levels of noise. The results prove that the modeling with our Transfer Learning plan yields the similar advantage.

The observation on modeling with noisy data suggests that using Transfer Learning for initialization has the potential of more resilience to noise compared to a baseline where initialization is performed randomly given any training CRP set size.

We also compare our method to the Transfer Learning method proposed by wang et al. in [11] . In their work, Transfer Learning is done on CNN models to reduce the required number of CRPs for training XOR Arbiter PUF variants. Their method relies on transferring the convolutional layer's weight values from the source domain to the target domain. This means that their method is applicable on CNNs only. Since their CNN specification was not given, we could not reproduce their modeling technique, as we did for Mursi's MLP in [10]. However, their work is the only attempt of modeling PUF with Transfer Learning, and closest and to our work. To compare, we consider the modeling targets (the XOR

TABLE II: Comparing Wang's Transfer Learning with CNN [11] (A) and our proposed transfer learing plan with MLP (B).

Challenge size	No. stage	Prediction accuracy	No. CRPs $\times 10^3$	
			A	B
64	2	90%	3.5	.8
	3	90%	31	3
	4	70%	32	-
		80%	39	9
		90%	NA	12
128	2	90%	5.5	2
	3	90%	115	8
	4	70%	170	30
		80%	190	35
		90%	380	45

Arbiter PUF variants) as the baseline for comparison. Table II shows the comparison between our method and Wang's in [11]. Accordingly, to obtain estimated model with 90% prediction accuracy, our method shows to require significantly less number of CRPs for training overall, compared to that of Wang's. We assume that a good proportion of CRP reduction is due to the structure of Mursi's MLP [10] as we reproduced. It is already proven in the experiments in [10] that this MLP requires much less CRPs to converge to an accurate model compared to previous MLP models. This therefore gives advantage to our method where we apply Transfer Learning to Mursi's MLP in order to further decrease the required training data.

V. Conclusion and future perspective

In this work, we proposed Transfer Learning technique to model strong PUF with Multi-layer Perceptron (MLP). We proved that a trained MLP model of PUF circuit can be a source domain, and the weight and bias values of the hidden layers the model can be reused to model a PUF circuit with similar structure with less training data. We experimentally proved that we can considerably decrease the required number of CRPs by approximately 50% compared to an MLP model which is initialized with random values. We also showed that our technique has a small failure-rate close to zero for noise-free modeling and maximum 15% for training with noisy CRPs where 10% of the CRPs are incorrect. We showed that using Transfer Learning on MLPs has resilience to various levels of noise up to 10% of noisy CRPs in the training data. In a future extension of this work, our goal is to investigate independent modeling schemes in which the exact structure of the PUF is not known before hand. These models should be large as the worst case is to anticipate that the target strong PUF's complexity is very high. The future work then will discover the potential of using Transfer Learning in order to reduce the training data for modeling strong PUF with large design complexity.

References

[1] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6823677/

[2] J. Delvaux, R. Peeters, D. Gu, and I. Verbauwhede, "A survey on lightweight entity authentication with strong pufs," *ACM Comput. Surv.*, vol. 48, no. 2, Oct. 2015. [Online]. Available: https://doi.org/10.1145/2818186

[3] A. Alipour, D. Hely, V. Beroulle, and G. Di Natale, "Power of prediction: Advantages of deep learning modeling as replacement for traditional PUF CRP enrollment," in *TrueDevice2020*, 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02954099

[4] A. Alipour, V. Beroulle, B. Cambou, J. Danger, G. D. Natale, D. Hely, S. Guilley, and N. Karimi, "Puf enrollment and life cycle management: Solutions and perspectives for the test community," in *2020 IEEE European Test Symposium (ETS)*, 2020, pp. 1–10, ISSN: 1558-1780.

[5] J.-Q. Huang, M. Zhu, B. Liu, and W. Ge, "Deep learning modeling attack analysis for multiple fpga-based apuf protection structures," in *2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*. IEEE, 2018, pp. 1–3. [Online]. Available: https://ieeexplore.ieee.org/document/8565728/

[6] F. Ganji and S. Tajik, "Physically unclonable functions and ai: Two decades of marriage," 2021.

[7] A. O. Aseeri, Y. Zhuang, and M. S. Alkatheiri, "A machine learning-based security vulnerability study on xor pufs for resource-constraint internet of things," in *2018 IEEE International Congress on Internet of Things (ICIOT)*, 2018, pp. 49–56.

[8] F. Regazzoni, S. Bhasin, A. A. Pour, I. Alshaer, F. Aydin, A. Aysu, V. Beroulle, G. Di Natale, P. Franzon, D. Hely, N. Homma, A. Ito, D. Jap, P. Kashyap, I. Polian, S. Potluri, R. Ueno, E.-I. Vatajelu, and V. Yli-Mäyry, "Machine learning and hardware security: Challenges and opportunities -invited talk-," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–6.

[9] J. Tobisch and G. T. Becker, "On the scaling of machine learning attacks on pufs with application to noise bifurcation," in *Radio Frequency Identification*, S. Mangard and P. Schaumont, Eds. Cham: Springer International Publishing, 2015, pp. 17–31.

[10] K. T. Mursi, B. Thapaliya, Y. Zhuang, A. O. Aseeri, and M. S. Alkatheiri, "A fast deep learning method for security vulnerability study of xor pufs," *Electronics*, vol. 9, no. 10, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/10/1715

[11] Q. Wang, O. Aramoon, P. Qiu, and G. Qu, "Efficient transfer learning on modeling physical unclonable functions," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2020, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9137057/

[12] M. Khalafalla and C. Gebotys, "PUFs deep attacks: Enhanced modeling attacks using deep learning techniques to break the security of double arbiter PUFs," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 204–209. [Online]. Available: https://ieeexplore.ieee.org/document/8714862/

[13] U. Ruhrmair, F. Sehnke, J. S olter, G. Dror, S. Devadas, and J. u. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security - CCS '10*. ACM Press, 2010, p. 237. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1866307.1866335

[14] Z. Lyu, A. ElSaid, J. Karns, M. Mkaouer, and T. Desell, "An experimental study of weight initialization and lamarckian inheritance on neuroevolution," *EvoApplications Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany*, vol. 12694, 2021.

[15] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Artificial Neural Networks and Machine Learning – ICANN 2018*, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds. Cham: Springer International Publishing, 2018, pp. 270–279.

[16] K. T. Mursi, Y. Zhuang, M. S. Alkatheiri, and A. O. Aseeri, "Extensive examination of XOR arbiter PUFs as security primitives for resource-constrained IoT devices," in *2019 17th International Conference on Privacy, Security and Trust (PST)*. IEEE, 2019, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/document/8949070/

[17] J. Lee, D. Lim, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525)*, 2004, pp. 176–179.

[18] http://www.pcp.in.tum.de/code/lr.zip.