applied sciences

MDPI

Article

The Limits of SEMA on Distinguishing Similar Activation Functions of Embedded Deep Neural Networks

Go Takatoi [1,*], Takeshi Sugawara [1], Kazuo Sakiyama [1], Yuko Hara-Azumi [2] and Yang Li [1,*]

[1] Department of Informatics, The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan; sugawara@uec.ac.jp (T.S.); sakiyama@uec.ac.jp (K.S.)
[2] Department of Information and Communications Engineering, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550, Japan; hara@cad.ict.e.titech.ac.jp
* Correspondence: g.takatoi@uec.ac.jp (G.T.); liyang@uec.ac.jp (Y.L.)

Abstract: Artificial intelligence (AI) is progressing rapidly, and in this trend, edge AI has been researched intensively. However, much less work has been performed around the security of edge AI. Machine learning models are a mass of intellectual property, and an optimized network is very valuable. Trained machine learning models need to be black boxes as well because they may give away information about the training data to the outside world. As selecting the appropriate activation functions to enable fast training of accurate deep neural networks is an active area of research, it is important to conceal the information of the activation functions used in a neural network architecture as well. There has been research on the use of physical attacks such as the side-channel attack (SCA) in areas other than cryptography. The SCA is highly effective against edge artificial intelligence due to its property of the device computing close to the user. We studied a previously proposed method to retrieve the activation functions of a black box neural network implemented on an edge device by using simple electromagnetic analysis (SEMA) and improved the signal processing procedure for further noisy measurements. The SEMA attack identifies activation functions by directly observing distinctive electromagnetic (EM) traces that correspond to the operations in the activation function. This method requires few executions and inputs and also has little implementation dependency on the activation functions. We distinguished eight similar activation functions with EM measurements and examined the versatility and limits of this attack. In this work, the machine learning architecture is a multilayer perceptron, evaluated on an Arduino Uno.

Keywords: machine learning; deep learning; side-channel; activation function; SEMA

Citation: Takatoi, G.; Sugawara, T.; Sakiyama, K.; Hara-Azumi, Y.; Li, Y. The Limits of SEMA on Distinguishing Similar Activation Functions of Embedded Deep Neural Networks. *Appl. Sci.* **2022**, *12*, 4135. https://doi.org/10.3390/app12094135

Academic Editors: Guy Gogniat, Vianney Lapotre and Maria Mushtaq

Received: 10 January 2022
Accepted: 16 April 2022
Published: 20 April 2022

1. Introduction

Artificial intelligence (AI) such as neural networks has been progressing rapidly to fit a wide range of applications in recent years. For example, image recognition [1,2], Internet-of-Things-based speech emotion detection [3], robotics [4], natural language processing [5], medical science [6,7], and security [8–10] are areas in which deep learning is being used. However, machine learning models that are trained with high costs and include confidential information from the training phase could quickly become a target for attack. Therefore, security over AI is a growing concern.

Lately, communication, privacy, and latency issues have caused certain deep neural networks to be calculated on edge devices. Due to the accessibility of edge devices, physical attacks such as side-channel attacks could be used to recover the architectures and parameters of neural networks. Side-channel attacks are used to break specific neural network accelerators [11,12]. Furthermore, although physical proximity is not necessary, in a setting where the attacker and the victim are located on the same machine and sharing its cache, the cache side-channel attack to recover neural network architectures is demonstrated to attack edge AI [13,14].

A recent work by Batina et al. showed that a black box multilayer perceptron (MLP) implemented on an 8 bit CPU and convolutional neural network implemented on a 32 bit CPU can be fully reverse-engineered using merely side-channel leakages [15]. The recovery of the network architecture is composed of four key parameters: the activation function, the pre-trained weights, the number of hidden layers, and the number of neurons in each layer. In this work, we focused on revealing the activation function. In [15], the activation function was discerned by timing attack from its distinctive computation time. The distribution patterns of the activation function calculation time were compared with the profile of each activation function to determine it. We replicate this attack in Section 3 and used it on four more activation functions to determine the strength of the attack.

The timing attack used in [15] had several limitations. Therefore, a different approach that uses the patterns of the electromagnetic (EM) trace directly to discern the computed operation was proposed. Takatoi et al. [16] identified activation functions using simple electromagnetic analysis (SEMA). The idea was that since the operations leak information into the measurements, the operations from the measurements could be recovered if the noise could be reduced from the trace. The signal processing in their work allowed the peaks to have distinctive features for each operation. The target activation functions and the implementation platform were the same for both [15,16]. By comparing SEMA-based identification and timing-based identification, Takatoi et al. considered SEMA as a better approach since it requires less measurements and is more resistant to implementation variations.

In [16], the four activation functions identified were selected according to the popularity of those functions. Two of the activation functions are very easy to identify from the others due to the distinct calculation complexity, and only two are left to compare with the same calculation complexity. However, there are many similar activation functions proposed and used in AI [17–19]. This work additionally selected another four activation functions in the activation function set to investigate the limitations of SEMA in identifying the similar activation functions. We were able to match the features to the activation functions for identification by using the improved signal processing and inputs for the activation functions. We believe this attack could be applied to different neural network models if the activation functions are being computed in the network without side-channel countermeasures.

The SEMA attack in this paper successfully identifies eight activation functions explained in Section 2 that were implemented on an Arduino Uno. By comparing SEMA-based identification and timing-based identification in a similar scenario, we considered SEMA to be more preferable as it requires less measurements and is more resistant to implementation variations.

We chose to recover the activation functions because they are used in many neural networks and play a very important role in the network. Non-linear functions are used as activation functions to output a result from the sum of the inputs and to solve non-linear problems. Designing and choosing activation functions that enable fast training of accurate deep neural networks is an active area of research. From this, it can be said that it is important to conceal the information of activation functions used in a neural network architecture.

As for the required measurements, SEMA requires only one or a few averaged EM traces, compared to the timing attack, which requires thousands of EM traces. Furthermore, we expect less implementation dependency, since compared to the timing information, the SEMA utilizes more information to identify the activation functions. We observed the activation function operations from the peaks of the EM trace; thus, this method has potential to overcome constant time mitigation as well.

In this work, we make the following contributions:

- We simulated the timing attack performed by Batina et al. [15] to recover the activation functions on various similar activation functions. The analysis of the timing behavior of the Leaky ReLU function [20], SELU function [17], Swish function [18], and Fast

Sigmoid function [19] was first performed in this work and showed that all of the activation functions could be identified. However, the different implementations of activation functions made for more difficult classification and showed the potential versatility of the SEMA-based attack to different activation function implementations. This contribution corresponds to Section 3.

- We utilized the idea proposed by Takatoi et al. [16] of using SEMA to identify activation functions to further upgrade the attack and analyze the limits of SEMA-based identification. Our work was also implemented and demonstrated on Arduino Uno. We improved the signal processing for noisy measurements by erasing the trend patterns that were not intrinsic to the data. We then report the results of the SEMA attack performed on an MLP using eight similar activation functions to show the limits of this attack. The SEMA attack can observe the differences in operations, but cannot observe the differences in values from the traces. Although this makes it difficult for the SEMA attack to identify activation functions with the same operations with differing values in the functions, this attack is still an effective method to distinguish many popular activation functions. This contribution corresponds to Section 5.
- We discuss the activation functions that are difficult to identify with the timing attack and SEMA attack and the potential of the SEMA attack to be used on different platforms, such as a GPU. This contribution corresponds to Section 5.6.

The rest of this article is organized as follows. In Section 2, we briefly review the background of this work, including the activation functions used in this article. In Section 3, we explain the timing-based activation function identification. In Section 4, we explain the experimental setup of the SEMA attack. In Section 5, we describe the identification of similar activation functions using the SEMA attack and discuss the limits of the attack. In Section 6, we survey and compare the related works. Finally, we summarize this work in Section 7.

2. Background

In this section, we describe the details and the architecture of the neural network (NN). We also give details on side-channel analysis, more specifically simple electromagnetic analysis.

2.1. Neural Network

2.1.1. Multilayer Perceptron

An MLP is made of fully connected layers with a given weight and seen as one of the most simple types of NNs. The output y of a neuron is calculated as Equation (1) where $(x_1, x_2, \ldots, x_n)$ represents the inputs, $(w_1, w_2, \ldots, w_n)$ represents the weights, b represents the bias, and $h(a)$ represents the activation function.

$$y = h\left(\sum_{i=1}^{n} x_i \times w_i + b\right) \tag{1}$$

The bias in this formula is often programmed as the weight of an input value 1.

An MLP is represented in Figure 1. An MLP must have at least three layers, composed of at least one input layer, hidden layer, and output layer to be considered a deep learning architecture. The MLP used in this work consists of an input layer, two hidden layers, and an output layer, as illustrated in Figure 1.

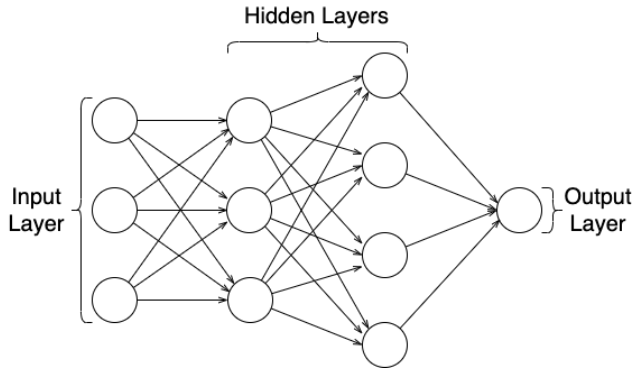

Figure 1. Multilayer Perceptron.

2.1.2. Activation Functions

In [15,16], the activation function set recovered included four activation functions, which are the Sigmoid function, Tanh function, softmax function, and rectified linear unit (ReLU) function. These functions were selected for the popularity of the functions. In this work, we also used the Leaky ReLU function [20], SELU function [17], Swish function [18], and Fast Sigmoid function [19].

For the identification, the ReLU and softmax functions are very easy to be identified from others due to the distinct calculation complexity. The only challenge one has is to distinguish the Tanh and Sigmoid functions, as only these two have similar calculation complexities. However, there are many similar activation function proposed and used in AI. Using the SEMA attack proposed in previous work, we attempted to distinguish more activation functions. We chose four activation functions that have similar operations to the functions used in our previous work to examine the versatility of this attack in identifying the activation functions.

A brief introduction of them is as follows:

Sigmoid The Sigmoid (logistic) function, i.e., Equation (2), is a nonlinear function that plots arbitrary inputs to outputs in the range $(0, 1)$.

$$h(a) = \frac{1}{1 + e^{-a}} \tag{2}$$

Tanh The Tanh function, i.e., Equation (3), is a rescaling of the Sigmoid function. The Tanh function maps arbitrary inputs to outputs in the range $(-1, 1)$. Different from Sigmoid, Tanh is symmetric by the origin.

$$h(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}} = \frac{2}{1 + e^{-2a}} - 1 \tag{3}$$

Softmax The softmax function, i.e., Equation (4), can map values into several outputs (or classes), the sum of which becomes 1. The output range is $(0, 1)$. It is able to be seen as a probability and is used for classification problems.

$$\begin{aligned} h(\mathbf{a})_j &= \frac{e^{a_j}}{\sum_{k=1}^{K} e^{a_k}}, \text{ for } j = 1, \ldots, K \\ \text{and } \mathbf{a} &= (a_1, \ldots, a_K) \in \mathbb{R}^K \end{aligned} \tag{4}$$

ReLU The ReLU function, i.e., Equation (5), is an extremely simple function, which can reduce the time of training and computing. It is mainly used as an activation function for deep neural networks (DNNs).

$$h(a) = \begin{cases} 0, & \text{for } a \leq 0 \\ a, & \text{for } a > 0 \end{cases} \tag{5}$$

Leaky ReLU The Leaky ReLU function, i.e., Equation (6), is similar to the ReLU function, except that it enables back propagation, even for negative input values by having the positive slope proposed in [20], where $\beta = 0.01$.

$$h(a) = \begin{cases} \beta a, & \text{for } a \leq 0 \\ a, & \text{for } a > 0 \end{cases} \tag{6}$$

SELU The SELU function, i.e., Equation (7), is an improvised version of the ReLU function for self-normalizing neural networks, a type of feed-forward neural network [17]. The SELU function has parameters λ and α, which have flexible values according to the applications, while we fixed them as $\lambda = 1.0507$, $\alpha = 1.6733$.

$$h(a) = \lambda \begin{cases} \alpha(e^a - 1), & \text{for } a \leq 0 \\ a, & \text{for } a > 0 \end{cases} \tag{7}$$

Swish The Swish function, i.e., Equation (8), is a self-gated activation function proposed in [18]. Swish is extremely similar to the Sigmoid function, where the flexible β parameter is fixed as $\beta = 1$ in this work.

$$h(a) = \frac{a}{1 + e^{-\beta a}} \tag{8}$$

Fast Sigmoid Some activation functions include the exponential function, which is expensive to compute. Timmons et al. evaluated several function approximation techniques to replace the original function with faster execution at the cost of some loss of accuracy [19]. We used one of them in our work, which we call the Fast Sigmoid function, i.e., Equation (9). This function is a cheap alternative to the original Sigmoid function with good enough accuracy. The flexible n parameter was fixed as $n = 1$ and $n = 9$ in this work.

$$h(a) = \frac{1}{1 + e^{-a}}, e^a = (1 + \frac{a}{2^n})^{2^n} \tag{9}$$

2.2. Simple Electromagnetic Analysis

Side-channel analysis analyzes internal operation by collecting some side-channel information, which comprises unintentional physical leakages such as the execution time, the power consumption, or the electromagnetic emissions. Kocher et al. proposed practical side-channel analysis to perform key recovery attacks on the implementation of cryptographic algorithms [21,22]. They presented an attack called simple power analysis (SPA), which involves visually inspecting power consumption traces collected during cryptographic operations to recover the secret key [21].

We used SEMA in our work, which is similar to SPA, but analyzes information by interpreting electromagnetic measurements instead of power consumption measurements. SEMA is a type of side-channel analysis that targets information through sensitive computation from one or a few electromagnetic traces. SEMA can analyze internal operation from electromagnetic traces that show unique patterns due to differences in the data.

In this paper, we utilized this technique to distinguish the activation functions in the deep neural network. Even though the parameters and architectures of the DNN model are encrypted, it needs to be decrypted during the computation phase. Therefore, by measuring EM emanations from the CPU of the edge AI device where the sensitive computation is being calculated, we are able to recover information about the activation functions.

3. Identification Using Timing

In [15], Batina et al. used timing information to identify activation functions. Specifically, the activation function was discerned by the distribution of the calculation timing for a set of inputs. The timing information of the activation function was measured from EM traces. The measurements were taken when the neural network was processing random inputs in the range they had chosen beforehand. A total of 2000 EM measurements were captured for each activation function. By plotting the processing time of each activation function by inputs, distinct signatures can be seen from each timing behavior. By making a profile, timing patterns or averaged timing can be compared with the profile of each function to determine the activation function. Hereafter, we discuss the limitations of the approach by recreating the attack with the activation functions described in Section 2.

To analyze the timing attack, we replicated the timing attack. We used the Arduino Uno simulator [23] to recreate the timing behavior, as shown in Figure 2a,c,d. We also performed the timing attack on similar activation functions, shown in Figure 2b,e–h. The timing delay is displayed in µs. The experiments were performed using an Arduino Uno simulator and Tinkercad Circuits without the use of physical hardware [23]. By using the simulator, we were able to gain more data due to the fast process. The processing times were averaged 10 times per input. We measured the calculation time of the activation functions with a micros function, which returns how many microseconds passed since the Arduino board started processing. Therefore, there is some difference in timing delay to when acquiring with the oscilloscope. However, this is in principle the same as measuring on a real system, as it counts the number of clocks, which is precise enough for our experiment.

3.1. Timing Attack on Activation Functions

The Tanh function is distinguished from the Sigmoid function since it is more symmetric in the timing pattern for positive and negative inputs compared to Sigmoid, as shown in Figure 2a,c. We also recreated the attack on the Swish function, drawn in Figure 2b. By observing the timing signatures, it is difficult to tell the Sigmoid function apart from the Swish function. The two functions are computed in a very similar way, and the pattern is indistinguishable with the timing attack. However, the minimum computation time for the Sigmoid function and Swish function is slightly different, being the only element to distinguish.

The three functions, ReLU, Leaky ReLU, and SELU, have a similar computation time; however, they have different equations for positive and negative inputs, causing distinct patterns. The timing patterns of these functions are depicted in Figure 2d–f. The ReLU function and Leaky ReLU function are very similar and are a mirror image of each other. The Leaky ReLU function has a independent computation time at input 0; however, this is due to the implementation method. The Leaky ReLU function, i.e., Equation (6), can also be implemented like Equation (10) to blend the computation time of input 0 with the other positive inputs.

$$h(a) = \begin{cases} \beta a, & \text{for } a < 0 \\ a, & \text{for } a \geq 0 \end{cases} \tag{10}$$

By comparing Figure 2a,b,d–f, the SELU function appears to be a mix of a pattern similar to the ReLU function for positive inputs and a pattern similar to the Sigmoid function for negative inputs. The unique pattern for the negative input is due to the exponential operation in the function.

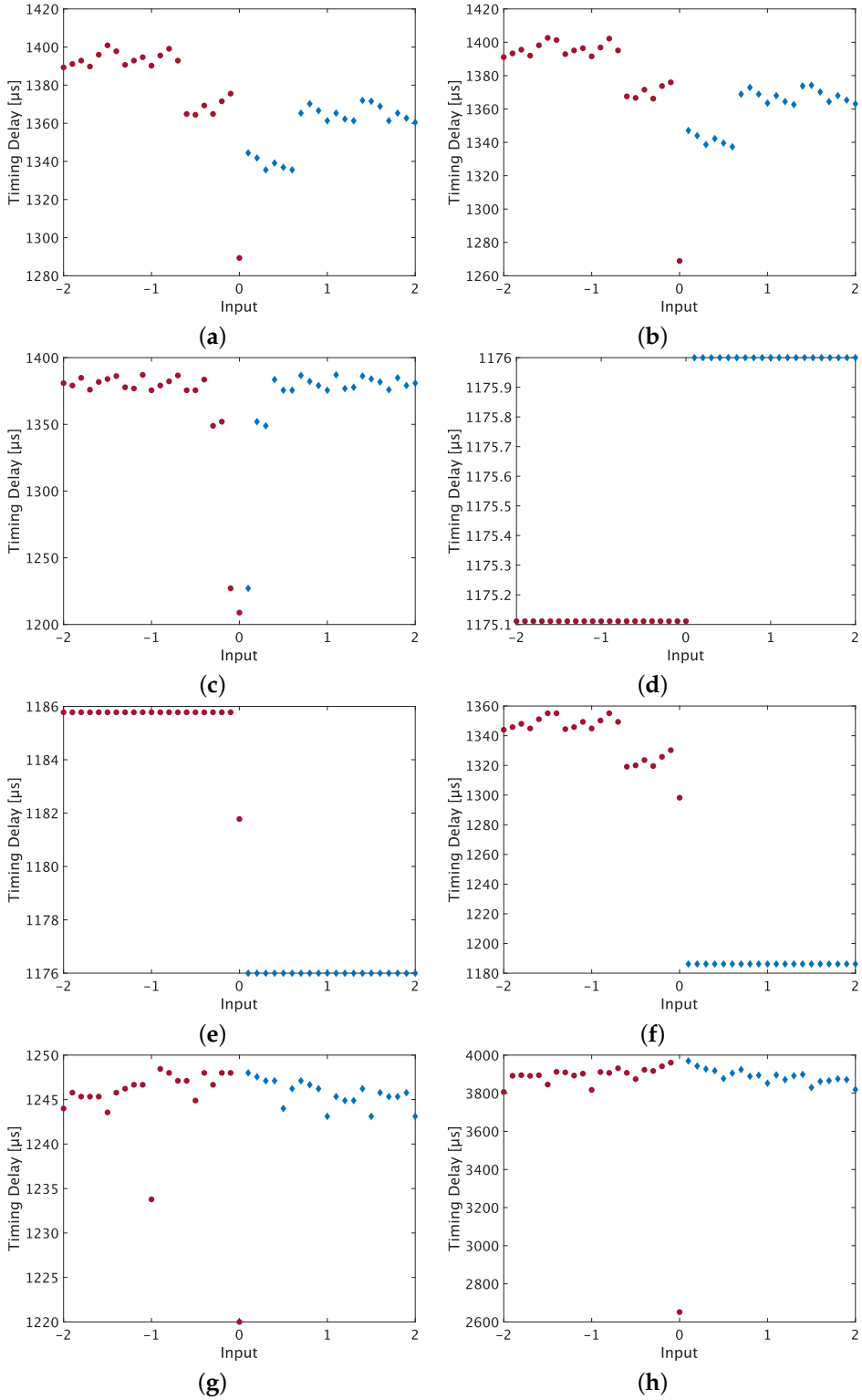

Figure 2. Timing Behavior of Similar Activation Functions. (**a**) Timing Behavior for Sigmoid. (**b**) Timing Behavior for Swish. (**c**) Timing Behavior for Tanh. (**d**) Timing Behavior for ReLU. (**e**) Timing Behavior for Leaky ReLU. (**f**) Timing Behavior for SELU. (**g**) Timing Behavior for Fast Sigmoid when $n = 1$. (**h**) Timing Behavior for Fast Sigmoid When $n = 9$.

The Fast Sigmoid function is an approximation of the Sigmoid function. By observing Figure 2g,h, the patterns are very unique to the other functions. The timing patterns appear to be symmetrical with some individual timing for some inputs. The computation times also differ quite significantly from the other functions, making for easy classification.

The features of the timing behavior can be summarized as follows:

- The Sigmoid and Swish functions are the same pattern, with a slightly different minimum computation time.
- The Tanh function is not symmetrical compared to the Sigmoid function, which has a similar computation time.
- The ReLU and Leaky ReLU functions are mirror images of each other with a similar computation time.
- The SELU function is similar to the Sigmoid function for negative inputs and similar to the ReLU function for positive inputs.
- The Fast Sigmoid function is symmetrical, yet significantly different from the other functions in the computation time.

3.2. Limitations

Although some functions such as the SELU function and Fast Sigmoid function are easily recognized from the timing patterns, we believe the other functions will be hard to define.

First, the distinct signatures of the timing behavior for activation functions could depend on the implementation method. The two activation functions used in [15], the Sigmoid and Tanh functions, were distinguished by the differences in the timing pattern. The Tanh function has a symmetric pattern compared to the Sigmoid function. However, Takatoi et al. implemented the Tanh function differently, causing the symmetric pattern to break and have a similar timing pattern to the Sigmoid function [16]. They insisted that the timing pattern for activation functions can change depending on the implementations, making them hard to characterize. The Taylor series approximation, for example, is one way to implement the exponential function. By using such an approximation, the Sigmoid function could be computed in a uniform timing, making the function a common pattern with no uniqueness. By trading off the computation time, the ReLU and Leaky ReLU functions could also be processed in a uniform timing by adding a dummy operation for either negative or positive inputs. Using such countermeasures, this timing attack is not as effective.

The other limitation is that thousands of measurements with uniformly distributed activation function inputs are required to plot the distribution of the calculation time. In [15], timing measurements were performed when the network was processing random inputs in the range, i.e., $x \in \{-2, 2\}$, and Takatoi et al. found that this input refers to the inputs to the activation function [16]. To plot the timing behavior for different activation functions based on the inputs, there would be a need to calculate the uniform input to the activation functions. They argued that without the knowledge of the weights, the bias, and the number of neurons in the layer, this timing attack will be difficult. Even with the knowledge of the weights, bias, and network structure, it would be difficult to obtain all of the inputs needed to make the profile of the timing behavior. To plot this graph specifically as in this previous work, a total of 2000 inputs are needed, making this graph difficult to plot, as a complex calculation is needed.

Accordingly, we used an attack method to substitute the timing-based attack. The attack is a SEMA-based approach first proposed by Takatoi et al. [16] that has potential to overcome these limitations.

4. Experimental Setup

Here, we specify the considered scenario and the hardware setup.

4.1. Threat Model

There was a set of activation functions, and the corresponding pre-trained networks were implemented in the C++ language and compiled on the target edge device. These pre-trained networks use proprietary models and, thus, should be black box. The activation functions in those pre-trained networks are concealed from the attacker. The attacker is

motivated to identify the activation functions used in the black box network. The attacker's capability is as follows.

- The attacker has no insight on the architecture of the network.
- The attacker is capable of accessing the target device and acquiring EM emanations. However, the attacker is non-invasive and can only operate the device normally such as accessing random inputs to the network.
- The attacker knows what set of activation functions could be implemented on the architecture.

The goal of the attacker is to reveal what activation function is being used in the network. The activation functions used in a black box network are intellectual property, and we can use these data to fully reverse engineer the neural network architecture as well.

4.2. Hardware Setup

Next, we briefly describe the experimental setup of EM measurement. We used the same setup as [16]. An MLP that calculates a 3-input XOR was implemented on the microcontroller Atmel ATmega328P on the microcontroller board Arduino Uno.

The C++ standard library functions were used to implement some operations including the exponential function, the Tanh function, and exponentiation. For the MLP with the softmax function, the dimensions for the 2 hidden layers were 3 neurons for the first hidden layer and 9 neurons for the second hidden layer. The MLP with the rest of the activation functions used in this work had 2 hidden layers with 9 neurons in both layers. We used three inputs for the activation function in the neural network, 4.154561, −1.443156, and 0, which we later refer to as positive inputs, negative inputs, and input 0. We chose these numbers as the input, as it was easy to calculate these numbers from the pre-trained weights in our MLP. By recovering weight parameters with correlation electromagnetic analysis proposed in [15], we can then use them and the inputs to the network to acquire the numbers, in our case 4.154561, −1.443156, and 0.

We used the Tektronix MSO64 oscilloscope with an RF-U 5-2 near-field EM probe from Langer to collect EM measurements. All measurements were 500 M samples/s. We also used a low-pass filter BLP-50+ from Mini-Circuits with the cutoff frequency of 48 MHz. We used the filter to obtain a clear signal from the measurements. We decapsulated the microcontroller by scraping the outer package to improve the quality of measurement of the microcontroller [24]. The EM probe was then placed above the decapsulated chip. The full measurement setup is depicted in Figure 3.

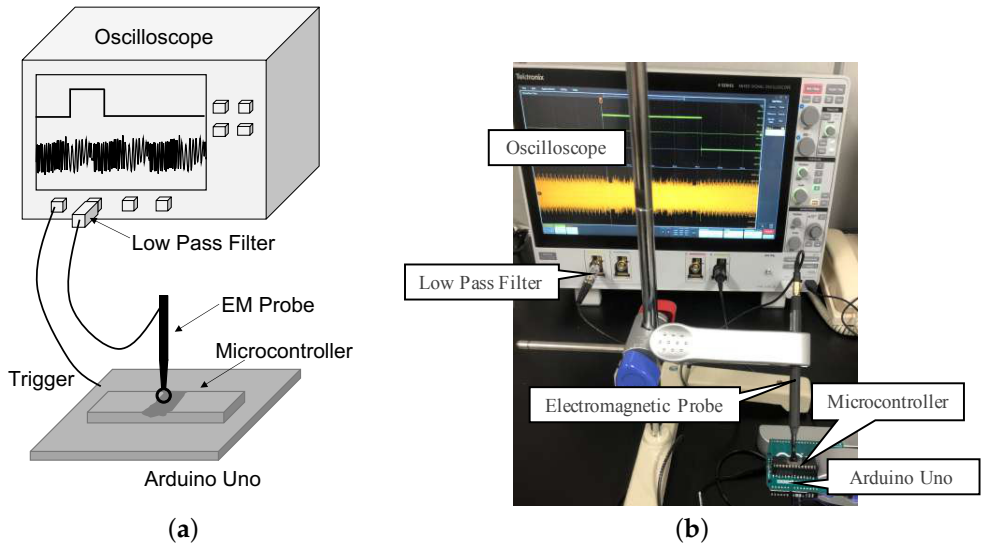

Figure 3. Experimental Setup. (**a**) Measurement Setup Model. (**b**) Full Measurement Setup.

5. Identification Using SEMA

In this section, we explain the identification of similar activation functions using SEMA. The identification was performed in the following procedures:

1. We first acquired the side-channel trace from the device. In our work, we used EM traces.
2. For the next step, we applied signal processing to the acquired trace to obtain the desired trace for visual inspection.
3. We then observed the trace and split the trace into two major parts, the multiplication operation of the weights and the activation function phase.
4. We classified the traces by the computation time of the activation functions. This was to compare the patterns of peaks in the next step with only similar activation functions that are difficult to distinguish with just timing.
5. Finally, we distinguished the similar activation functions by the pattern of the peaks.

5.1. Signal Processing

After the EM measurement taken from the microcontroller, we applied several steps to retrieve distinctive patterns from the EM trace as signal processing. We used a 5-step methodology to obtain the desired EM trace. We improved the signal processing used in [16] to be used in situations with more noise in the traces.

First, a measurement (or trace) was taken from the target device. Next, averaging was applied to the measured trace. Then, to extract the peaks, we computed the upper half of the EM trace's envelope. At this point, by smoothing the trace, the desired trace was acquired. For the last step, we erased the trend patterns from the trace that hindered the analysis. To make the peak stand out from the averaged trace, we applied signal processing using Matlab. The upper half of the EM trace's envelope was calculated to check the peak of the trace. However, if only the envelope was calculated, the pulse component still remained. Next, we smoothed the EM trace by removing the high-frequency signal using a Gaussian-weighted moving average filter. Smoothing was applied to the noisy traces keeping important patterns in our measurements while leaving out noise. Finally, to erase the trend patterns that were not intrinsic to the data, which can hinder the analysis, we fit a low-order polynomial to the measurement and subtracted it. After the signal processing, the patterns can be compared more easily with smoothing. The multiplication and activation function can be easily distinguished with their different patterns.

By observing the patterns of the multiplication operation, the weight multiplication, the addition of the outputs, and the addition of the bias can be distinguished from the trace. The multiplication operation is a repeating pattern before the activation function. The activation function can be observed after the repeating pattern of multiplications, and this signal processing allows the unique patterns in the traces to be visually inspected.

5.2. Classification Using Calculation Time

Here, we analyze the processed measurements to discern the activation functions. First, we start by grouping the activation functions by computation time. We analyzed and grouped the computation time of activation functions to make the comparison between the similar activation functions easy. We only used several inputs for this as this phase is merely to make the comparison easier between the similar activation functions, not to completely identify the activation functions. Table 1 presents the computation time of the activation functions. The approximate range of computation time in this table is the timing that we observed from our setup.

Table 1. Classification of Similar Activation Functions Using Calculation Time.

Activation Function	Calculation Time	Approximate Range of Computation Time (in μs)
ReLU, Leaky ReLU, SELU	Short	5∼15
Sigmoid, Tanh, Swish	Medium	80∼210
Softmax, Fast Sigmoid	Variable	50∼

The softmax function and the Fast Sigmoid function both include exponentiation operations that repeat the multiplication of the base. The exponent can vary upon the the number of neurons in the output layer or implementation method. The complexity of the softmax function causes it to take the longest time to process. This activation function can be easily distinguished from the computation time. For the Fast Sigmoid function, it can take a short to medium time to process, depending on the trade-off between complexity and accuracy.

The Sigmoid function, Tanh function, and Swish function have a medium computation time. The exponential function exists in these 3 functions, however not as much as the softmax function, which is the reason for the computing time being far less than the softmax function.

The ReLU function is the simplest function out of all of the functions and can be computed in a very short amount of time. The Leaky ReLU and SELU functions are improvised versions of the ReLU function and therefore have similar computation times. However, the SELU function includes the exponential function for negative inputs, hence having similar computation times to the Sigmoid function when processing negative inputs.

Next, for similar activation functions, we investigated the processed leakage patterns through the patterns of the EM traces.

5.3. Sigmoid, Tanh, and Swish

Figure 4 compares the leakage patterns of the Sigmoid, Tanh, and Swish functions with positive inputs, in this case 4.154561. It can be observed that the Sigmoid and Swish functions are identical; however, the Tanh function has a slight difference in peaks. Although there is no obvious gap in the processing time, the peak patterns differ, and extra peaks can be seen in two sections from the Tanh function when comparing with the Sigmoid and Swish function.

The extra peaks observed from the Tanh function are surrounded in a red box shown in Figure 4b. The first peak can be seen right after the multiplication. The second peak can be seen in the latter half of the activation function. The differences in peaks comes from the difference in the functions. The Tanh function has an additional multiplication and subtraction compared with the Sigmoid function. We observed that these operations cause the difference in the peak patterns. The Sigmoid function and Tanh function can be distinguished with the different peak patterns, with the Tanh function having more peak patterns.

The processed leakage patterns of the Sigmoid and Swish functions with input 0 are shown in Figure 5. When processing input 0, the Swish function has 0 as the numerator, reducing the time for the division operation compared to the Sigmoid function. The division operation for the Sigmoid function is surrounded in a red box, and the division operation for the Swish function is surrounded in a green box. It can be clearly observed that the Swish function is shorter in computation time and also has fewer peaks compared to the Sigmoid function.

The identification of activation functions by SEMA is mainly performed by observing the differences in peak patterns and the number of peaks. The differences in the shapes of the traces and peaks originates from the variability in the operations of the functions. The operations and peaks of the traces have a relation, allowing SEMA to distinguish the activation functions. The differences in just the value, however, do not cause changes to the peak patterns and number of peaks. The only difference between the Swish and

Sigmoid function is just the value of the numerator, which resulted in the same EM trace peak patterns. The same could be said for the peaks for the ReLU function with positive and negative inputs. We believe that the limit of SEMA is that only the operations become visible from the EM trace. Therefore, it is important to observe the timing for when the numerator is 0, because the computation will become much faster.

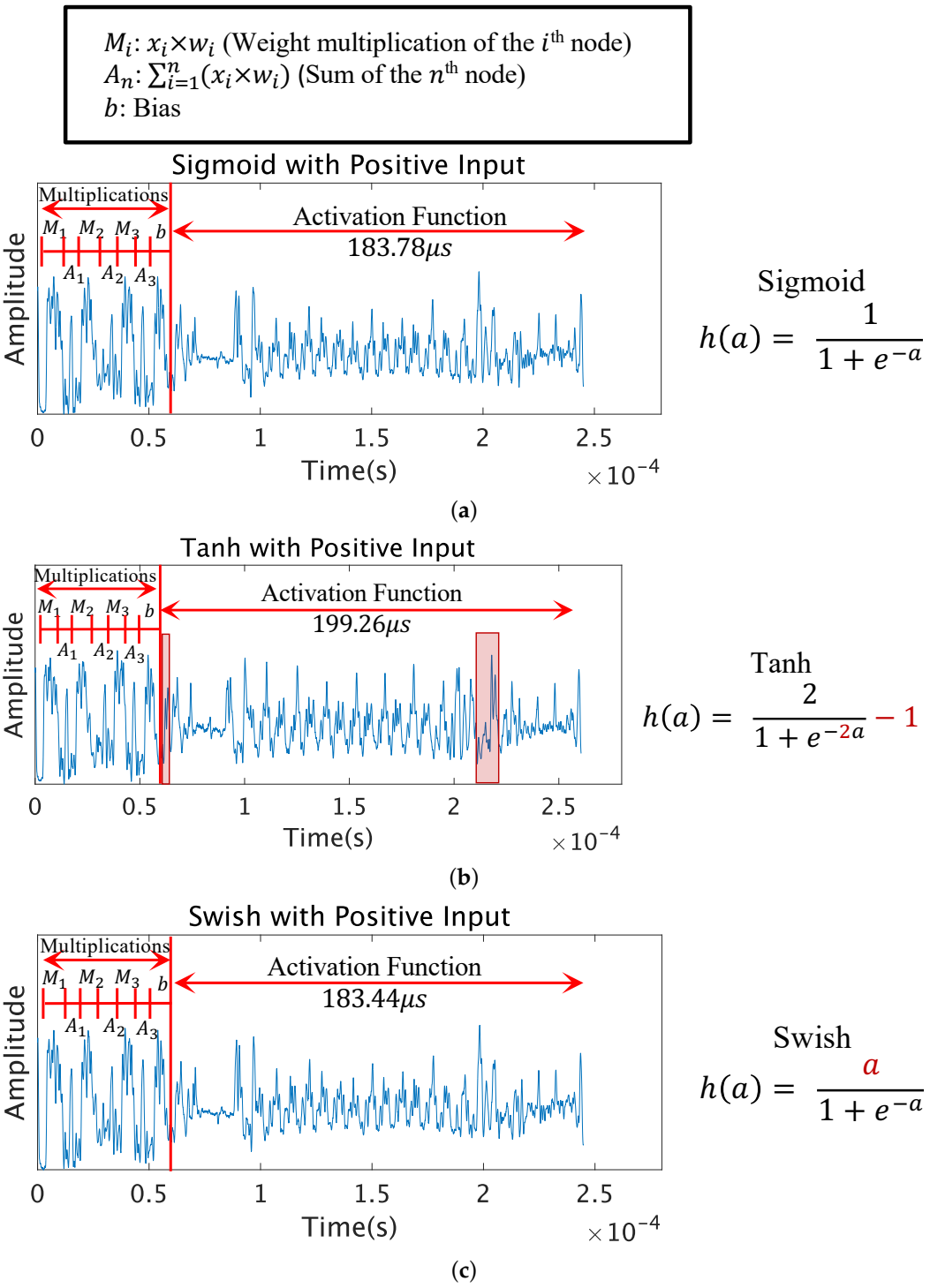

Figure 4. Comparison of the Patterns of Sigmoid, Tanh, and Swish Functions for Positive Input. (**a**) Sigmoid Function. (**b**) Tanh Function. (**c**) Swish Function.

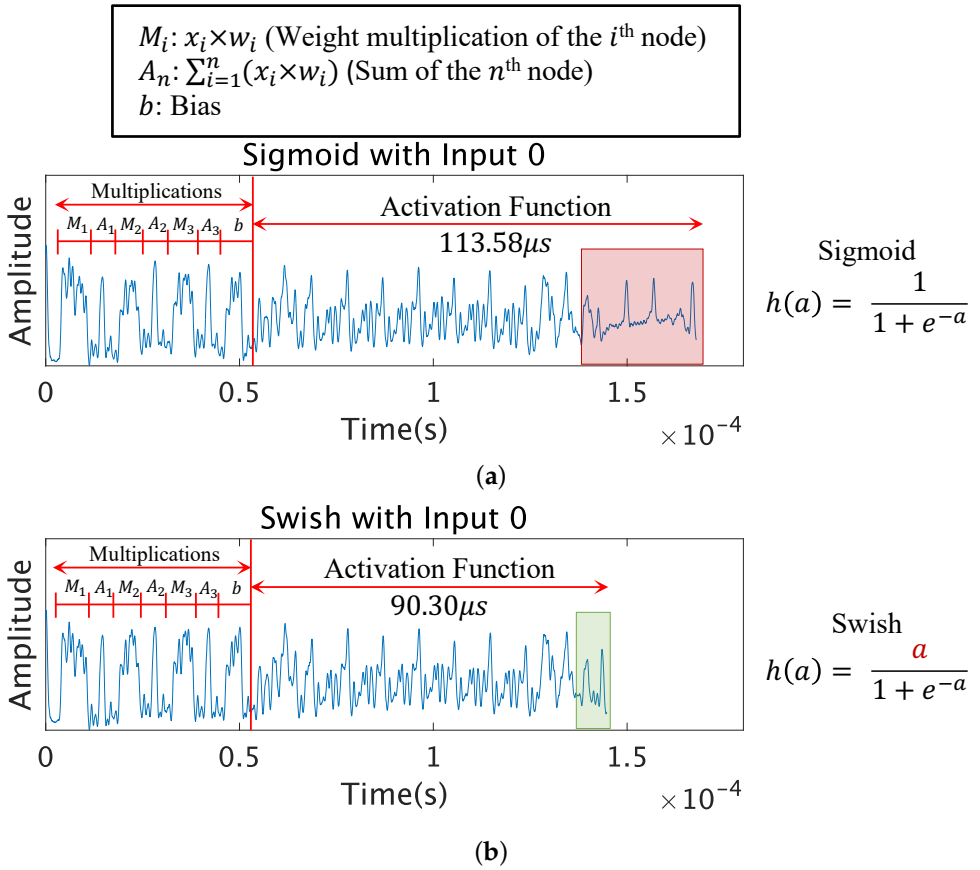

Figure 5. Comparison of the Patterns of Sigmoid and Swish Functions for Input 0. (**a**) Sigmoid Function. (**b**) Swish Function.

5.4. ReLU, Leaky ReLU, and SELU

Due to the use of similar activation functions, the functions will become more difficult to distinguish just by the computation time. We decided to observe the Leaky ReLU and SELU functions along with the popular ReLU function, which are extremely similar to the ReLU function in terms of the operations it calculates. The ReLU, Leaky ReLU, and SELU functions have different outputs for positive and negative inputs. Therefore, we measured the traces of these functions with positive and negative inputs, to see if the traces would change for different inputs.

Figures 6–8 compare the leakage patterns of the ReLU function, Leaky ReLU function, and SELU function with positive and negative inputs. It can be observed from this graph that the ReLU function has the same peak patterns for both positive and negative inputs. However, the patterns differ for the Leaky ReLU function and SELU function. Table 2 presents the comparison of the peak patterns seen in our setup with limited test trials. The Leaky ReLU function with positive inputs has the same peak patterns as the ReLU function, and the negative inputs have the same peak patterns as the SELU function with positive inputs. The SELU function with negative inputs has a very similar peak pattern to the Tanh function, with slightly different peaks in the latter half of the activation function.

Table 2. Comparison of Peak Patterns for ReLU, Leaky ReLU, and SELU.

	ReLU with PI	**Leaky ReLU with PI**	**SELU with PI**
ReLU with NI	Same	Same	-
Leaky ReLU with NI	Same	-	Same
SELU with NI	-	-	-

PI: positive inputs; NI: negative inputs.

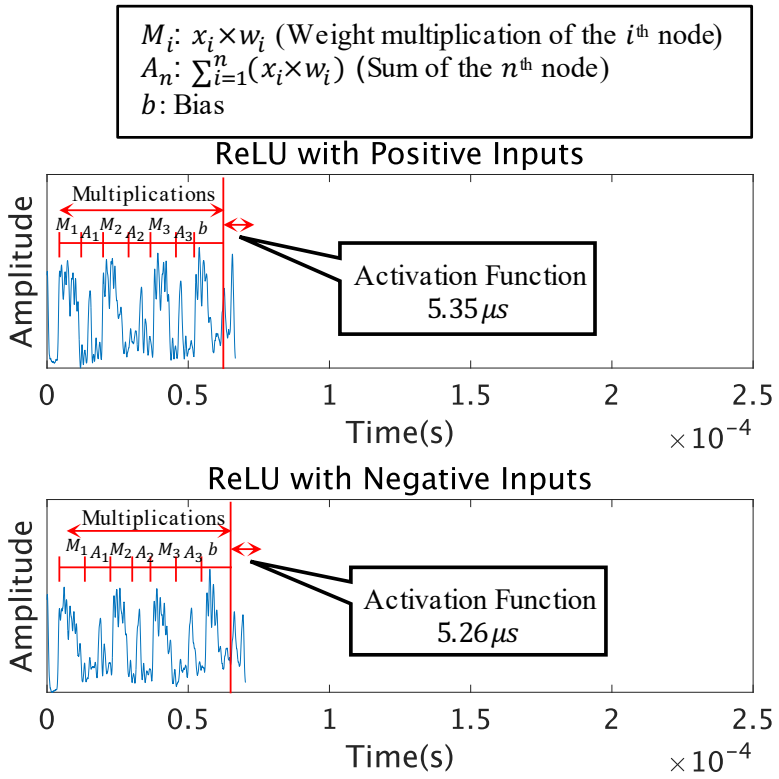

Figure 6. Extracted Pattern Measurements of the ReLU Function.

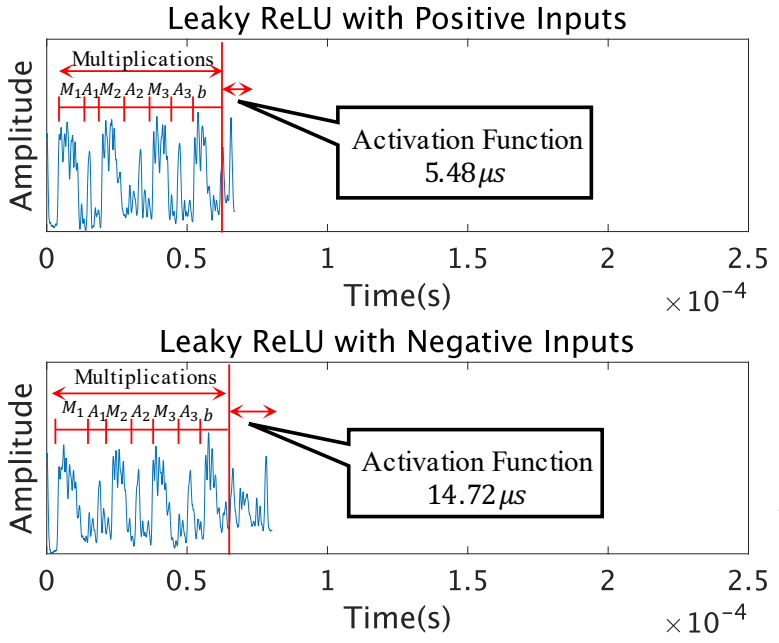

Figure 7. Extracted Pattern Measurements of the Leaky ReLU Function.

From this, it can be said that these activation functions could be distinguished if the attacker has access to the inputs to measure the traces of the activation functions with positive and negative inputs.

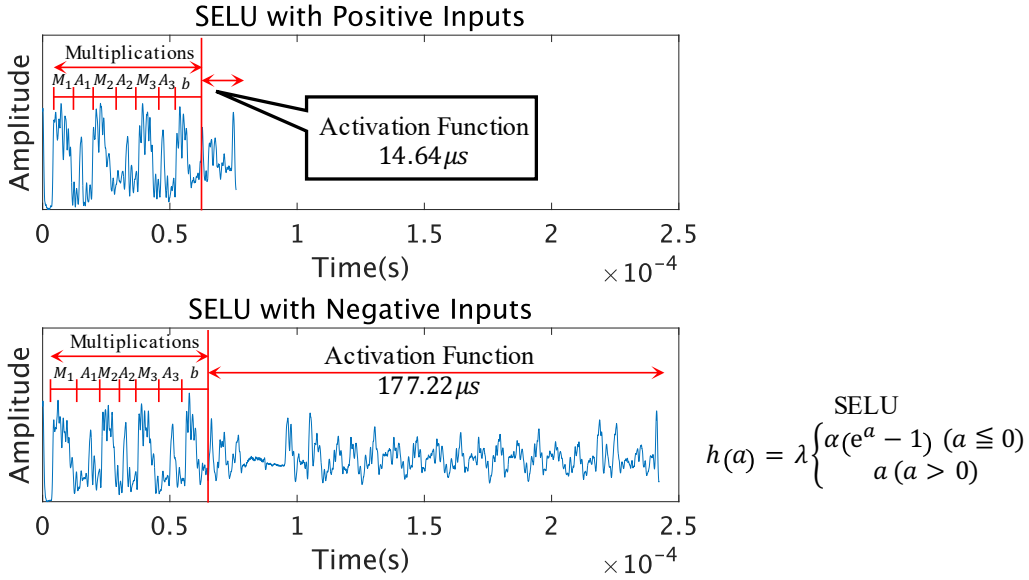

Figure 8. Extracted Pattern Measurements of the SELU Function.

5.5. Softmax and Fast Sigmoid

The softmax function computes the exponential function operation several times; therefore, the pattern of the exponential function will repeat itself for the number of neurons in the output layer as shown in Figure 9. If the number of neurons in the output layer is n, the exponential function pattern will repeat itself n times. This makes the trace longer than the other activation functions.

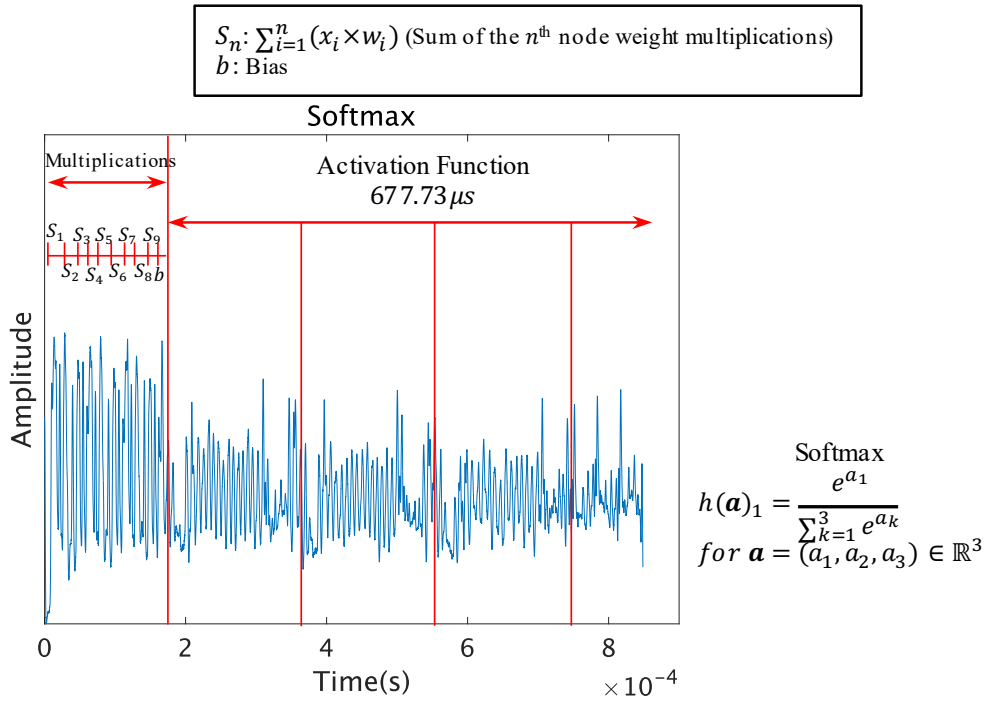

Figure 9. Processed Peak Patterns of Softmax Function.

The Fast Sigmoid function includes the exponentiation operation. The approximation of the exponential function has a unique pattern; therefore, it can be easily distinguished from the other functions as shown in Figure 10. When $n = 1$ from Equation (9), the peak patterns were very unique and had a shorter computation time than the Sigmoid function.

When $n = 9$ from Equation (9), the peak patterns were also unique, however having a resemblance to the Sigmoid function in the latter half of the activation function.

The softmax function and Fast Sigmoid function both had unique patterns, causing easy identification from pre-characterization of the peak patterns. We also believe that the Fast Sigmoid function has the possibility of being a constant-time implementation of the exponential function. The timing attack has difficulty in distinguishing constant-time activation function, making the potential of SEMA advantageous for the timing attack in our setup.

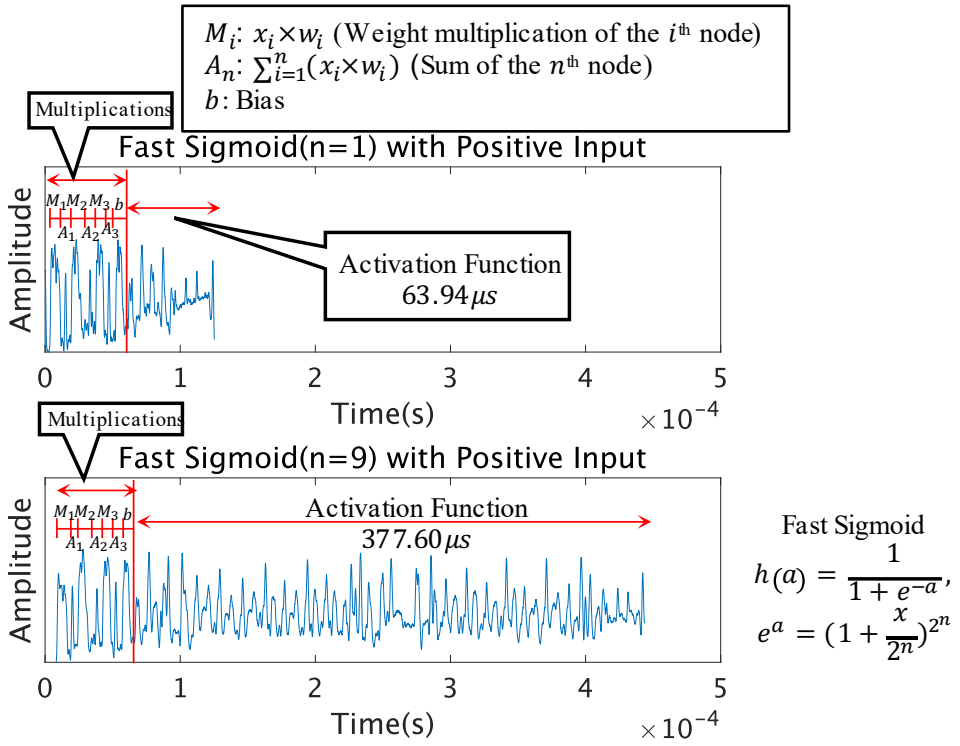

Figure 10. Processed Peak Patterns of Fast Sigmoid Function.

5.6. Discussion

We were able to distinguish the eight activation functions. We acquired the same peaks for the Sigmoid function, Tanh function, softmax function, and rectified linear unit (ReLU) function as the work by Takatoi et al. [16]. However, these popular activation functions become very difficult to distinguish with the similar activation functions. The Sigmoid function and Swish function are difficult to distinguish by SEMA except when processing the input 0. The identification by SEMA was performed through observing the differences in peak patterns coming from the variability in the operations of the functions. We believe the operations and peaks of the traces have a relation, allowing SEMA to distinguish the activation functions, but the limit of SEMA is that difference in values will be ignored. From the results, it can be noted that using activation functions with the same operations with different values would be the safest way to make a black box neural network, one example being the Sigmoid function or Swish function.

There are neural networks implemented on different platforms, such as a GPU platform or a FPGA platform. We believe the SEMA-based approach and methodology could be applied to different platforms, such as these similar microcontroller platforms, because side-channel analysis has been demonstrated on these platforms in several previous works [12,15,25]. One work by Chmielewski et al. reverse-engineered neural network designs, such as the number of layers and neurons, as well as types of activation functions on GPUs through SEMA [26]. When we carefully observe the traces recovered by Chmielewski et al., we can see the difference in peak patterns through different types of

activation functions. Though the peaks are not necessarily the exact same as the ones on our microcontroller, by using the same method to extract the peaks and pre-characterize them, we believe the peaks can be used to distinguish activation functions more clearly with reasoning than with just computation time, as done in their work.

The proposed SEMA attack can be combined with the full reverse-engineering of the neural network in [15]. By making a template and pre-characterizing the operations in the EM trace, activation functions can be distinguished with SEMA. Using the SEMA-based identification to replace the timing-based activation function identification, we believe the reverse-engineering can become a more versatile attack.

6. Related Works

As DNNs are intellectual property, there are many attacks targeting these network models and parameters. DNNs on edge devices especially are quickly becoming a target. There are many model extraction attacks being researched using side-channel attacks. Some of the previous works were focused on an NN embedded on an FPGA accelerator. Dubey et al. used power analysis to extract weights and bias from the adder-tree function of the binarized neural network (BNN) hardware by focusing on the switching activity of registers [27]. Yu et al. used EM side-channel information leakages to accurately recover the large-scale BNN accelerators embedded on an FPGA [11]. This study is based on a black box setting, but can only collect EM side-channel information under inference operations and observe the networks' outputs. They recovered the architecture of a targeted BNN through the analysis of EM emanations from the data bus and built substitute models with both a classification performance and layout architecture similar to the victim BNNs on the same hardware platforms using such information. Regazzoni et al. targeted a practical BNN hardware and showed that the overall structure of the BNN can be identified by the corresponding EM emanations [28]. In addition, they targeted the convolution layers. With actual experiments, the weight values convolved in the layers were revealed by EM side-channel attacks. There are also works around GPUs. Hu et al. used EM emanations to recover the architecture in a black box setting, but requiring physical access to the system [29]. Their objective was to learn the relation between extracted architectural hints such as the volumes of memory reads/writes obtained by side-channel and model internal architectures, and the results showed that a high accuracy of model extraction could be achieved. There are also works on VGG architectures. The black box NN extraction attack was presented by recovering the layer's depth through exploiting timing side-channel information [30]. Duddu et al. [30] utilized a reinforcement-learning-based neural architecture search to search for the optimal substitute model with functionality and test accuracy similar to the victim networks.

Maji et al. demonstrated that even simple power and timing analysis enable recovering an embedded neural network model with different precision on three microcontroller platforms [31]. They used the same microcontroller platform Atmel ATmega328P; however, they did not distinguish activation functions like in our work. Wang et al. demonstrated that one can utilize the cache-based SCA of Flush+Reload to spy on DNN models' computations and recover the label information [32].

There also exists side-channel analysis, which targets the parameters of the NN model, one of the parameters being activation functions. Batina et al. showed that NNs implemented on CPUs can be fully reverse-engineered using merely side-channel leakages in a black box setting [15]. In [15], the activation function was discerned by the timing attack from its distinctive computation time with multiple executions and inputs. Chmielewski et al. reverse-engineered neural network designs, such as the number of layers and neurons, as well as the types of activation functions on GPUs through SEMA [26]. In [26], the activation function was discerned by the timing observed from a singular execution of SEMA. Takatoi et al. identified activation functions using SEMA [16]. In [16], the activation function was discerned by observing the timing and unique patterns from a singular execution of SEMA. In [15,16,26], the activation functions distinguished were the Sigmoid

function, Tanh function, softmax function, and ReLU function. In this work, we additionally distinguished similar activation functions, the Leaky ReLU function, SELU function, Swish function, and Fast Sigmoid function. These additional activation functions have not yet been distinguished to the best of our knowledge. In our work, up to three inputs and executions were necessary depending on the computation time of the activation function acquired.

As for the countermeasures, in [33], Maji et al. demonstrated a neural network accelerator that uses a threshold implementation to protect the model parameters and input from the power-based side-channel attacks.

7. Conclusions

In this work, we replicated the timing attack in the previous work by Batina et al. [15] and used it on various similar activation functions. The timing attack had a weakness towards various implementation methods of activation functions. Therefore, we improved an alternative method to recover similar activation functions using SEMA. We investigated the SEMA attack and its limits to distinguish activation functions from a black box neural network using side-channel information. The identification by SEMA was performed through observing the differences in peak patterns coming from the variability in the operations of the functions. We believe the operations and peaks of the traces have a relation, allowing SEMA to distinguish the activation functions, but the limit of SEMA is that a difference in values will be ignored. However, by having access to the inputs, we were able to identify all eight similar activation functions in our work. We showed that the potential of the SEMA attack is advantageous for the timing attack, as the operations of the activation functions can be depicted from the EM traces. This attack shows the vulnerabilities of edge AI to side-channel attacks and the need for neural networks on edge devices to implement effective countermeasures against side-channel attacks to strengthen the security of these embedded devices.

Finally, we have suggestions on countermeasures to mitigate these threats. The countermeasure listed in this article is to use activation functions with differences in only the values. There is also a method to use a memory-based implementation of the activation function, used often in side-channel countermeasures. We believe that the lookup table is a good way to implement activation functions. Although there is a trade-off of accuracy, we believe it to be an efficient way to counter both timing and SEMA attacks, as the lookup table can be constant-time and have no difference in operations.

Author Contributions: Conceptualization, G.T. and Y.L.; methodology, G.T. and Y.L.; software, G.T.; validation, G.T. and Y.L.; investigation, G.T.; resources, T.S., K.S. and Y.L.; data curation, G.T.; writing—original draft preparation, G.T.; writing—review and editing, T.S., K.S., Y.H.-A. and Y.L.; visualization, G.T.; supervision, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by JST AIP Acceleration Research Grant Number JPMJCR20U2 and JSPS KAKENHI Grant Number JP20H00590.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012, Lake Tahoe, NV, USA, 3–6 December 2012; Curran Associates, Inc.: New York, NY, USA, 2012; pp. 1106–1114.
2. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.

3. Tariq, Z.; Shah, S.K.; Lee, Y. Speech Emotion Detection using IoT based Deep Learning for Health Care. In Proceedings of the 2019 IEEE International Conference on Big Data (IEEE BigData), Los Angeles, CA, USA, 9–12 December 2019; pp. 4191–4196.
4. Kober, J.; Bagnell, J.A.; Peters, J. Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274. [CrossRef]
5. Teufl, P.; Payer, U.; Lackner, G. From NLP (Natural Language Processing) to MLP (Machine Language Processing). In Proceedings of the Computer Network Security, 5th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS 2010, St. Petersburg, Russia, 8–10 September 2010; Kotenko, I.V., Skormin, V.A., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6258, pp. 256–269.
6. Gilad-Bachrach, R.; Dowlin, N.; Laine, K.; Lauter, K.E.; Naehrig, M.; Wernsing, J. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York, NY, USA, 19–24 June 2016; Volume 48, pp. 201–210.
7. Fredrikson, M.; Lantz, E.; Jha, S.; Lin, S.M.; Page, D.; Ristenpart, T. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. In Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, 20–22 August 2014; Fu, K., Jung, J., Eds.; USENIX Association: Berkeley, CA, USA, 2014; pp. 17–32.
8. Xu, X.; Liu, C.; Feng, Q.; Yin, H.; Song, L.; Song, D. Neural Network-Based Graph Embedding for Cross-Platform Binary Code Similarity Detection. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, 30 October–3 November 2017; Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D., Eds.; ACM: New York, NY, USA, 2017; pp. 363–376.
9. Kucera, M.; Tsankov, P.; Gehr, T.; Guarnieri, M.; Vechev, M.T. Synthesis of Probabilistic Privacy Enforcement. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, 30 October–3 November 2017; Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D., Eds.; ACM: New York, NY, USA, 2017; pp. 391–408.
10. Riscure. Available online: https://www.riscure.com/ (accessed on 5 January 2022).
11. Yu, H.; Ma, H.; Yang, K.; Zhao, Y.; Jin, Y. DeepEM: Deep Neural Networks Model Recovery through EM Side-Channel Information Leakage. In Proceedings of the 2020 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2020, San Jose, CA, USA, 7–11 December 2020; pp. 209–218.
12. Wei, L.; Luo, B.; Li, Y.; Liu, Y.; Xu, Q. I Know What You See: Power Side-Channel Attack on Convolutional Neural Network Accelerators. In Proceedings of the 34th Annual Computer Security Applications Conference, ACSAC 2018, San Juan, PR, USA, 3–7 December 2018; ACM: New York, NY, USA, 2018; pp. 393–406.
13. Yan, M.; Fletcher, C.W.; Torrellas, J. Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures. In Proceedings of the 29th USENIX Security Symposium, Baltimore, MD, USA, 15–17 August 2018.
14. Hong, S.; Davinroy, M.; Kaya, Y.; Dachman-Soled, D.; Dumitras, T. How to 0wn the NAS in Your Spare Time. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
15. Batina, L.; Bhasin, S.; Jap, D.; Picek, S. CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In Proceedings of the 28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, 14–16 August 2019; Heninger, N.; Traynor, P., Eds.; USENIX Association: Berkeley, CA, USA, 2019; pp. 515–532.
16. Takatoi, G.; Sugawara, T.; Sakiyama, K.; Li, Y. Simple Electromagnetic Analysis Against Activation Functions of Deep Neural Networks. In Proceedings of the International Conference on Applied Cryptography and Network Security, Rome, Italy, 19–22 October 2020; pp. 181–197.
17. Klambauer, G.; Unterthiner, T.; Mayr, A.; Hochreiter, S. Self-Normalizing Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 971–980.
18. Ramachandran, P.; Zoph, B.; Le, Q.V. Searching for Activation Functions. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
19. Timmons, N.G.; Rice, A. Approximating Activation Functions. *arXiv* **2020**, arXiv:2001.06370.
20. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In Proceedings of the International Conference on Machine Learning (ICML) Workshop on Deep Learning for Audio, Speech and Language Processing, Atlanta, GA, USA, 16 June 2013.
21. Kocher, P.C.; Jaffe, J.; Jun, B. Differential Power Analysis. In Proceedings of the Advances in Cryptology—CRYPTO'99, 19th Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 1999; Wiener, M.J., Ed.; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1666, pp. 388–397.
22. Kocher, P.C. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Proceedings of the Advances in Cryptology—CRYPTO'96, 16th Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 1996; Koblitz, N., Ed.; Springer: Berlin/Heidelberg, Germany, 1996; Volume 1109, pp. 104–113.
23. Tinkercad. Available online: https://www.tinkercad.com/ (accessed on 5 January 2022).
24. Patranabis, S.; Mukhopadhyay, D. *Fault Tolerant Architectures for Cryptography and Hardware Security*; Springer: Berlin/Heidelberg, Germany, 2018.
25. Luo, C.; Fei, Y.; Luo, P.; Mukherjee, S.; Kaeli, D.R. Side-channel power analysis of a GPU AES implementation. In Proceedings of the 33rd IEEE International Conference on Computer Design, ICCD 2015, New York, NY, USA, 18–21 October 2015; pp. 281–288.

26. Chmielewski, L.; Weissbart, L. On Reverse Engineering Neural Network Implementation on GPU. In Proceedings of the Applied Cryptography and Network Security Workshops—ACNS 2021 Satellite Workshops, AIBlock, AIHWS, AIoTS, CIMSS, Cloud S&P, SCI, SecMT, and SiMLA, Kamakura, Japan, 21–24 June 2021; Zhou, J., Ahmed, C.M., Batina, L., Chattopadhyay, S., Gadyatskaya, O., Jin, C., Lin, J., Losiouk, E., Luo, B., Majumdar, S., et al., Eds.; Springer: Amsterdam, The Netherlands, 2021; Volume 12809, pp. 96–113.
27. Dubey, A.; Cammarota, R.; Aysu, A. MaskedNet: The First Hardware Inference Engine Aiming Power Side-Channel Protection. In Proceedings of the 2020 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2020, San Jose, CA, USA, 7–11 December 2020; pp. 197–208.
28. Regazzoni, F.; Bhasin, S.; Alipour, A.; Alshaer, I.; Aydin, F.; Aysu, A.; Beroulle, V.; Natale, G.D.; Franzon, P.D.; Hély, D.; et al. Machine Learning and Hardware security: Challenges and Opportunities Invited Talk. In Proceedings of the IEEE/ACM International Conference on Computer Aided Design, ICCAD 2020, San Diego, CA, USA, 2–5 November 2020; pp. 141:1–141:6.
29. Hu, X.; Liang, L.; Li, S.; Deng, L.; Zuo, P.; Ji, Y.; Xie, X.; Ding, Y.; Liu, C.; Sherwood, T.; et al. DeepSniffer: A DNN Model Extraction Framework Based on Learning Architectural Hints. In Proceedings of the ASPLOS'20: Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, 16–20 March 2020; ACM: New York, NY, USA, 2020; pp. 385–399.
30. Duddu, V.; Samanta, D.; Rao, D.V.; Balas, V.E. Stealing Neural Networks via Timing Side Channels. *arXiv* **2018**, arXiv:1812.11720.
31. Maji, S.; Banerjee, U.; Chandrakasan, A.P. Leaky Nets: Recovering Embedded Neural Network Models and Inputs Through Simple Power and Timing Side-Channels—Attacks and Defenses. *IEEE Internet Things J.* **2021**, *8*, 12079–12092. [CrossRef]
32. Wang, H.; Hafiz, S.M.; Patwari, K.; Chuah, C.N.; Shafiq, Z.; Homayoun, H. Stealthy Inference Attack on DNN via Cache-based Side-Channel Attacks. Available online: https://web.cs.ucdavis.edu/~zubair/files/dnn-sc-date2022.pdf (accessed on 15 April 2022).
33. Maji, S.; Banerjee, U.; Fuller, S.H.; Chandrakasan, A.P. A ThreshoId-ImpIementation-Based Neural-Network Accelerator Securing Model Parameters and Inputs Against Power Side-Channel Attacks. In Proceedings of the 2022 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, 20–26 February 2022; Volume 65, pp. 518–520.