Power-Based Attacks on Spatial DNN Accelerators

GE LI, MOHIT TIWARI, and MICHAEL ORSHANSKY, The University of Texas at Austin, USA

With proliferation of DNN-based applications, the confidentiality of DNN model is an important commercial goal. Spatial accelerators, that parallelize matrix/vector operations, are utilized for enhancing energy efficiency of DNN computation. Recently, model extraction attacks on simple accelerators, either with a single processing element or running a binarized network, were demonstrated using the methodology derived from differential power analysis (DPA) attack on cryptographic devices. This paper investigates the vulnerability of realistic spatial accelerators using general, 8-bit, number representation.

We investigate two systolic array architectures with weight-stationary dataflow: (1) a 3×1 array for a dot-product operation, and (2) a 3×3 array for matrix-vector multiplication. Both are implemented on the SAKURA-G FPGA board. We show that both architectures are ultimately vulnerable. A conventional DPA succeeds fully on the 1D array, requiring 20K power measurements. However, the 2D array exhibits higher security even with 460K traces. We show that this is because the 2D array intrinsically entails multiple MACs simultaneously dependent on the same input. However, we find that a novel template-based DPA with multiple profiling phases is able to fully break the 2D array with only 40K traces. Corresponding countermeasures need to be investigated for spatial DNN accelerators.

CCS Concepts: • **Security and privacy → Side-channel analysis and countermeasures**.

Additional Key Words and Phrases: power side-channel attack, DNN model extraction, spatial DNN accelerators

ACM Reference Format:
Ge Li, Mohit Tiwari, and Michael Orshansky. 2021. Power-Based Attacks on Spatial DNN Accelerators. *ACM J. Emerg. Technol. Comput. Syst.* Special issue on Trustworthy AI, 1 (August 2021), 18 pages.

1 INTRODUCTION

Progress in Deep Neural Networks (DNNs) has been driving various applications including computer vision [10], [11], speech recognition [1], [7], medical image analysis [22], [26], malware detection [24][30] and many others. DNNs enable learning complex features in input data [28], achieving superior performance in a variety of tasks, compared to conventional machine learning algorithms.

Excellent performance of DNNs depends on tremendous effort to train the neural network model. The cost of model creation can be captured in the following aspects: (1) Labor and time to create/collect a dataset for training, (2) the cost of computing resources to run the DNN training algorithm, and (3) the search for hyper-parameters which result in optimal DNN models. Therefore, DNN models for specific applications need to be considered a form of intellectual property with high commercial value. This creates a motivation to obtain the DNN models via adversarial/non-commercial means, e.g. via attacks that extract the model. Besides the loss of commercial value, the loss of DNN model confidentiality can also lead to security and privacy problems. An exposed model may facilitate adversarial attacks, where an attacker crafts input samples that make the

Authors' address: Ge Li, lige@utexas.edu; Mohit Tiwari, tiwari@utexas.edu; Michael Orshansky, orshansky@utexas.edu, The University of Texas at Austin, Department of Electrical and Computer Engineering, 2501 Speedway, Austin, Texas, USA, 78712.

1550-4832/2021/8-ART $15.00
https://doi.org/

target model mis-classify [20], or, membership inference attacks, in which an attacker aims to learn whether an input to the target model belongs to its private training set [27].

Model extraction attacks utilizing class probabilities (confidence scores) from model output have been demonstrated. In [32], Tramèr et al., demonstrated a model extraction attack against some DNNs by observing outputs of API queries. In [14], Jagielski et al., showed an efficient learning-based model extraction attack that utilizes query outputs. Query-based attacks can be effectively defended by limiting model output. Side-channel attacks have also been demonstrated as effective for DNN model extraction [3, 8, 9, 12, 13, 34–36]. Side-channel attacks offer a significant risk because side-channel information emanating during DNN execution is difficult to eliminate. Exploitation of digital side channels enables the attacker to extract coarse-grained information of the target DNN model, such as its architecture. In [13], Hua et al., proposed a reverse-engineering attack based on observing the accelerator off-chip memory access patterns and enumerating the possible architectures that satisfy the constraints. In [35], Yan et al., utilized both memory access and timing side channels to reverse-engineer the size of matrices involved in matrix multiplication allowing to infer the DNN architecture. Hu et al., extracted the DNN architecture from the noisy PCIe and memory-bus events on a GPU platform [12]. Duddu et al. utilized execution time to infer target DNN layer depth [9].

As this paper demonstrates, attacks that exploit device power consumption or EM emanation enable a direct retrieval of DNN weights. In an attack modality, borrowed from attacks on cryptographic implementations, the attacker feeds a large number of inputs and collects corresponding power/EM traces using a hypothesis-testing foundation. The attacker selects a power model which represents the power of an intermediate state of the secret-related computation, makes a hypothesis on the secret, and evaluates the power model based on the hypothesis. The correct hypothesis results in the highest correlation between power/EM traces and predicted power values.

Several efforts have demonstrated such attacks for DNN weight retrieval. In [3], Batina et al., demonstrated a correlation EM attack on a micro-controller to retrieve approximate values of single-precision weights of a multi-layer perceptron. While the attack is demonstrated on a conventional micro-controller, its feasibility on a customized DNN hardware accelerator with high parallelism is unexplored. In [36], Yoshida et al., implemented an FPGA-based DNN accelerator with a single processing element (PE) and performed a correlation EM attack to retrieve the weights, stored in a 8-bit fixed-point representation, by analyzing the multiply-and-accumulate operation. However, this work has not studied the feasibility of an attack on a high-performance accelerator with multiple processing elements, which is a more common deployment scenario. Dubey et al. demonstrated a differential power attack (DPA) to retrieve the binarized weights of a model implemented in an FPGA-based DNN accelerator with an adder tree used for accumulation [8]. The attack is only evaluated against binarized weights. Whether the attack can be effective against a 8-bit weight implementation, which is typically used by state-of-the-art accelerators, needs to be examined.

In this work, we demonstrate a differential power attack to retrieve the weights from the FPGA-based matrix multiplication accelerator. We adopt a 8-bit integer (INT8) weight representation and implement a design with multiple processing elements performing parallel multiply-and-accumulate (MAC) operations. The design adopts a weight-stationary dataflow. Matrix multiplication between input activations and weights is the core computational kernel of DNNs, as computation of both convolutional and fully-connected layers can be mapped to matrix multiplication. We propose a multi-step DPA framework which utilizes the dependency between MAC results of different weights to determine the value of each weight using measured power traces.

We study both 1D and 2D systolic arrays. We consider a 1D array as a separate case because it is an important VLSI model. In addition, compared to 2D arrays, it can offer better reconfigurability, lower memory bandwidth requirement, and better energy efficiency due to the reduced inter-PE

communication and control logic complexity, which may be desired in certain scenarios [16, 33]. We first study a 3×1 systolic array. The results show that we are able to retrieve all weights in the weight vector of the 1D array using 20K power traces with a conventional DPA. Next, we study the scalability of the DPA attack to larger designs. We implement a 3×3 systolic array. The results show that the 2D accelerator exhibits higher security. Both a conventional DPA and a stronger, template-based DPA fail: only 30% of weights are recovered after 460K traces. We investigate the causes of higher resistance of the 2D array to the attack compared to the 1D accelerator. We explain the reason for higher security of a 2D array design by the fact that it intrinsically entails multiple MAC outputs that are simultaneously dependent on the same input. We show that this is fundamentally a feature of the weight-stationary dataflow. However, we discover that an enhanced template-based DPA with multiple profiling phases manages to expose leakage of each PE column step-by-step and retrieves weights from each column. The attack is able to retrieve all weights from the 2D array with only 40K traces. The results on both 1D and 2D arrays show that both architectures are ultimately vulnerable.

Our contribution can be summarized as follows:

- We investigate the vulnerability of a 1D systolic array. Our results show that a conventional DPA on the 1D array succeeds fully, requiring 20K power measurements.
- We investigate the vulnerability of a 2D systolic array. However, only 30% of the model weights are retrieved even after 460K power traces with a conventional DPA and a stronger, template-based DPA. Our analysis finds that the higher security of a 2D array arises from the fact that it intrinsically entails multiple MACs that are simultaneously dependent on the same input.
- We propose a novel template-based DPA with multiple profiling phases which fully breaks the 2D systolic array.
- We investigate Hamming distance, Hamming weight, and bit-level power models in the attack on a 2D array.

2 DNN ACCELERATOR DESIGN

2.1 DNN and Matrix Multiplication

The computation of major DNN layers, including fully-connected layers and convolution layers, can be mapped to matrix multiplication. The fully-connected layer computes a weighted sum of all its input activations for each output activation. For a fully-connected layer with M input neurons and N output neurons, this process can be summarized as the multiplication between an $N \times M$ weight matrix and an $M \times 1$ input activation vector, where each row of the weight matrix represents the weights used to calculate the corresponding output activation. The convolution layer computes a dot product between the filter weights and input feature map pixels within the filter for each output feature map pixel. The output feature map is computed by sliding the filter with a certain stride. This computation process can be mapped to matrix multiplication between the filter weights and the input activations as well. For a convolution layer with C_{in} input channels, C_{out} output channels, $W_{in} \times H_{in}$ input feature map, $W_f \times H_f$ filter, stride S, and padding size P, the filter weights can be represented by a matrix with C_{out} rows and $C_{in} \times W_f \times H_f$ columns. The input feature map pixels can be organized as a matrix with $C_{in} \times W_f \times H_f$ rows and $W_{out} \times H_{out}$ columns, where W_{out} and H_{out} represent width and height of the output feature map and can be computed as:

$$W_{out} = (W_{in} - W_f + 2P)/S + 1 \tag{1}$$

$$H_{out} = (H_{in} - H_f + 2P)/S + 1 \tag{2}$$

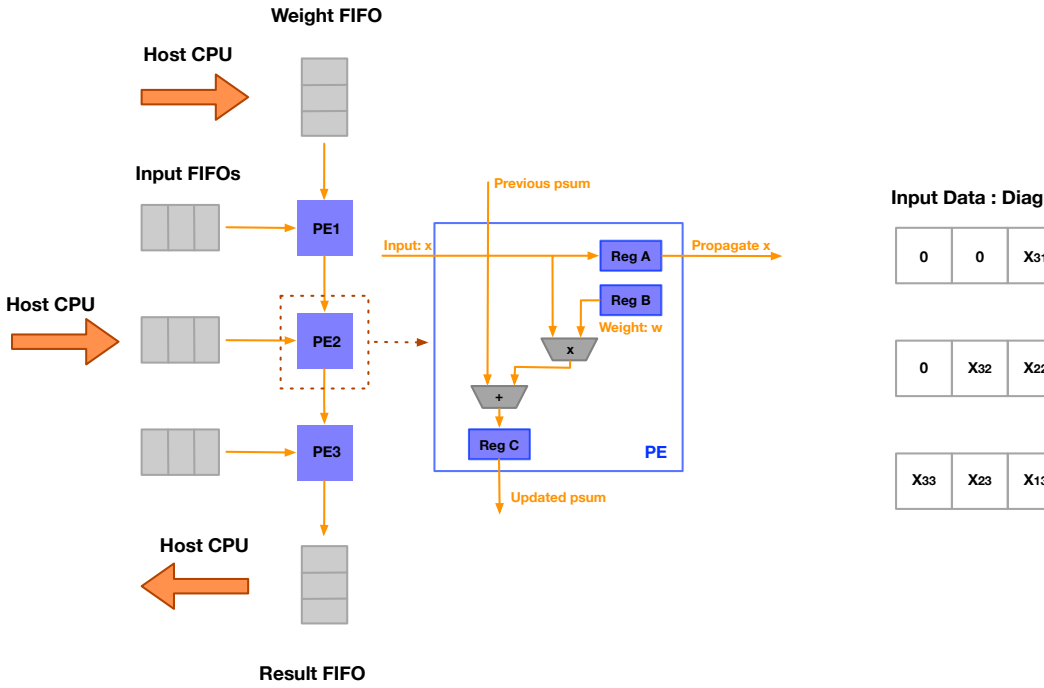


Fig. 1. A 1D dot product array that uses weight-stationary dataflow.

We now provide the details of the implementation of the systolic array matrix multiplication accelerator for neural networks. We implemented two versions of the accelerator: a 1D systolic array with 3 PEs, and a 2D systolic array with 9 PEs. Each PE of the accelerator can perform a 8-bit MAC on signed integers. The systolic array uses a weight-stationary dataflow, similar to Google's TPU design [15]. Since the 1D systolic array computes the dot product while the 2D systolic array computes the matrix-vector multiplication over the input vector, we refer to the 1D systolic array design as the dot product accelerator and to the 2D systolic array design as the matrix-vector multiplication (MVM) accelerator.

2.2 Dot Product Accelerator

The dot product array consists of input, weight, and result FIFOs and 3 PEs arranged in a single column, as shown in Fig. 1. Each PE contains a multiplier and an adder, as well as registers to hold the input, the weight and the resulting partial sum. First, the accelerator receives inputs and weights from a host CPU and pushes them into the FIFOs. The design has the input FIFO depth of 3, which means that the dot products for 3 3×1 input vectors are computed for each host-to-accelerator data transfer. Next, the weights are popped into a register of the corresponding PE. After pre-loading all weights into PEs, the main phase of the dot product computation begins.

During the computation phase, the inputs are propagated from input FIFOs to PEs. The propagation of inputs is arranged in a diagonal wave-front format: the start of data propagation for adjacent rows of input FIFOs differs by one clock cycle, as shown in Fig. 2. Each clock cycle the inputs multiply the weight in each PE. The result is accumulated with the partial sum from a previous PE. The updated partial sum is propagated down to the next PE. The first PE performs only multiplication as there is no previous partial sum. It takes 5 clock cycles to compute the dot

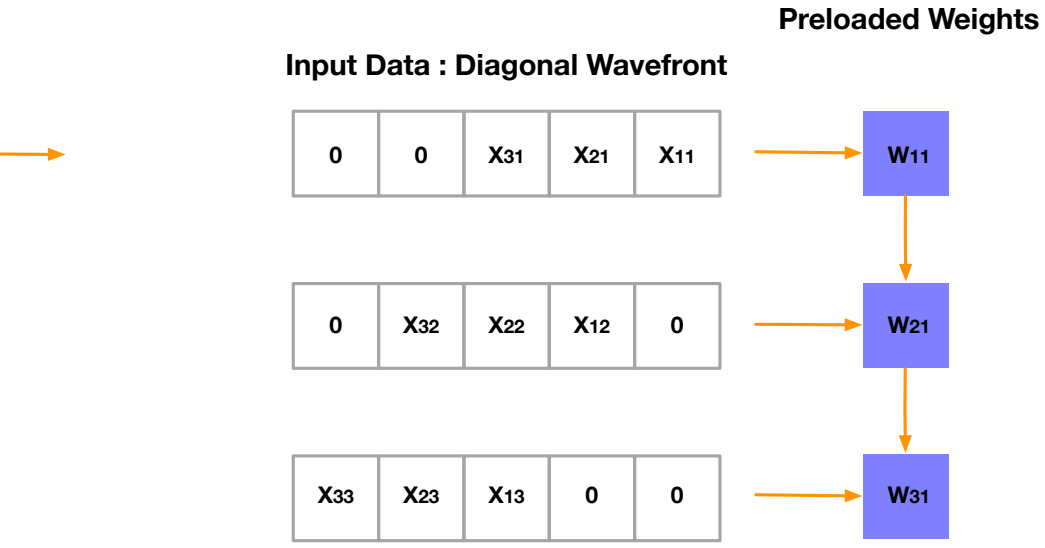


Fig. 2. The diagonal wave-front propagation of inputs in the dot product accelerator.

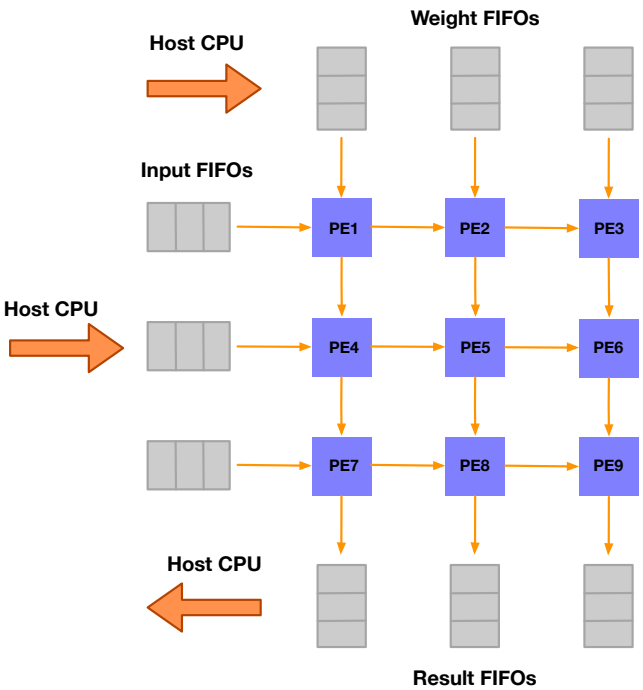


Fig. 3. A 2D spatial matrix-vector multiplication array.

products for 3 input vectors. The results are pushed into the result FIFO, and read out by the host machine.

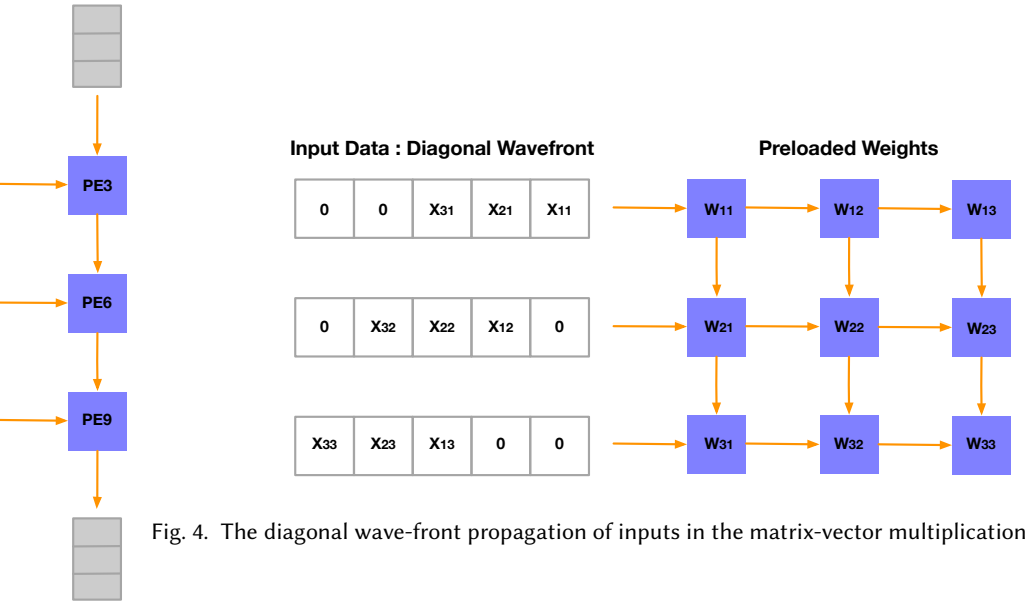


Fig. 4. The diagonal wave-front propagation of inputs in the matrix-vector multiplication accelerator.

2.3 Matrix-Vector Multiplication Accelerator

The matrix-vector multiplication accelerator is a 2D array with multiple PE columns, as shown in Fig. 3. The weight and result FIFOs, of the same size as before, are implemented for each PE column. The PE design is unchanged, and propagates inputs from left to right, across PE columns. As before, the accelerator receives inputs and weights from the host CPU and loads the weight matrix into the 3×3 systolic array. During computation, the inputs are arranged in a diagonal wave-front format, Fig. 4. For each host-to-accelerator data transfer, the matrix-vector multiplication of 3 3×1 input vectors is computed, taking 7 clock cycles. The host CPU reads out the content of the result FIFOs.

3 DPA-STYLE ATTACK: SETUP

In this section, we demonstrate a DPA on the dot product and MVM accelerators.

3.1 Threat Model

We consider two types of attackers. The first type can only observe the inputs to the target device and measure power signatures of the device [25]. The second type has a full access to an identical device with the ability to modify the secret values and collect power profiles to facilitate attacks on the target device [6], [17]. We also assume that implementation details of the target device, with the exception of secret weights, are known to the attacker [8]. This is reasonable since the neural network model is often trained after a large effort while the implementation of hardware is usually widely known (public).

3.2 Experimental Setup

We synthesized and implemented the dot product and matrix-vector multiplication accelerators, individually, on the SAKURA-G FPGA board. The multiplier and adder of each PE are implemented using a DSP slice. On the FPGA, we floor-planned multiple square physical regions next to one another, assigned one region to each PE, and constrained the implementation of each PE to utilize the resources within its corresponding region. We assigned a trigger signal, which indicates the start of computation, to one of the user GPIO pins of the FPGA board. We used the PicoScope 2408B to capture power traces. FPGA clock frequency is set to 1.5MHz. The power traces are collected at 500MS/s. We choose a low FPGA clock frequency due to sampling rate limit (1GS/s maximum) of the oscilloscope.

Fig. 5 shows one power trace for the dot product and matrix-vector multiplication accelerators. We use a 5-th order low-pass Chebyshev Type I filter with the 0.002 dB passband ripple and a pass-edge frequency of 25MHz to process the raw power traces. The red dash line highlights the

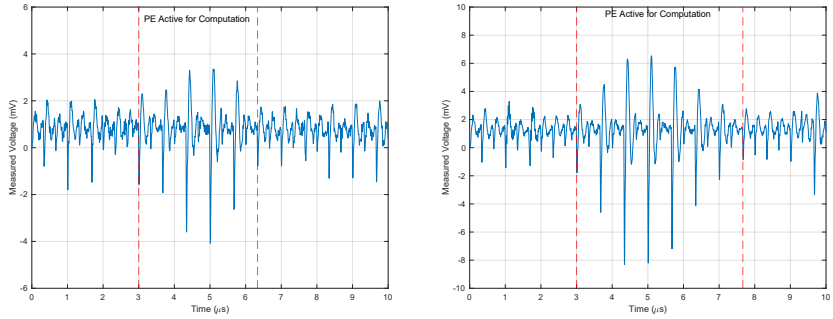


Fig. 5. Measured power traces processed by the low pass filter. Left: A trace from the dot product accelerator. Right: A trace from the matrix-vector multiplication accelerator.

clock cycles where PEs are involved in a MAC operation on inputs and weights. These clock cycles are the focus of the DPA. The voltage pulses in the middle of the highlighted region have a larger peak-to-peak value compared to other pulses, since more PEs are active at each MAC operation in these clock cycles due to the diagonal wave-front of inputs. We note that the MVM accelerator shows larger voltage pulses, because a larger systolic array is used.

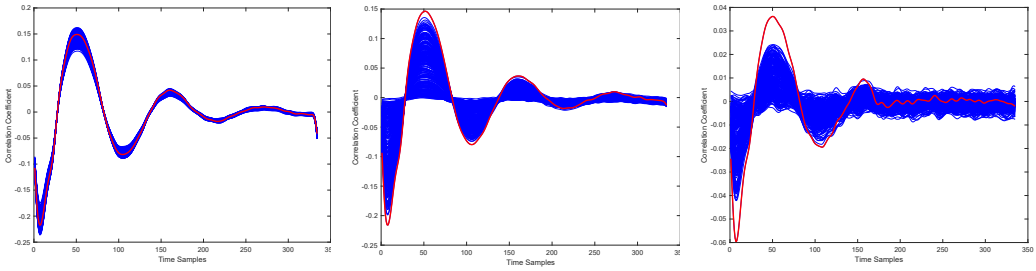
4 DPA ON DOT PRODUCT ACCELERATOR

We demonstrate how the DPA framework can be modified to successfully attack the dot product accelerator. The strategy allows the attack to succeed in fully identifying the weight vector pre-loaded on the accelerator. We demonstrate the details of the modified DPA framework that utilizes the dependence of the instantaneous MAC output on the earlier-processed weights. Without such a modification, the naive extension of the DPA, as it is used in attacks on common cryptographic ciphers, e.g. AES, fails.

We start by revisiting the algorithm for attacking AES [4]:

- The attacker collects N power traces with T samples per trace. Each trace corresponds to one encryption on a random input plaintext. The attacker arranges the collected power traces into a $N \times T$ matrix.
- For a target AES key byte, and for each AES encryption, the attacker calculates a hypothetical power value based on all 256 possible values of the key byte. The calculated hypothetical power values are arranged in a $256 \times N$ matrix.
- The attacker calculates Pearson correlation between each row of the power model matrix and each column of the power trace matrix. The correlation coefficients are arranged in a $256 \times T$ matrix. The row index with the largest value of the correlation matrix represents the attacker's best guess of the target key byte.

In the DPA on AES, each key byte of the entire key is extracted individually. The attacker typically focuses on the last round of AES encryption. We propose the following strategy for an attack on a DNN. The attack focuses on a single 8-bit weight at a time. The attacker first collects N power traces, each corresponding to a single dot product computation. Then, the attacker locates the clock cycles in the power traces where the MAC operation involving the weight occurred. We propose that the power model be based on the Hamming distance of the register holding the MAC result between consecutive inputs for each weight. The DPA starts analysis from the first weight in the weight vector and targets weights serially.



(a) DPA on w_{11}. (b) DPA on w_{21}. (c) DPA on w_{31}.

Fig. 6. DPA results with 100K power traces. DPA fails to retrieve correct w_{11} value directly, (a). If assume w_{11} is known, DPA succeeded to further retrieve w_{21} and w_{31}, (b) and (c). Red curves represent correct values: $w_{11} = 120$, $w_{21} = 73$ and $w_{31} = -96$.

First, we follow the above framework directly. We collect 100K power traces from the FPGA board, each corresponding to a dot product computation of three random 3×1 input vectors (the input FIFO depth is 3) and the fixed (secret) 3×1 weight vector. We start by focusing on the first weight w_{11}, which is associated with the first PE in the column. We use the Hamming distance of *Reg C* of PE1, over the first two consecutive inputs to form the power model. If we denote the two inputs as x_{11} and x_{21}, the power model H_{11} is:

$$H_{11} = HW[(x_{11} \times w_{11}) \oplus (x_{21} \times w_{11})] \tag{3}$$

We use the phase of the power traces that corresponds to the switching of the target register and organize data into a 100000×334 matrix. Then, for each possible value of w_{11}, we calculate H_{11}. The resulting power models are arranged in a 256×100000 matrix. Finally, we correlate each row of the power model matrix with each column of the power trace matrix, and select the value of w_{11} with the largest correlation as the attack guess.

Unfortunately, this direct procedure fails to retrieve the correct value of w_{11}, Fig. 6a. We describe a solution for finding w_{11} later. We first describe the strategy for extracting other weights, assuming w_{11} has already been retrieved and the partial sums produced by w_{11} can be calculated. We now construct the power model H_{21} for w_{21} as:

$$H_{21} = HW[(\sum_{i=1}^{2} x_{1i} \times w_{i1}) \oplus (\sum_{i=1}^{2} x_{2i} \times w_{i1})] \tag{4}$$

In Equation 4, x_{12} and x_{22} are the first two inputs within the input FIFO for w_{21}. We correlate the power models with the targeted portion of the power traces. The correlation coefficients for all possible values of w_{21} are shown in Fig. 6b. This time the guess for w_{21} is based on the highest correlation corresponding to the correct value. The power model H_{31} for the next weight w_{31} is:

$$H_{31} = HW[(\sum_{i=1}^{3} x_{1i} \times w_{i1}) \oplus (\sum_{i=1}^{3} x_{2i} \times w_{i1})] \tag{5}$$

Here, x_{13} and x_{23} are the first two consecutive input values into the input FIFO for w_{31}. The correlation for w_{31} is in Fig. 6c. The attack again succeeds to retrieve the correct value of w_{31}.

Based on the above discussion the weight w_{11} seems to be the bottleneck: w_{11} needs to be retrieved first in order to retrieve the subsequent weights w_{21} and w_{31}. It is critical to understand why the DPA fails to retrieve w_{11} directly. We note that w_{11} is the first element in the weight vector. This means that the previous partial sum input to PE1 is zero. Therefore, the MAC associated

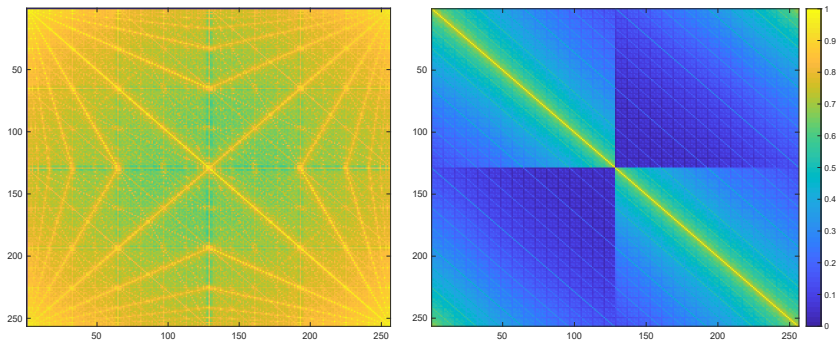


Fig. 7. Pairwise correlation across power models of all possible guesses on w_{11} (multiplication only, left) and w_{21} (multiplication and accumulation, right).

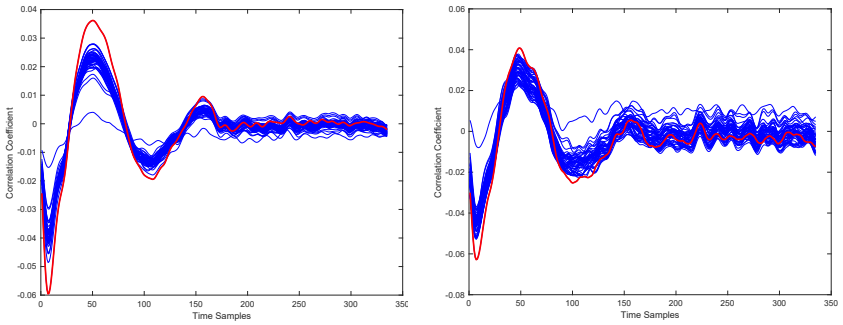
with w_{11} involves only multiplication. In contrast, the MACs related to w_{21} and w_{31} involve both multiplication and accumulation.

DPA examines the correlation between power traces and power models. It is critical that the hypothetical power models based on the incorrect guesses of the secret do not show correlation with the model based on the correct guess. If there is aliasing (correlation between the power model of the correct guess and that of a different guess), it will cause the incorrect guess to also show high correlation. This effectively reduces the confidence in the correct guess. In block ciphers, such as AES, this issue does not arise due to the fundamental non-linearity of the S-box [21].

In our case, aliasing occurs. We investigate this further to understand the difference in the behavior of w_{11} and w_{21}. We calculate pairwise correlation across 256 rows of the power model matrix for w_{11}, based on the DPA described above, and repeat the calculation for w_{21}. Fig. 7 shows the results plotted as 2D color maps. It can be seen that power models of different guesses on w_{11} show large correlation indicating large aliasing. Power models of different guesses of w_{21} show only a large self-correlation. Note that the difference is due to the absence of accumulation. (In the case of w_{11} extraction, the role of accumulation is intriguing and we plan to explore it further in the future.)

To overcome the difficulty of retrieving w_{11} directly, we use the MACs involving additional, subsequent weights to extract the correct w_{11}. Since the product generated by w_{11} also determines the MAC outputs for w_{21} and w_{31}, the hypothetical power model will show high correlation only for the correct weight combinations. To achieve this, we modify the attack procedure to be:

- Perform DPA on w_{11}. Construct the power model matrix using Equation 3 and the power trace matrix. Calculate the correlation between each row of the power model matrix and each column of the power trace matrix. Sort all guesses on w_{11} based on the maximum correlation found in the time window. Since the rank of correct guess on w_{11} is close to 50, we record the w_{11} guesses corresponding to 50 highest correlations.
- For each recorded guess on w_{11}, perform DPA on w_{21}. Construct the power model matrix using Equation 4 and the power trace matrix and calculate the correlations. Record the guess on w_{21} with the highest correlation and its corresponding w_{11} guess. This results in 50 (w_{11}, w_{21}) guesses.
- With each recorded (w_{11}, w_{21}) guess, perform DPA on w_{31}. Construct the power model matrix using Equation 5 and the power trace matrix and calculate the correlations. Record the guess on w_{31} with the highest correlation and its corresponding (w_{11}, w_{21}) guess. Return



(a) DPA with 100K traces. (b) DPA with 20K traces.

Fig. 8. Modified DPA framework successfully retrieves (w_{11}, w_{21}, w_{31}) with 100K power traces, (a). 20K traces is identified as the minimum number of traces required for a successful attack, (b). Red curves represent the correct combination: $(w_{11}, w_{21}, w_{31}) = (120, 73, -96)$.

the (w_{11}, w_{21}, w_{31}) guess with the highest correlation among all 50 recorded combinations as the final guess.

The time complexity of the modified DPA framework depends on how many guesses of w_{11} are recorded. The number of w_{11} guesses to record leads to a linear increase in computation complexity. We apply the above modified DPA framework to 100K power traces. The correct guess on (w_{11}, w_{21}, w_{31}) is successfully retrieved, as shown in Fig. 8a. We start with 10K power traces and increase the number of traces used in steps of 10K, repeating the experiments. We identify 20K traces as the Measurement to Disclosure (MTD) for the dot product accelerator. The correlation plot of DPA with 20K power traces is shown in Fig. 8b. We summarize the results of the attack on the dot product accelerator in Table 1. The Rank column shows the rank of the correct guess on (w_{11}, w_{21}, w_{31}) among all 50 recorded combinations and the Correlation column shows the Pearson correlation of the correct guess at 20K (MTD) power traces.

Weight	Correct Guess	Rank	Correlation	MTD
(w_{11}, w_{21}, w_{31})	(120, 73, -96)	1	0.0628	20000

Table 1. DPA results on the dot product accelerator. The correct weights can be retrieved with 20K traces.

5 HIGHER SECURITY OF MATRIX-VECTOR MULTIPLICATION ACCELERATOR

In this section, we study the vulnerability of a 2D matrix-vector multiplication accelerator to a DPA-style attack. We find that the 2D accelerator exhibits higher security. Both a conventional DPA and a stronger template-based DPA fail. We investigate the causes of higher resistance of the 2D array to the attack compared to the dot product accelerator. *We explain the reason for higher security of a 2D array design by the fact that it intrinsically results in multiple instantaneous MAC outputs being dependent on the same input.* We show that this is a fundamental feature of the commonly-used weight-stationary dataflow.

We first investigate the conventional DPA attack that uses the approach described above. A conventional DPA does not require access to an identical profiled device. We collect 460K power traces from the matrix-vector multiplication accelerator. (We stopped at 460K traces due to measurement

time budget.) For each PE column, the top 50 guesses of the first weight of each column (w_{11} for column 1, w_{12} for column 2, w_{13} for column 3) are recorded and the relevant phases of power traces are selected. However, the conventional DPA fails to retrieve the weights, as shown in Table 2, where NA indicates that the correct weight combination does not appear in the 50 recorded weight pairs.

Weight	Correct Guess	Rank	Correlation	MTD
(w_{11}, w_{21}, w_{31})	(23, -107, 74)	NA	NA	NA
(w_{12}, w_{22}, w_{32})	(120, 73, -96)	NA	NA	NA
(w_{13}, w_{23}, w_{33})	(-6, -31, 17)	1	0.0640	20000

Table 2. Results of conventional DPA on the matrix-vector multiplication accelerator. Conventional DPA fails to retrieve (w_{11}, w_{21}, w_{31}) and (w_{12}, w_{22}, w_{32}) with 460K power traces: the correct weight combination does not even appear in the 50 recorded weight pairs. (w_{13}, w_{23}, w_{33}) can be retrieved with 20K traces.

We now investigate a template-based DPA, which assumes a stronger adversary. Since DPA relies on the analysis of power consumed by MACs, to improve the effective SNR, we propose a profiling technique that removes all non-MAC power. We assume an attacker has full access to an identical device, which can be used for profiling. Hence, the attacker can modify the secret weights of the profiled device and collect its power traces. Specifically, the attacker sets all the weights of the profiled device to zero. A trace (template) from the profiled device captures the systolic array power minus the MAC power. The attacker produces a power template for each observed input. This means the same number of template power traces need to be collected from the profiled device as the target power traces. The attacker subtracts the template power trace from the target power trace, which is then used for DPA. We note that the proposed attack is different from the template attack of Chari et al. [6], which constructs a template using the mean trace and the noise covariance matrix for each key value. We call our attack the template-based DPA to reflect the fact that an identical device is used for profiling.

Weight	Correct Guess	Rank	Correlation	MTD
(w_{11}, w_{21}, w_{31})	(23, -107, 74)	32	0.0623	NA
(w_{12}, w_{22}, w_{32})	(120, 73, -96)	NA	NA	NA
(w_{13}, w_{23}, w_{33})	(-6, -31, 17)	1	0.0633	20000

Table 3. Results of the template-based DPA on matrix-vector multiplication accelerator. Template-based DPA fails to retrieve (w_{11}, w_{21}, w_{31}) and (w_{12}, w_{22}, w_{32}) with 460K traces. The correct combination for (w_{11}, w_{21}, w_{31}) appears in the 50 recorded weight pairs but does not show the highest correlation. (w_{13}, w_{23}, w_{33}) can be retrieved with 20K traces.

The attack just described allows extracting some, but not all, weights, even after collecting a much larger number of traces (460K traces). Table 3 shows that the template-based DPA fails to retrieve the weights of 2 out of 3 columns.

We verify our conclusions on the 2D matrix-vector multiplication accelerator by performing the attack using simulated noise-free power traces. We generate the simulated traces by modeling the register switching in each PE. The power consumption of each PE at a specific clock cycle is modeled as:

$$P_x = \alpha \cdot HD(Reg\ A) + \beta \cdot HD(Reg\ C) \tag{6}$$

The power contribution of *Reg A* and *Reg C* defines each term of the equation. $HD(\cdot)$ denotes the Hamming distance of the register values over two consecutive clock cycles. Coefficients α and β are used to adjust the contributions of registers in different PEs, based on their load capacitance. Specifically, for PE1, PE2, PE4, PE5, PE7 and PE8, we use $\alpha = 3$ and $\beta = 2$; for PE3, PE6 and PE9, we use $\alpha = 1$ and $\beta = 2$. We sum the power of each PE to get the total power at a specific clock cycle. This represents the power averaged over one clock cycle. We repeat the register-switching calculation for each compute-active clock cycle of the 2D systolic array, obtaining a simulated trace with 7 samples.

We generate 460K simulated noise-free traces using the same inputs as the 460K measured traces. We perform both the conventional DPA and template-based DPA with the simulated traces and summarize the results in Table 4 and 5.

Weight	Correct Guess	Rank	Correlation	MTD
(w_{11}, w_{21}, w_{31})	(23, -107, 74)	45	0.2627	NA
(w_{12}, w_{22}, w_{32})	(120, 73, -96)	NA	NA	NA
(w_{13}, w_{23}, w_{33})	(-6, -31, 17)	1	0.3878	10000

Table 4. Results of conventional DPA on matrix-vector multiplication accelerator with simulated noise-free traces. Only (w_{13}, w_{23}, w_{33}) are retrieved, which matches results in Table 2.

Weight	Correct Guess	Rank	Correlation	MTD
(w_{11}, w_{21}, w_{31})	(23, -107, 74)	35	0.2952	NA
(w_{12}, w_{22}, w_{32})	(120, 73, -96)	NA	NA	NA
(w_{13}, w_{23}, w_{33})	(-6, -31, 17)	1	0.4420	10000

Table 5. Results of template-based DPA on matrix-vector multiplication accelerator with simulated noise-free traces. Contribution of *Reg A* is removed from the simulated traces. Only (w_{13}, w_{23}, w_{33}) are retrieved, which matches results in Table 3.

We also explored the Hamming weight power models and bit-level power models in the template-based DPA. For a specific PE at a specific clock cycle, the Hamming weight power model is constructed as the Hamming weight of *Reg C*. The bit-level power model is constructed as the Hamming distance of a single bit of *Reg C* [19] over two consecutive cycles. (We selected one bit to explore the behavior but we believe the results do not depend on which bit is used.) We substitute the power models given by Equation 3 to 5 with the Hamming weight and bit-level power models and repeat the DPA described above. Unfortunately, *none* of the weights could be retrieved with the new power models even with 460K power traces, as shown in Table 6 and 7. We believe that the failure of the Hamming weight power model is due to the inaccurate capture of the PE power consumption while the failure of the bit-level model is due to the interference of switching of different register bits.

The 2D matrix-vector multiplication accelerator appears significantly less vulnerable to a DPA-style attack compared to a simpler 1D array. To understand the source of the improved resistance to DPA we conduct additional experiments. We investigate the exploratory case where only a single column of PE of the 3×3 systolic array is activated. In this case, the accelerator is essentially performing a dot product of the input vector with the weight column. The difference is that inputs still propagate horizontally across the PEs and contribute to power. We implement this case by pre-loading the weights of the target PE column only and pre-loading zero weights to the remaining PE columns.

Weight	Correct Guess	Rank	Correlation	MTD
(w_{11}, w_{21}, w_{31})	(23, -107, 74)	NA	NA	NA
(w_{12}, w_{22}, w_{32})	(120, 73, -96)	NA	NA	NA
(w_{13}, w_{23}, w_{33})	(-6, -31, 17)	NA	NA	NA

Table 6. Results of template-based DPA on matrix-vector multiplication accelerator with the Hamming weight power model. None of the weights can be retrieved with 460K traces.

Weight	Correct Guess	Rank	Correlation	MTD
(w_{11}, w_{21}, w_{31})	(23, -107, 74)	17	0.0080	NA
(w_{12}, w_{22}, w_{32})	(120, 73, -96)	6	0.0100	NA
(w_{13}, w_{23}, w_{33})	(-6, -31, 17)	6	0.0085	NA

Table 7. Results of template-based DPA on matrix-vector multiplication accelerator with the bit-level power model. None of the weights can be retrieved with 460K traces.

MACs of these columns do not contribute power since all their partial sums remain zero. For consistency with the previous template-based experiment, we assume the attacker has access to an identical device to collect power traces. We perform exploratory study for each individual PE column: the PE column is active while other columns are inactive. In each study, the templates (to be subtracted) are based on 100K traces. Table 8 summarizes the results of the exploratory studies for the individual PE columns.

Weight	Correct Guess	Rank	Correlation	MTD
(w_{11}, w_{21}, w_{31})	(23, -107, 74)	1	0.0448	30000
(w_{12}, w_{22}, w_{32})	(120, 73, -96)	1	0.0359	20000
(w_{13}, w_{23}, w_{33})	(-6, -31, 17)	1	0.0651	30000

Table 8. Results of template-based DPA for exploratory studies. Template-based DPA successfully retrieves weights of each individual PE column, while other PE columns are inactive. Corresponding MTDs are identified.

The experiments demonstrate that all weights in each PE column were retrieved, needing at most 30K traces. This confirms the vulnerability of the dot product to DPA. *It also proves that simultaneous MACs of different PE columns create a higher resistance to DPA by contributing power interfering with the selection of the correct hypothesis.*

We believe that this behavior, in which MAC operations of different columns cause issues for DPA, is a general characteristic of 2D accelerators based on the weight-stationary dataflow. The behavior is caused by *multiple MAC outputs depending on the same inputs simultaneously.* This is because the weight-stationary dataflow results in inputs to the matrix-vector multiplication array to be arranged in a diagonal wave-front format, as shown in Fig. 4. The MAC outputs of a PE and the PE on its lower left in its adjacent column (if applicable), will be affected by the same input(s) simultaneously. To illustrate this dataflow feature, we focus on PE2, PE3 and PE5. The MAC output to be computed in PE3, and the previous partial sum to PE5, are determined by the same input at a specific clock cycle. PE5 accumulates its input-weight products with the previous partial sum. This means the same input affects the MAC outputs of both PE3 and PE5 *simultaneously*, Fig. 9.

We assess the interference by examining the correlation between switching of *Reg C* of different PEs in the same clock cycle. Since *Reg C* holds a MAC output of each PE, the power due to the

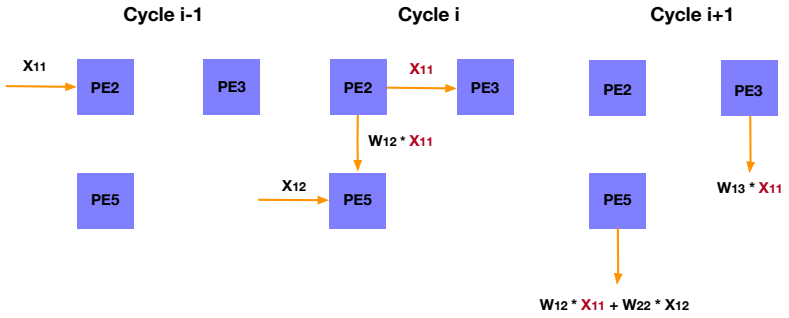


Fig. 9. Illustration of multiple simultaneous MAC outputs dependent on the same input with PE2, PE3, and PE5: At cycle $i-1$, input X_{11} propagates to PE2 (Left); At cycle i, the MAC output to be computed in PE3 and the previous partial sum to PE5 are determined by the same input X_{11} (Middle); At cycle $i+1$, the MAC output of both PE3 and PE5 depends on input X_{11} (Right).

switching of *Reg C* represents each PE's MAC power for the purpose of DPA correlation analysis. We calculate the pairwise correlation of Hamming distances of *Reg C* in PE1 to PE8, Table 9.

	PE1	PE2	PE3	PE4	PE5	PE6	PE7	PE8
PE1	1.00	0	0	0	0	0	0	-0.01
PE2	0	1.00	0	0.01	0	0	0	0
PE3	0	0	1.00	0	0.29	0	0.01	0
PE4	0	0.01	0	1.00	0	0	0	0
PE5	0	0	0.29	0	1.00	0	0.01	0
PE6	0	0	0	0	0	1.00	0	-0.13
PE7	0	0	0.01	0	0.01	0	1.00	0
PE8	-0.01	0	0	0	0	-0.13	0	1.00

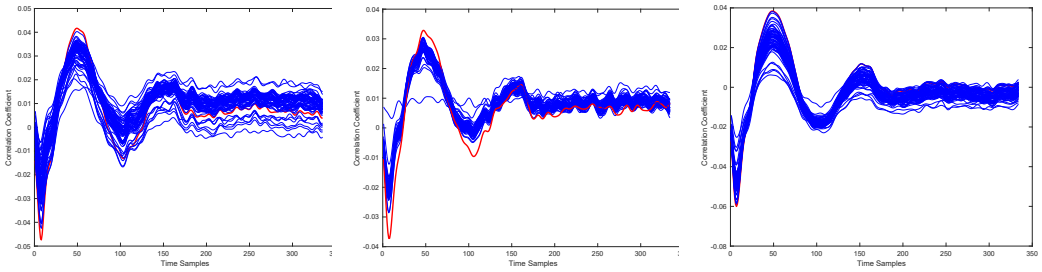
Table 9. Pairwise correlation of Hamming Distance of *Reg C* in PE1 to PE8.

We observe that some PEs exhibit large correlation in addition to self-correlation. Specifically, PEs whose MAC results are simultaneously affected by the same inputs show non-zero correlation: (PE2, PE4), (PE3, PE5), (PE5, PE7) and (PE6, PE8). Some of them, (PE3, PE5) and (PE6, PE8), show high correlation. As discussed earlier, such correlation will interfere with DPA's effectiveness and reduce the confidence of the correct hypothesis for a target PE.

In contrast, for a 1D array, such interference does not occur since the inputs stop propagation after being consumed by the PEs and the same input is never used across multiple PEs. The MAC outputs of different PEs in the 1D array, at any clock cycle, are determined by different inputs. Thus, MAC power of different PEs will *not* show correlation as shown by zero correlations between (PE1, PE4, PE7) and between (PE2, PE5, PE8) in Table 9.

6 BREAKING 2D ACCELERATOR WITH ENHANCED TEMPLATE DPA

In this section, we demonstrate a stronger template-based DPA that succeeds in fully retrieving the weights of the 2D array. The attack requires multiple profiling phases. We call it multi-phase template-based DPA.



(a) DPA on (w_{11}, w_{21}, w_{31}). (b) DPA on (w_{12}, w_{22}, w_{32}). (c) DPA on (w_{13}, w_{23}, w_{33}).

Fig. 10. Multi-phase template-based DPA results with 60K traces. The attack succeeds to fully retrieve the weights on matrix-vector multiplication accelerator. Red curves represent the correct combination.

As discussed, 2D array exhibits higher resistance to DPA due to parallel MAC operations of different PE columns. To fully retrieve the weights, it is critical to focus on each PE column individually and remove the interference of other columns. However, localizing leakage from each PE column is challenging because the power trace captures aggregate power of the entire 2D array. However, it is possible to expose the leakage of each PE column via a sequential analysis. Specifically, we identify the most vulnerable PE column, extract its weights, remove the effect of the column by template, and move to the next most vulnerable remaining PE column. The process can be repeated until the weights from all PE columns are retrieved. The main question is how to find the most vulnerable PE column of the 2D array.

Using previous results, we observe that the rightmost PE column (PE3, PE6 and PE9) appears to be the easiest to break. The attack can break the column with 20K power traces while the other columns remain secure even after 460K power traces. The reason for this behavior is that input propagation stops at the rightmost PE column. We believe that *the PE column furthest from inputs is the most vulnerable one and its weights can be retrieved most easily*. Based on the hypothesis, we propose the following attack. For simplicity, we use the term "trace" to refer to a power trace and "template trace" to refer to a power trace collected from a profiled device with the fixed weights.

- Perform DPA on column 3 (PE3, PE6, PE9). Run DPA for 1D array. Retrieve weights (w_{13}, w_{23}, w_{33}).
- Set the weights of PE3, PE6 and PE9 on the profiled device to (w_{13}, w_{23}, w_{33}). Set other weights to zero. Collect a phase-1 template trace for each input of the original set of traces.
- Subtract phase-1 template traces from the corresponding original traces. Perform DPA on column 2 (PE2, PE5, PE8) using updated traces. Run DPA for 1D array. Retrieve weights (w_{12}, w_{22}, w_{32}).
- Set the weights of PE3, PE6 and PE9 on the profiled device to (w_{13}, w_{23}, w_{33}). Set the weights of PE2, PE5 and PE8 to (w_{12}, w_{22}, w_{32}). Set other weights to zero. Collect a phase-2 template trace for each input of the original set of traces.
- Subtract phase-2 template traces from the corresponding original traces. Perform DPA on column 1 (PE1, PE4, PE7) using updated traces. Run DPA for 1D array. Retrieve the final set of weights (w_{11}, w_{21}, w_{31}).

We collect 60K traces from the matrix-vector multiplication accelerator and perform the above attack. The process requires 120K template traces to be collected in total. The attack succeeds in retrieving *all* weights from the 2D array, Fig.10. We summarize the results for each PE column in Table 10, identifying the MTD for each column.

Weight	Correct Guess	Rank	Correlation	MTD
(w_{11}, w_{21}, w_{31})	(23, -107, 74)	1	0.0588	20000
(w_{12}, w_{22}, w_{32})	(120, 73, -96)	1	0.0398	40000
(w_{13}, w_{23}, w_{33})	(-6, -31, 17)	1	0.0755	20000

Table 10. Results of multi-phase template-based DPA for matrix-vector multiplication accelerator. The attack retrieves all weights from the accelerator with a MTD of 40K traces.

We now analyze the time complexity of the multi-phase template-based DPA. The attack needs to apply a 1D DPA for each profiling phase of an individual PE column. Therefore, the complexity of the attack is proportional to the number of PEs in the 2D array. The cost of template construction is proportional to the number of PE columns. The complexity also depends on the number of guesses to record for the first weight of each PE column.

7 DISCUSSION

The time complexity of the DPA-based model extraction attack scales up linearly with the number of weights. Each weight vector loaded onto the systolic array is retrieved individually. To retrieve a large DNN model, the cost scales up with the model size. The linear increase of cost due to model size is inevitable in both side-channel-based [3, 8, 36] and query-based [5, 14] model extraction attacks. Techniques to reduce MTD of each individual weight vector need to be investigated for a higher attack efficiency.

Attacks based on EM measurements have the potential to further improve attack efficiency. EM attacks allow to localize leakage from individual components of the circuit, which can improve the SNR of the collected traces significantly. We anticipate that EM attacks with a high resolution EM probe, that is able to localize leakage from individual PE columns or even PEs, can break the 1D/2D arrays faster.

In this work, spatial DNN accelerators are shown to be vulnerable to DPA-based model extraction attacks. It is useful to consider some countermeasures to reduce or even eliminate side-channel leakage. Various countermeasures for embedded cryptographic hardware have been demonstrated over the years, such as masking [2], [23], which adds random values to intermediate computations, and hiding [29], [31], [18], which aims to hide the power draw of the actual computation. These techniques are likely to also work on DNN accelerators. For instance, the masking scheme could be adopted to obfuscate the intermediate partial sums generated by MAC operations, which would break the correlation between the power models and the actual power consumption. Hiding could be realized by using dual-rail logic, or adding noise to hide the actual MAC power.

8 CONCLUSION

We investigate the vulnerability of spatial DNN accelerators using a general 8-bit number representation to DPA-style attacks. Specifically, we investigate two systolic array architectures based on the weight-stationary dataflow: (1) a 3×1 array for a dot-product operation, and (2) a 3×3 array for matrix-vector multiplication. Both are implemented on the SAKURA-G FPGA board. We show that both architectures are ultimately vulnerable. A conventional DPA succeeds fully on the 1D array, requiring 20K power measurements. However, the 2D array exhibits higher security even with 460K traces. We show that this is because the 2D array intrinsically entails multiple MACs simultaneously dependent on the same input. However, we find that a novel template-based DPA with multiple profiling phases is able to fully break the 2D array with only 40K traces. Novel countermeasures need to be investigated to protect spatial DNN accelerators from such attacks.

9 ACKNOWLEDGEMENTS

This research was made possible by the support from the National Science Foundation under award 1901446. We also thank the anonymous reviewers for their feedback.

REFERENCES

[1] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. 2014. Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22, 10 (2014), 1533–1545.

[2] Mehdi-Laurent Akkar and Christophe Giraud. 2001. An Implementation of DES and AES, Secure against Some Attacks. In *Cryptographic Hardware and Embedded Systems — CHES 2001*, Çetin K. Koç, David Naccache, and Christof Paar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 309–318.

[3] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2019. CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 515–532. https://www.usenix.org/conference/usenixsecurity19/presentation/batina

[4] Eric Brier, Christophe Clavier, and Francis Olivier. 2004. Correlation Power Analysis with a Leakage Model. In *Cryptographic Hardware and Embedded Systems - CHES 2004*, Marc Joye and Jean-Jacques Quisquater (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 16–29.

[5] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. 2020. Cryptanalytic Extraction of Neural Network Models. In *Advances in Cryptology – CRYPTO 2020*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer International Publishing, Cham, 189–218.

[6] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. 2003. Template Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, Burton S. Kaliski, çetin K. Koç, and Christof Paar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 13–28.

[7] G. E. Dahl, D. Yu, L. Deng, and A. Acero. 2012. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing* 20, 1 (2012), 30–42.

[8] Anuj Dubey, Rosario Cammarota, and Aydin Aysu. 2019. MaskedNet: The First Hardware Inference Engine Aiming Power Side-Channel Protection. arXiv:1910.13063 [cs.CR]

[9] Vasisht Duddu, Debasis Samanta, D Vijay Rao, and Valentina E. Balas. 2018. Stealing Neural Networks via Timing Side Channels. arXiv:1812.11720 [cs.CR]

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs.CV]

[11] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861 [cs.CV]

[12] Xing Hu, Ling Liang, Lei Deng, Shuangchen Li, Xinfeng Xie, Yu Ji, Yufei Ding, Chang Liu, Timothy Sherwood, and Yuan Xie. 2019. Neural Network Model Extraction Attacks in Edge Devices by Hearing Architectural Hints. arXiv:1903.03916 [cs.CR]

[13] Weizhe Hua, Zhiru Zhang, and G. Edward Suh. 2018. Reverse Engineering Convolutional Neural Networks through Side-Channel Information Leaks. In *Proceedings of the 55th Annual Design Automation Conference* (San Francisco, California) *(DAC '18)*. Association for Computing Machinery, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.1145/3195970.3196105

[14] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. 2019. High Accuracy and High Fidelity Extraction of Neural Networks. arXiv:1909.01838 [cs.LG]

[15] Norman P. Jouppi and et al. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. *SIGARCH Comput. Archit. News* 45, 2 (June 2017), 1–12. https://doi.org/10.1145/3140659.3080246

[16] H. Kung and R. L. Picard. 1984. One-Dimensional Systolic Arrays for Multidimensional Convolution and Resampling.

[17] G. Li, V. Iyer, and M. Orshansky. 2019. Securing AES against Localized EM Attacks through Spatial Randomization of Dataflow. In *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 191–197.

[18] P. Liu, H. Chang, and C. Lee. 2010. A Low Overhead DPA Countermeasure Circuit Based on Ring Oscillators. *IEEE Transactions on Circuits and Systems II: Express Briefs* 57, 7 (2010), 546–550.

[19] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2007. *Power Analysis Attacks*. Springer.

[20] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2016. Practical Black-Box Attacks against Machine Learning. arXiv:1602.02697 [cs.CR]

[21] Emmanuel Prouff. 2005. DPA Attacks and S-Boxes. In *Fast Software Encryption*, Henri Gilbert and Helena Handschuh (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 424–441.

[22] Muhammad Imran Razzak, Saeeda Naz, and Ahmad Zaib. 2018. *Deep Learning for Medical Image Processing: Overview, Challenges and the Future*. Springer International Publishing, Cham, 323–350. https://doi.org/10.1007/978-3-319-65981-7_12

[23] Matthieu Rivain and Emmanuel Prouff. 2010. Provably Secure Higher-Order Masking of AES. In *Cryptographic Hardware and Embedded Systems, CHES 2010*, Stefan Mangard and François-Xavier Standaert (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 413–427.

[24] Joshua Saxe and Konstantin Berlin. 2015. Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features. arXiv:1508.03096 [cs.CR]

[25] Patrick Schaumont and Zhimin Chen. 2012. *Side-Channel Attacks and Countermeasures for Embedded Microcontrollers*. Springer New York, New York, NY, 263–282. https://doi.org/10.1007/978-1-4419-8080-9_11

[26] Dinggang Shen, Guorong Wu, and Heung-Il Suk. 2017. Deep Learning in Medical Image Analysis. *Annual Review of Biomedical Engineering* 19, 1 (2017), 221–248. https://doi.org/10.1146/annurev-bioeng-071516-044442 arXiv:https://doi.org/10.1146/annurev-bioeng-071516-044442 PMID: 28301734.

[27] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy (SP)*. 3–18.

[28] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel Emer. 2017. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. arXiv:1703.09039 [cs.CV]

[29] Kris Tiri, David Hwang, Alireza Hodjat, Bo cheng Lai, Shenglin Yang, Patrick Schaumont, and Ingrid Verbauwhede. [n.d.]. Prototype IC with WDDL and Differential Routing - DPA Resistance Assessment. In *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop*. Springer, 354–365.

[30] S. Tobiyama, Y. Yamaguchi, H. Shimada, T. Ikuse, and T. Yagi. 2016. Malware Detection with Deep Neural Network Using Process Behavior. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 2. 577–582.

[31] C. Tokunaga and D. Blaauw. 2009. Secure AES engine with a local switched-capacitor current equalizer. In *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*. 64–65,65a.

[32] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing Machine Learning Models via Prediction APIs. In *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, Austin, TX, 601–618. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer

[33] S. Wang, D. Zhou, X. Han, and T. Yoshimura. 2017. Chain-NN: An energy-efficient 1D chain architecture for accelerating deep convolutional neural networks. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*. 1032–1037. https://doi.org/10.23919/DATE.2017.7927142

[34] Y. Xiang, Z. Chen, Z. Chen, Z. Fang, H. Hao, J. Chen, Y. Liu, Z. Wu, Q. Xuan, and X. Yang. 2020. Open DNN Box by Power Side-Channel Attack. *IEEE Transactions on Circuits and Systems II: Express Briefs* (2020), 1–1.

[35] Mengjia Yan, Christopher W. Fletcher, and Josep Torrellas. 2020. Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Boston, MA. https://www.usenix.org/conference/usenixsecurity20/presentation/yan

[36] K. Yoshida, T. Kubota, M. Shiozaki, and T. Fujino. 2019. Model-Extraction Attack Against FPGA-DNN Accelerator Utilizing Correlation Electromagnetic Analysis. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 318–318.