IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 August 2023, accepted 30 August 2023, date of publication 1 September 2023, date of current version 7 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3311370

RESEARCH ARTICLE

Autoscaled-Wavelet Convolutional Layer for Deep Learning-Based Side-Channel Analysis

DAEHYEON BAE[1], DONGJUN PARK[1], GYUSANG KIM[1], MINSIG CHOI[1], NAYEON LEE[1], HEESEOK KIM[2], (Member, IEEE), AND SEOKHIE HONG[1], (Member, IEEE)

[1]School of Cybersecurity, Korea University, Seoul 02841, South Korea
[2]Department of AI Cyber Security, College of Science and Technology, Korea University, Sejong 30019, South Korea

Corresponding authors: Seokhie Hong (shhong@korea.ac.kr) and Heeseok Kim (80khs@korea.ac.kr)

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea Government (Ministry of Science and ICT) (No.2021-0-00903, Development of Physical Channel Vulnerability-based Attacks and its Countermeasures for Reliable On-Device Deep Learning Accelerator Design).

ABSTRACT Continuous Wavelet Transform (CWT) is rarely used in the field of side-channel analysis due to problems related to parameter (wavelet scale) selection; There is no way to find the optimal wavelet scale other than an exhaustive search, and the resulting spectrogram analysis can introduce significant analysis complexity. However, a well-scaled CWT can improve the signal-to-noise ratio of side-channel signals, which can lead to better attack performance. And our insights suggest that there is scope for CWT and deep learning approaches to be combined, which could help the models to train more effectively while overcoming the problems of CWT. In this context, we propose a novel feature extraction layer that combines a CWT with a Convolutional Neural Network (CNN). The proposed method can leverage neural network training to automatically adjust a wavelet scale, which is a critical parameter of CWT. Furthermore, the proposed method can lead to performance improvements by enabling a deep learning model to perform on-the-fly multi-frequency analysis without any pre-processing. By bringing the two approaches together, we were able to overcome the limitations of CWT and improve the performance of deep learning-based side-channel analysis. As an experimental result using open dataset ASCAD, a de facto standard in deep learning-based side-channel analysis, we confirmed that the proposed method could improve the performance by inserting the proposed layer into existing state-of-the-art deep learning models.

INDEX TERMS Convolutional neural network, deep learning, hardware security, side-channel analysis, wavelet transform.

I. INTRODUCTION

Side-channel analysis remains the most practical attack targeting existing cryptographic systems even after more than 20 years since the development of the timing attack [1] and DPA (Differential Power Analysis) [2]. Side-channel analysis is a type of cryptanalysis that extracts sensitive information inside an integrated circuit from unintentional information leakage caused by physical characteristics. These unintentional information leakages, i.e., side-channel leakages, include time, power consumption, and electromagnetic emission. Once a cryptographic algorithm is operated in an integrated circuit, various side-channel information related to sensitive value (e.g., a cryptographic key) could be leaked. Therefore, hardware (implementation) level security of a cryptographic algorithm cannot be guaranteed, even if the algorithm or scheme level security has been mathematically proven unless side-channel analysis-specific countermeasures are investigated. For this reason, side-channel analysis is still actively researched in cryptanalysis/hardware security academia and the semiconductor industry [3].

To improve the performance of side-channel analysis, various signal processing methods have been studied: signal aligning, filtering, and transforming [4], [5]. One example of this is wavelet transform [6]. The wavelet transform, which is the main focus of this paper, can compress a

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamed Elhoseny.

signal or decompose frequency by performing convolution between a short waveform called a wavelet and an unrefined signal. Discrete Wavelet Transform (DWT) is often used in side-channel analysis to take advantage of compressing side-channel signals while preserving Signal-to-Noise Ratio (SNR) [7]. Whereas Continuous Wavelet Transform (CWT), which decomposes signals for time-frequency analysis, has been rarely used in the field of side-channel analysis due to scale selection and data size-increasing problems [8]. That is, there is no effective algorithm for calculating the optimal wavelet scale, which is a critical parameter of CWT, to improve the performance of side-channel analysis due to many factors: hardware and implementation characteristics (e.g., instruction structure), signal quality, and the presence of countermeasures; exhaustive search is required. In addition, the size of the data increases in proportion to the number of selected wavelet scales, which may cause an increase in analysis complexity. However, we have found that CWT provides significant performance gains over DWT if appropriate wavelet scales are selected. We aim to overcome the wavelet scale selection problem in CWT by combining it with deep learning approaches. In addition, we leverage the advantages of CWT to improve the performance of existing deep learning-based side-channel analysis.

Deep learning is a technique for approximating functions using a deep neural network. Over the past decade, deep learning has been applied to side-channel analysis; Deep learning-based side-channel analysis. Deep learning-based side-channel analysis is used to approximate a function that maps from a time series side-channel leakage (e.g., power consumption or electromagnetic emission) to an intermediate value (IV) of a cryptographic algorithm. To approximate a function more accurately, a deep learning model tunes itself using error backpropagation, i.e. training. We aim to combine CWT primitives with models for deep learning-based side-channel analysis. This allows the wavelet scales (critical parameters of CWT) to be automatically adjusted by itself through error backpropagation during the training process. In addition, the effects of multi-frequency analysis driven by the CWT primitives can be used to improve the attack performance of the model.

In this paper, we bring together two worlds that have been studied independently to overcome the shortcomings of CWT while improving the performance of deep learning-based side-channel analysis: CWT from the traditional signal-processing world with CNN from the deep-learning world. In this context, we propose a novel feature extraction layer called the *Autoscaled-Wavelet Convolutional Layer (ASW-CL)* for deep learning models that takes advantage of the wavelet transform (CWT) and the neural network training. We were motivated by the similarity of the underlying operations of CWT and one-dimensional CNN. The proposed ASW-CL enables a deep learning model to conduct on-the-fly multi-frequency analysis in a single model without any signal pre-processing. Furthermore, it can automatically adjust wavelet scale by leveraging neural network training. Here, the wavelet scale is a critical parameter of the CWT, but there is no effective algorithm to find its optimal value. As a result, we could mitigate the inefficiency and inaccuracy of CWT as well as improve the performance of the deep learning model. We experimentally evaluate the proposed method using ASCAD [9] datasets, a de facto standard for deep learning side-channel analysis. And we confirmed that performance can be improved by simply inserting ASW-CL into existing state-of-the-art CNN models.

Contributions. The following is the summary of the major contributions of this paper:

- To overcome the limitations of CWT and utilize its advantages, we combined CWT with deep learning techniques. In this context, we propose an ASW-CL, a novel feature extraction layer for deep learning-based side-channel analysis. The proposed method enables a deep learning model (CNN) to perform on-the-fly multi-frequency analysis in a single model with automatically adjusted parameters (wavelet scales) of CWT.
- We investigated the power of CWT, which has rarely been used in the field of side-channel analysis because of several limitations related to parameter (wavelet scale) selection. As a result, we show that the CWT with appropriately selected parameters can outperform the DWT, which is often adopted for processing side-channel channel signals.
- We reveal problems related to the reliability of Nt_{ge}, a commonly used evaluation metric in deep learning-based side-channel analysis. And we propose a solution to mitigate the problems.

Organization. The remainder of this paper is organized as follows. Section II presents a preliminary background. Section III introduces related works, including studies similar to ours and deep learning-based side-channel analysis. Section IV proposes a novel ASW-CL for the deep learning-based side-channel analysis. Section V evaluates our ASW-CL using an open dataset ASCAD and discusses open problems. Finally, Section VI concludes this paper.

II. PRELIMINARIES

A. SIDE-CHANNEL ANALYSIS

Side-channel analysis is a type of attack that reveals internal information by analyzing signals unintentionally leaked from hardware. Here, internal information includes not only data values such as cryptographic keys but also the instructions of a processor. For this reason, it is possible to recover the instructions operating in the microprocessor by analyzing the side-channel signal [8]. To exploit data-dependent components of side-channel leakage, an adversary should define a leakage model (power consumption model). Because of the limited information about target devices, the adversary can only define a coarse-grained leakage model, which allows the leveraging of relative differences in power consumption according to values. There are two leakage models

that reflect hardware characteristics: Hamming distance and Hamming weight models. The Hamming distance model is based on the fact that dynamic power is consumed when a bit flip occurs due to the characteristics of the CMOS (Complementary Metal–Oxide–Semiconductor) circuit, which is adopted by modern semiconductors. For given pre-state value x and post-state value y, Hamming distance function $\mathrm{HD}(x, y)$ is as described in (1) where x_i denotes i^{th} bit of $x = (x_{n-1}x_{n-2}\ldots x_0)_2$, and the same is true for y_i.

$$\mathrm{HD}(x, y) = \sum_{i=0}^{n-1} x_i \oplus y_i \tag{1}$$

whereas, the Hamming weight model characterizes the leakage caused by precharge of internal buses [10]. For given state x, Hamming weight function $\mathrm{HW}(x)$ is as follows:

$$\mathrm{HW}(x) = \sum_{i=0}^{n-1} x_i. \tag{2}$$

In terms of attack scenario and ability of an adversary, side-channel analysis is divided into two types: non-profiled and profiled side-channel analysis. In the case of non-profiled side-channel analysis, the adversary can only access a target device. And they can acquire side-channel leakages and additional information (e.g., plaintext or ciphertext) related to cryptographic operations. Then, they recover a cryptographic key by analyzing the side-channel leakage using statistical methods. This type of analysis includes SPA (Simple Power Analysis) [11], DPA [2], CPA (Correlation Power Analysis) [12], and DDLA (Differential Deep Learning Analysis) [13].

In the case of the profiled side-channel analysis, an adversary analyzes a target device using a profile generated from a profiling device. Here, the profiling device is an open copy of the target device used to characterize the leakage in advance. To generate a leakage profile, the adversary needs to know the inputs (key, plain/ciphertext) of the profiling device to calculate intermediate values of a cryptographic algorithm. This type of analysis includes template attack [14] and machine learning-based side-channel analysis [15].

B. WAVELET TRANSFORM

Wavelet transform is one of the signal processing methods for compressing or decomposing signals using a wavelet [16]. Here, the wavelet is a waveform that oscillates briefly around zero and must satisfy the following (3) where $\psi(t)$ denotes a mother wavelet function over time t.

$$\int_{-\infty}^{\infty} \psi(t)dt = 0 \tag{3}$$

Well-known wavelets include Ricker (Mexican hat), Gaussian, Daubechies, and Morlet wavelets. This paper proposes and uses a *wavelet kernel*, which is a one-dimensional CNN kernel in the shape of a wavelet. And we adopt the Ricker wavelets in all experiments of this paper. This is because the Ricker wavelet showed the largest SNR improvement over the other wavelets (see Section V-E). The Ricker wavelet ψ_{Ricker} over time t is as described in (4), where s is a scale that determines the frequency to be decomposed. The wavelet is stretched or shrunk along the t-axis depending on the wavelet scale s.

$$\psi_{\mathrm{Ricker}}(t) = \frac{2}{\sqrt{3s}\pi^{\frac{1}{4}}}\left(1 - \left(\frac{t}{s}\right)^2\right)e^{-\frac{1}{2}\left(\frac{t}{s}\right)^2} \tag{4}$$

The wavelet transform F of a signal x using a wavelet ψ is described in (5). Here, s, τ, and $x(t)$ are denotes a wavelet scale, a time shift factor, and a target signal over t, respectively. And, converting (5) to a discrete operation is described in (6).

$$F_{\psi}^{x}(\tau) = \frac{1}{\sqrt{|s|}}\int_{-\infty}^{\infty} x(t)\psi(t-\tau)dt \tag{5}$$

$$F_{\psi}^{x}(\tau) = \frac{1}{\sqrt{|s|}}\sum_{t} x(t)\psi(t-\tau) \tag{6}$$

We can describe (6) as (7) using the convolution operator $(*)$. That is, if ψ is a kernel and there is no multiplied constant term $(\frac{1}{\sqrt{|s|}})$ in (7), it is the same as the convolution of the one-dimensional convolutional neural network (CNN). Therefore, we replaced the kernel of 1D-CNN with a wavelet to indirectly perform the wavelet transform in a single model (see Section IV).

$$F_{\psi}^{x}(\tau) = \frac{1}{\sqrt{|s|}}(x * \psi)(\tau) \tag{7}$$

The relation between the wavelet scale s and an extracted frequency f_s is as described in (8) where f_c and Δ denote the central frequency of the mother wavelet and sampling period, respectively. Here, the central frequency of the Ricker wavelet is 0.25(Hz). Our approach is to automatically determine an optimal scale s, which can improve guessing (classification) performance, using neural network training.

$$f_s = \frac{f_c}{\Delta \times s} \tag{8}$$

C. ADVANTAGES OF WAVELET TRANSFORM IN SIDE-CHANNEL ANALYSIS

There are several studies where two types of wavelet transform have already been investigated in side-channel analysis: DWT and CWT.

The DWT compresses a signal by repeatedly performing the same process, and the number of repetitions denotes the level. In each level of DWT, an input signal passes through low-pass and high-pass filters, which operate by wavelet transform, respectively. Then, the length of the signal passing through the filters is downsampled by half. The next level uses the signal that passed through the low-pass filter of the previous level. Here, we do not need to specify a scale factor since frequencies corresponding to power-of-two scales are to be filtered. Therefore, DWT is used to reduce space or analysis complexity rather than performance improvement in side-channel analysis.

On the other hand, CWT is used to generate a spectrogram composed of time and frequency axes for given scales,

i.e., decompose the frequency. Therefore, we can extract multi-frequency information using CWT by selecting the wavelet scales. Here, data size increases in proportion to the number of selected scales (data size is multiplied by the number of scales). And there is no deterministic algorithm for finding an optimal scale to improve side-channel analysis performance. For this reason, CWT is rarely used in side-channel analysis unless it is used with exhaustive search and feature extraction algorithms such as side-channel-based disassembler implementation [17]. In another case, CWT is sometimes used to convert a 1-dimensional time-axis signal into 2-dimensional time-frequency data in order to utilize 2D-CNN [18]. As described above, the CWT is not even studied due to disadvantages (i.e., scale selection problem and data size increasing problem) in the field of side-channel analysis.

To show the advantages of CWT, we compare SNR using an open dataset ASCAD [9] (see Section V-A). To calculate SNR, we use 50k profiling traces of ASCAD (fixed key, v1). The target algorithm is first-order masked (sbox recomputation-based) AES-128 implemented in assembly language, and the SNR is calculated for the following four intermediate values in the same as [9] (except useless SNR1 of [9]).

- SNR2: HW($sbox(p[3] \oplus k[e]) \oplus r_{out}$)
- SNR3: HW(r_{out})
- SNR4: HW($sbox(p[3] \oplus k[e]) \oplus r[3]$)
- SNR5: HW($r[3]$)

Here, r_{out} and $r[3]$ are the output mask of the recomputed-sbox and linear mask of the AES state, respectively. We show the SNR calculated from the raw traces and the CWT coefficients for the previous four values in Figure 1 (jitter-free) and Figure 2 (jittery). To apply CWT, we divided the range from 1.6MHz to 32MHz into 100 equal parts and used the corresponding wavelet scales (see eq. (8)), i.e., data size increases 100 times. We confirmed that the SNR can be higher than raw traces at specific frequencies. This is the same for the jitter-free version (desync0) as well as the jitter-added versions (desync50, desync100). The following Table 1 summarizes the SNR values for the raw traces, CWT coefficients, and the DWT results. The result indicates that the CWT can significantly contribute to performance improvements over the DWT, which are often used for side-channel analysis. This paper proposes the feature extraction layer ASW-CL to take advantage of CWT while mitigating disadvantages using neural networks (see Section IV). Meanwhile, if the CWT is used only for signal processing (not with deep learning), the Least-Squares Wavelet Analysis (LSWA), which has better time-frequency resolution, can be considered [19].

TABLE 1. Comparison of maximum SNR according to jitter and wavelet transform methods for four intermediate values (SNR2-SNR5). The SNR is the highest when CWT is applied, which is significantly higher than DWT.

Dataset	WT	Maximum Signal-to-Noise Ratio			
		SNR2	SNR3	SNR4	SNR5
$ASCAD^{fixed}_{desync0}$	Raw	2.45	0.78	9.87	2.59
	DWT	2.45	0.81	10.8	2.98
	CWT	**3.31**	**0.89**	**12.1**	**3.20**
$ASCAD^{fixed}_{desync50}$	Raw	0.0029	0.0028	0.0048	0.0054
	DWT	0.026	0.058	0.038	0.40
	CWT	**0.66**	**0.35**	**1.34**	**1.49**
$ASCAD^{fixed}_{desync100}$	Raw	0.0032	0.0028	0.0037	0.0032
	DWT	0.032	0.049	0.068	0.34
	CWT	**0.20**	**0.16**	**0.60**	**0.82**

D. DEEP LEARNING TECHNIQUES

Deep learning is a kind of machine learning that trains and infers data using neural networks for function approximation. Various deep-learning models have been investigated for side-channel analysis. Among them, we introduce MLP, the most basic neural network, and CNN, which was adopted by numerous state-of-the-art works [20].

1) MULTI-LAYER PERCEPTRON (MLP)

Multi-Layer Perceptron is a well-known example of a feed-forward neural network that can approximate functions. The MLP consists of an input layer, hidden layers, and an output layer. According to the universal approximation theorem, a neural network composed of one or more hidden layers with an activation function that satisfies specific conditions can approximate any function [21]. Each layer consists of perceptrons that accumulate input values and apply a non-linear activation function. Well-known non-linear activation functions include ReLU, Sigmoid, and tanh functions. The layers are fully connected by weights, which are important parameters for approximating functions. The structure of MLP ($\hat{f}_{MLP}$) is as described in (9) where s, n, σ, λ, and $\circ$ denote the softmax function, the number of layers, an activation function, the fully connected layer, and function composition, respectively. And I denote an input layer, which is the identity function. Here, the fully connected (λ) denotes matrix multiplication between input and weights.

$$\hat{f}_{MLP} = s \circ \sigma_n \circ \lambda_n \circ \cdots \circ \sigma_1 \circ \lambda_1 \circ I \tag{9}$$

2) CONVOLUTIONAL NEURAL NETWORK (CNN)

Convolutional Neural Network refers to a deep learning model that combines convolutional layers for feature extraction and a multi-layer perceptron (fully-connected layer) for classification [22]. The convolutional layer extracts features (patterns) by performing the convolution between input data and kernels. The Dimension of the kernel determines the overall structure of the CNN model. That is, CNN with one-dimensional kernels can handle time-series data, whereas CNN with two-dimensional kernels can handle image data. All values of the CNN kernels are trainable parameters of the model and are automatically adjusted to extract features that can improve the classification performance during the neural network training phase. The features extracted through the convolutional layer are downsampled to the max./min./avg. values for the specific window size in the pooling layer. The output values of the pooling layer do not change significantly

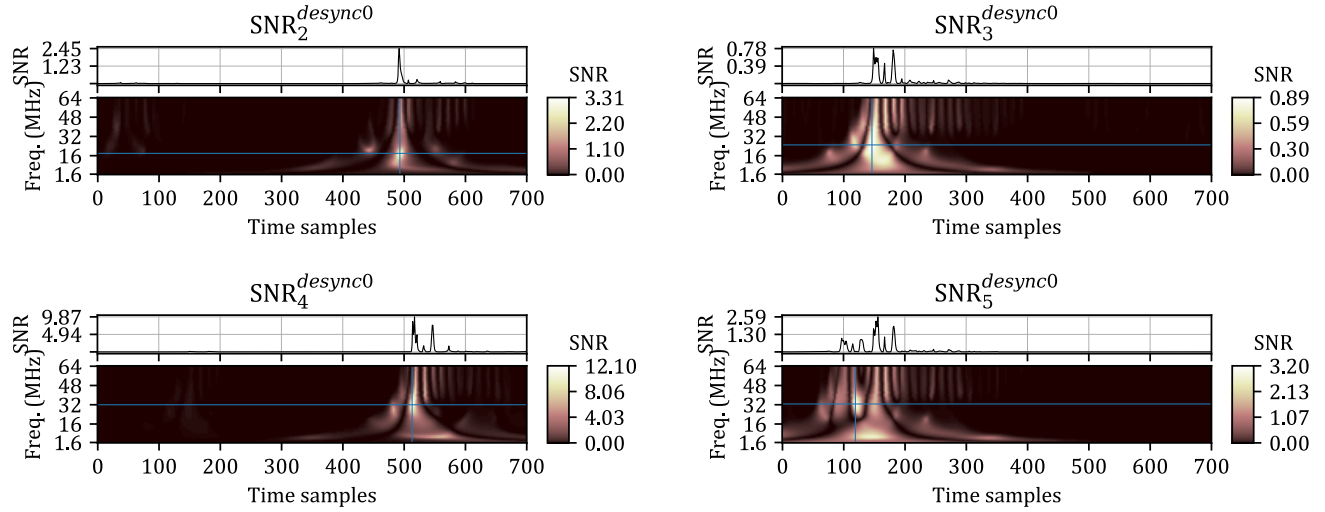


FIGURE 1. The SNR values of raw traces and CWT coefficients for $ASCAD^{fixed}_{desync0}$. The CWT coefficient shows high SNR over a wide frequency band.

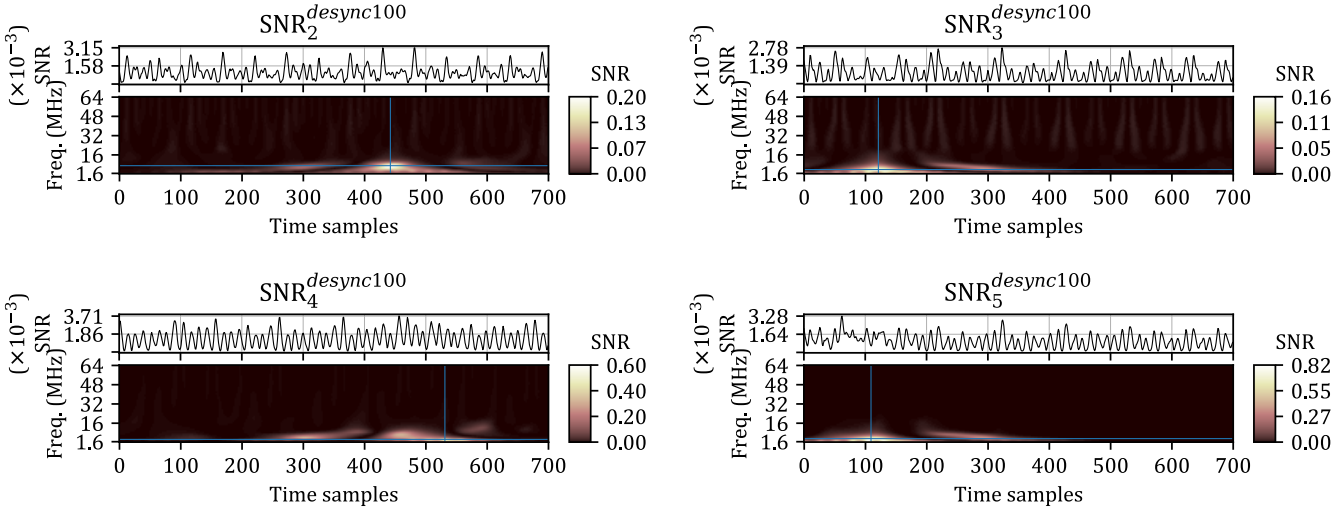


FIGURE 2. The SNR values of raw traces and CWT coefficients for $ASCAD^{fixed}_{desync100}$. The CWT coefficient shows high SNR in a narrow frequency band.

even if there is a change in the position of the input features. This property is called translation invariance, which is why CNN can defeat jitter-based hiding countermeasures. The structure of CNN ($\hat{f}_{CNN}$) is as described in (10) where n_{fc}, n_{conv}, δ, and γ denote the number of fully-connected layers, the number of convolutional layers, a pooling layer, and a convolutional layer, respectively.

$$\hat{f}_{CNN} = s \circ [\sigma \circ \lambda]^{n_{fc}} \circ [\delta \circ \sigma \circ \gamma]^{n_{conv}} \circ I \tag{10}$$

3) TRAINING NEURAL NETWORK

The training of a neural network is to reduce the error of approximating a function by adjusting weights. Here, the gradient descent method is used to update the weights (for all trainable parameters). The training (adjusting) of a weight w using the gradient descent method is as described in (11) where L, α, w', and ∂ denote a loss function of $\hat{f}$, learning rate, an adjusted weight, and the partial derivative, respectively. The loss function calculates the error between f and $\hat{f}$, and the learning rate determines the degree of adjusting.

$$w' \leftarrow w - \alpha \cdot \frac{\partial L}{\partial w} \tag{11}$$

here, the chain rule is used to calculate the gradient ($\frac{\partial L}{\partial w}$) for deep neural networks. For example, the gradient calculation for adjusting a weight of the m^{th}-layer of (9) is as described in (12).

$$\frac{\partial L}{\partial w_m} = \frac{\partial L}{\partial \sigma_n} \cdot \frac{\partial \sigma_n}{\partial \lambda_n} \cdots \frac{\partial \lambda_m}{\partial w_m} \tag{12}$$

In the case of CNN, kernel values are also adjusted as same as the weights, to extract features that can reduce the output error of the fully-connected layer.

TABLE 2. A comparison of related works (WCNN, ML-CNN, wCwNN, and HDW-CNN).

Model	Type of WT†	Wavelet scale	Input data	Application	Where the WT is performed
WCNN [32]	DWT	Fixed	Image	Texture Classification	Across the convolutional layer (Insertion)
ML-CNN [33]	DWT, IWT‡	Fixed	Image	Image Classification	Across the convolutional layer (Replacement)
wCwNN [34]	DWT	Fixed	Image	Image Classification	In activation functions
HDW-CNN [35]	DWT	Fixed	Time-series	Fault Diagnosis	In front of the model (CNN)
Proposed (ASW-CL)	**CWT**	**Autoscaled**	Time-series	**Side-Channel Analysis**	**In 1D-conv. operations (using wavelet kernel)**

†: Wavelet Transform, ‡: Inverse Wavelet Transform

4) APPLICATION TO THE SIDE-CHANNEL ANALYSIS

Deep learning-based side-channel analysis aims to approximate a function that maps from a time series side-channel leakage to an IV of a cryptographic algorithm using deep learning models. It is attractive enough because an adversary can leave out the preparation steps, such as signal preprocessing and PoI (Point of Interest) selection [23]. Furthermore, it can defeat secret sharing [24] and hiding [25], [26] countermeasures without any obvious preprocessing, such as leakage combination or signal alignment [27], [28]. Thanks to these superiorities, deep learning-based side-channel analysis has become one of the mainstream in the side-channel analysis community and academia. Numerous papers have already shown that deep learning models, especially CNN, can defeat secret sharing and hiding countermeasures and outperform classical side-channel analysis methods [28], [29], [30], [31].

For deep learning-based side-channel analysis, a deep learning model has input nodes equal to the length of a time-series side-channel signal and output nodes equal to the intermediate value (to be recovered) space of a cryptographic algorithm. That is, the deep learning model cannot directly recover a key like other profiling attacks, but can recover an intermediate value of the cryptographic algorithm based on a leakage model (e.g., Hamming weight or Hamming distance). Therefore, additional information, such as plaintext or ciphertext corresponding to each trace, is required to recover a cryptographic key. Deep learning-based side-channel analysis was initially applied only to profiling scenarios, but B. Timon et al. applied it to a non-profiling scenario in 2019 for the first time [13].

In the profiling scenario, an adversary must train a deep learning model to predict an intermediate value of a cryptographic algorithm using a profiling device in advance. Here, the adversary only needs to know the targeted intermediate value regardless of the constraints of the profiling device. Then, the adversary sends numerous trace queries (acquired from target devices with a fixed key) to the trained model. Finally, the adversary can guess a key using maximum log-likelihood estimation [27].

III. RELATED WORKS

A. DEEP LEARNING MODELS WITH WAVELET

Related works have already been conducted to insert wavelet transforms into deep learning models. In 2018, Fujieda et al. proposed a Wavelet-CNN (WCNN) to perform multi-frequency analysis with a single deep learning model [32]. The WCNN extracts features by performing convolutional and pooling operations in the same way as classical CNN but has the difference that high and low-frequency signals decomposed by DWT are fed into midways of feature extraction layers using channel-wise concatenation. In 2019, Liu et al. proposed a Multi-Layer WCNN (ML-WCNN) to achieve a better trade-off between receptive field size and computational efficiency [33]. The ML-WCNN uses DWT to create subnetworks and IWT (Inverse Wavelet Transform) to reconstruct them, making it effective for denoising, image super-resolution, and more. In 2021, Liu et al. proposed a wavelet Convolutional wavelet Neural Network (wCwNN) to improve the performance of image classification [34]. The wCwNN replaces the activation functions of the convolutional layers and the fully-connected layers with wavelet functions. In 2023, Paraskevopoulos et al. proposed a Hybrid Discrete Wavelet-CNN (HDW-CNN) for fault diagnosis [35]. The HDW-CNN statically decomposes the signals using DWT at the front of a conventional CNN. We show a comparison of related works in Table 2.

The purpose of WCNN and its variants is to perform multi-frequency analysis with a single deep learning model, similar to our work. However, most have a different structure or purpose and are mostly based on DWT. The main contributions of ASW-CL over WCNN and its variants are as follows:

- Unlike DWT, which operates as low-pass and high-pass filters, ASW-CL adopts a CWT-like method to decompose signals in specific frequency bands according to given wavelet scales.
- The wavelet scales, which are critical parameters that determine the frequency of signals to be decomposed, are *automatically* adjusted in the neural network training phase; the wavelet scale is a trainable parameter of a deep learning model in the proposed method.

B. DEEP LEARNING-BASED SIDE-CHANNEL ANALYSIS

For the first time, in 2013, Z. Martinasek et al. attempted side-channel analysis using an MLP consisting of three layers: an input layer, a hidden layer, and an output layer. However, the concept of deep learning was not established at this time and it was simply used as a machine learning algorithm, perceptron [36]. Therefore, some papers do not classify this paper as a deep learning-based side-channel analysis [20]. In 2016, Maghrebi et al. showed that the masking (sbox recomputation) countermeasure could be defeated using deep learning models, i.e., MLP and CNN, without leakage

combination and knowledge about mask values [27]. After that, deep learning-based side-channel analysis has been actively investigated. According to [20], more than 183 papers were published during 2016-2022. Among them, most state-of-the-art works are based on the CNN model [37]. The CNN can show outstanding performance without signal preprocessing even for unsynchronized and noisy signals owing to characteristics of convolutional and pooling layers. Therefore, the CNN model is capable of defeating both masking and hiding countermeasures by itself.

More recently, non-CNN models have been applied to side-channel analysis. In 2021, Rijsdijk et al. performed hyperparameter tuning using reinforcement learning [30], and in 2022, Chang et al. adopt the RNN and LSTM [38] for side-channel analysis. In 2022, Lei et al. showed that an effective profiling attack can be performed on high-dimensional data with assist of an autoencoder [39]. Also in 2022, Knežević et al. proposed an activation function for side-channel analysis rather than a specific model [40]. However, the previous works on deep learning-based side-channel analysis are not very relevant to our work as we propose a new feature extraction layer that is a modification of CNN primitive. Therefore, in this paper, we conduct experiments by applying our ASW-CL to the genuine CNN-based models in [9] and [37], which are the most state-of-the-art to date.

IV. ASW-CL: AUTOSCALED-WAVELET CONVOLUTIONAL LAYER

As shown in Section II-C, CWT can improve the performance of side-channel analysis by increasing the SNR of signals if optimal wavelet scales are selected. Here, one might think that, just like we did when calculating the SNR of Figure 1, anyone could determine the optimal scale by performing an exhaustive search of specific a frequency range. But, considering the attack scenarios, it is usually unfeasible. That is, it is difficult to acquire the random numbers used in masking countermeasures even if an adversary can access the profiling device and can calculate the intermediate value (unmasked) of the cryptographic algorithm for profile generation unless an open dataset. Therefore, the adversary has no choice but to calculate the SNR using only unmasked values; this is useless because there is no first-order leakage. In the case of a non-profiling scenario, it is also unfeasible because the SNR cannot be calculated unless the key is known. The adversary can also analyze entire decomposed data, which can lead to a huge increase in analysis complexity. Furthermore, the frequencies we select at uniform intervals may not contain the optimal frequencies to improve side-channel analysis performance. It can occur much more frequently for jittery traces. Therefore, the adversary has to select frequencies more tightly, which also increases the complexity. To overcome these limitations and take advantage of CWT, we propose a novel feature extraction layer '*Autoscaled-Wavelet Convolutional Layer (ASW-CL)*'.

The proposed ASW-CL has a '*wavelet kernel*', a modified version of the classical one-dimensional kernel of the CNN. And the scale parameter of the *wavelet kernel* is designed to be trainable, which is automatically determined during the neural network training. As a result, an adversary does not have to consider scale parameters and can perform one-the-fly multi-frequency analysis with a single model to improve side-channel analysis performance. The main difference between the *wavelet kernel* and the classical kernel is where the trainable part is, as shown in Figure 3. When we only pay attention to the kernel value, our ASW-CL is not trainable, whereas all values of the classical kernel are trainable. However, the wavelet kernel is generated from a wavelet function (e.g., Ricker wavelet) and a wavelet scale. Here, the wavelet scale is a trainable parameter. Then, the scales are adjusted automatically to extract features that can improve the key recovery performance in the training process.

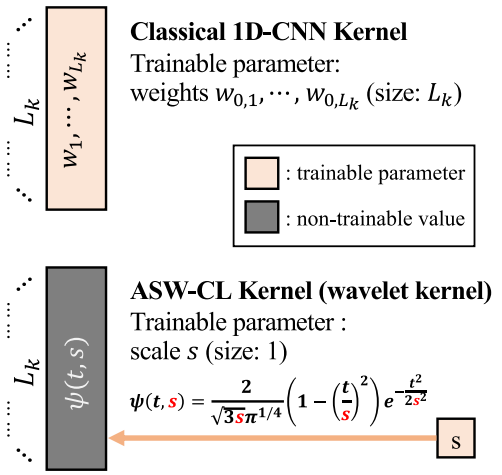


FIGURE 3. A structure of wavelet kernel and classical kernel.

Training and updating the wavelet scales can be done in the same context as updating the kernel values in a classical 1D-CNN. Suppose the first convolutional layer in the previous (10) is ASW-CL. Then, the wavelet scale s can affect the output of $\hat{f}$; the $\hat{f}$ is a function composed of s. Therefore, the scale s can be adjusted as described in (13) where L is the loss function of $\hat{f}$.

$$s' \leftarrow s - \alpha \cdot \frac{\partial L}{\partial s} \tag{13}$$

here, the gradient of s ($\frac{\partial L}{\partial s}$) can be calculated by (14) in the same context as updating the kernel values in the classical CNN where ψ is the wavelet generation function such as previous (4).

$$\frac{\partial L}{\partial s} = \frac{\partial L}{\partial \sigma_n} \cdot \frac{\partial \sigma_n}{\partial \lambda_n} \cdots \frac{\partial \lambda_1}{\partial \sigma_m} \cdot \frac{\partial \sigma_m}{\partial \gamma_m} \cdots \frac{\partial \gamma_1}{\partial \psi} \cdot \frac{\partial \psi}{\partial s} \tag{14}$$

The following Algorithm 1 is the initialization procedure for the proposed ASW-CL. It is executed only once when the model is created. In the initialization phase, the initial wavelet scales are determined and converted into trainable parameters. The initial wavelet scale is determined by a random

real number within a specific range. To calculate the specific range, the center frequency of the wavelet (f_c), the given maximum frequency to be extracted (f_{max}), the sampling rate (SR), the length of the kernel (L_k), and the type of wavelet are used. Here, the maximum frequency can be defined by an attacker. In general, f_{max} should be a (low) integer multiple of the clock because there is little meaningful information at frequencies much higher than the operating clock of DUT (Device Under Test). For all experiments in this paper, we set f_{max} to a 5×DUT clock. Then, we calculate the minimum wavelet scale as shown in line 2 of Algorithm 1. On the other hand, the maximum wavelet scale should be determined by the length of the kernel rather than the frequency aspect. This is because only a small portion of a wavelet may be reflected in the kernel if the scale is too large. Considering the shape of the Ricker wavelet, we used $L_k/10$ as the maximum scale. And we used the *autograd* module in PyTorch to convert the scale into the trainable parameter. It can be done by creating an instance of the '*Parameter*' class of the pytorch with the '*requires_grad=True*' option. This step is library dependent.

When the model is used for training or inference, the following Algorithm 2 is used to forward input signals to the next layer. First, the layer needs to load two kinds of kernels. The classical can be loaded by simply reading the values from the model. On the other hand, the wavelet kernel should generate kernel values using given scales ($S[i]$) and the wavelet function (ψ). Since the generation of wavelet kernel values must be done every time the scale value changes, it takes longer to train/infer than the classical kernel. Once the input signal is padded and ready for convolution, the input signal is convolved with two different wavelet kernels, respectively. Finally, the layer concatenates the two convolution results and returns them. Additional activation functions, pooling, batch normalization, and more can be applied to the returned values of Algorithm 2. The procedure for initializing and training the model with ASW-CL is as follows:

1) (ASW-CL Initialization) Initialize ASW-CL using Algorithm 1 when a model is created. Here, wavelet scales are registered as trainable parameters of the model.
2) (ASW-CL Forwarding) When a batch of input signals is fed into ASW-CL during the inference process, it decomposes the frequencies on the fly using Algorithm 2 and feeds them to the next layer.
3) Apply additional activation functions, pooling, batch normalization, and more to the results of ASW-CL.
4) Compute the final output of the model and then calculate the error using a loss function.
5) Update the trainable parameters of the model using error backpropagation. The wavelet scales are also updated at this time. We can delegate this task to the engine of the library.
6) Repeat (2)-(5) until training is complete.

In order for ASW-CL to work properly, some constraints are required. First, the ASW-CL must be at the front of the model. That is, an input of the ASW-CL must be raw signals, not the output of a previous feature extraction layer. Since the wavelet transformation is a signal-processing method, it can be used as the first feature extraction layer of the model for multi-frequency analysis. Second, ASW-CL should only be combined with genuine CNN models. This means that ASW-CL may not work well with models that use modified CNN primitives. We discuss this limitation further in Section V-E. We show an example of 1D-CNN adopting the ASW-CL with three wavelet kernels and three classic kernels in Figure 4.

To demonstrate the feasibility of our feature extraction layer, we show the evolution of key guessing entropy (see Section V-B), the result of side-channel analysis, using three different models in Figure 5. The first model is a genuine CNN, which is the same as the model that will be used in Section V. See Table 5 and Table 9 in Appendix A for the structure and training information of this model. The second model is based on the first model, with only the first

Algorithm 1 Pseudo-Code for ASW-CL Initialization

Input: N_{wk}: The number of wavelet kernels,
L_k: The length of each kernels,
f_c: Center frequency of the wavelet,
f_{max}: Maximum frequency to be extracted,
SR: Sampling rate of signals (samples per second)
Ensure: ASW-CL initialization

1: $S \leftarrow [0, 0, \cdots 0]$ ▷ Len.: (N_{wk})
2: $s_{min} \leftarrow (f_c \times SR)/f_{max}$ ▷ According to eq. (8)
3: $s_{max} \leftarrow$ calc_max_scale(L_k) ▷ Wavelet/L_k-dependent
4: **for** $i = 1$ to N_{wk} **do**
5: $S[i] \leftarrow$ random_gen(s_{min}, s_{max}) ▷ $s_{min} \sim s_{max}$
6: **end for**
7: set_initial_wavelet_scale(S)
8: convert_to_trainable(S) ▷ Library-dependent

Algorithm 2 Pseudo-Code for ASW-CL Forwarding

Input: X: Batch of input signals,
N_{nk}: The number of classical kernels,
N_{wk}: The number of wavelet kernels,
S: Wavelet scales (N_{wk} scales),
L_k: The length of each kernels
Output: Signals decomposed by wavelet scales S

1: $n\_kernels \leftarrow$ load_classical_kernels(N_{nk})
2: $w\_kernels \leftarrow [[0, 0, \cdots 0], \cdots]$ ▷ Shape: (N_{wk}, L_k)
3: **for** $i = 1$ to N_{wk} **do**
4: $w\_kernels[i] \leftarrow \psi(S[i], L_k)$ ▷ Generate wavelets
5: **end for**
6: $X \leftarrow$ add_pad(X) ▷ Apply 'same padding'
7: $X\_wk \leftarrow$ Conv1D$(X, w_kernels)$
8: $X\_nk \leftarrow$ Conv1D$(X, n_kernels)$
9: $X \leftarrow$ Concatenate(X_wk, X_nk)
10: **return** X ▷ Activation and pooling are applied after ASW-CL forwarding

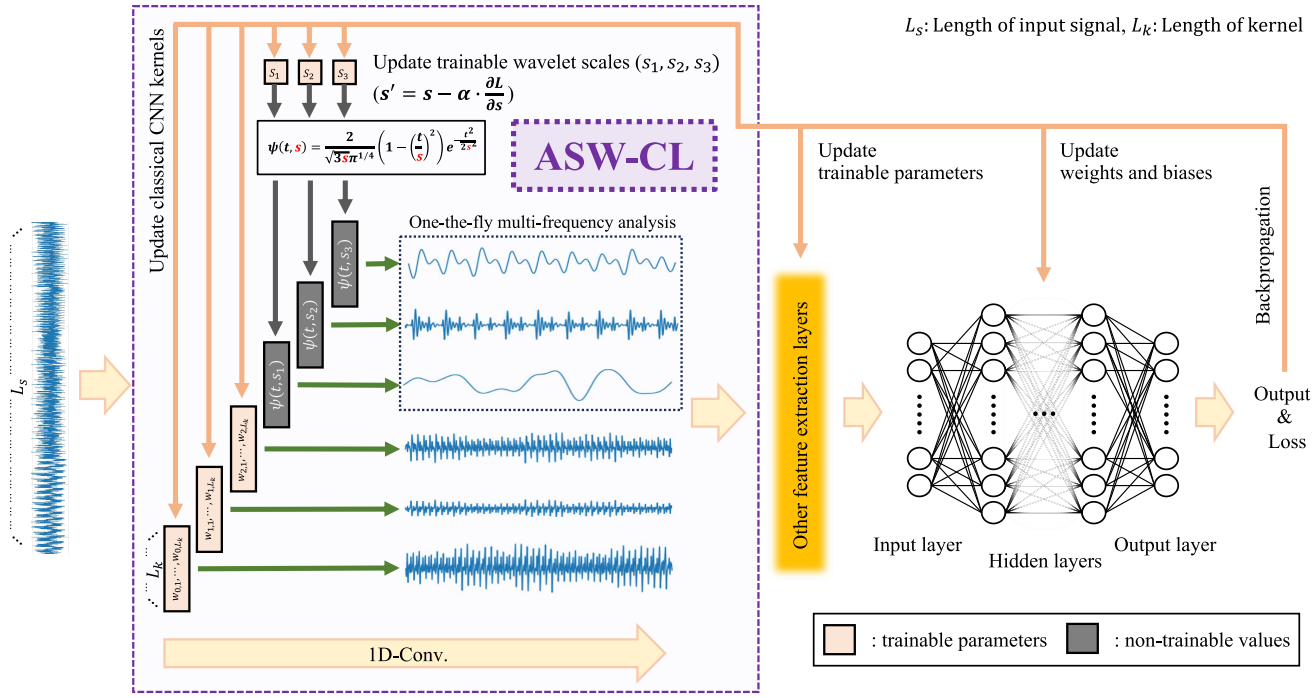


FIGURE 4. An example of a 1D-CNN model with ASW-CL consisting of three wavelet kernels and classical kernels, respectively.

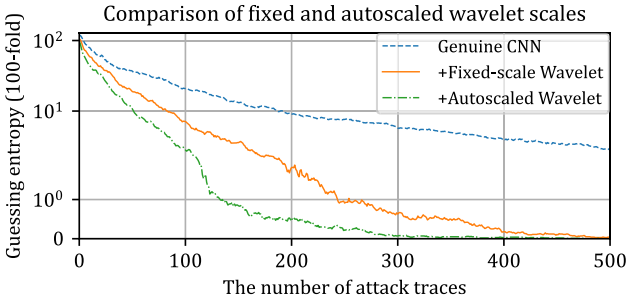


FIGURE 5. Comparison of GE evolution caused by the insertion of ASW-CL into a genuine CNN. The results show that inserting an ASW-CL with autoscaling enabled can significantly improve performance.

convolutional layer changed to ASW-CL. See Table 7 and Table 10 in Appendix A for the structure and training information of this model. The third model is the same structure as the second model, but with the autoscaling feature disabled (fixed-scale wavelet). As a result, we found that simply inserting ASW-CL into the existing CNN model could improve performance. Furthermore, the result shows that the performance can be improved even when autoscaling is disabled if randomly chosen and fixed wavelet scales can improve the SNR for synchronous signals (see Figure 1). This result suggests that the ASW-CL is superior in both performance and efficiency to statically applying CWT outside the model.

V. EVALUATION AND DISCUSSION

A. EXPERIMENTAL SETUP

For our experiment, we use the ASCAD dataset, which was released in 2018 and is the de facto standard for deep learning-based side-channel analysis. The ASCAD dataset is divided into two versions depending on the target hardware and implementation. However, ASCAD version 2 is rarely used because it appeared only recently; there are not many reference results [20]. Therefore, we use only version 1 (ATmega8515 @4MHz) which is EM side-channel leakage measured at 2GS/s. The ASCAD version 1 is also divided into two types depending on whether the key of the profiling dataset is variable or fixed. We use the fixed key version, which is mostly used for evaluation and has a lot of reference results. In addition, the ASCAD dataset provides an extracted version consisting of 700 samples in order to avoid unnecessary signal processing. The extracted version is divided into a jitter-free version and jitter-added versions with the 50-sample and 100-sample windows. We denote jitter-free version and jitter-added versions with 50-sample, and 100-sample windows as $ASCAD^{fixed}_{desync0}$, $ASCAD^{fixed}_{desync50}$, and $ASCAD^{fixed}_{desync100}$, respectively. We train the model using 50k profiling traces and evaluate performance using 10k attack traces in all experiments of this paper. Here, the profiling traces and the attack traces do not overlap, i.e., the new traces are used to evaluate our attacks.

We performed all the experiments in this paper using the PyTorch (v1.12.1) library in Python (v3.9). That is, we reconstructed the reference models written in Keras and TensorFlow into PyTorch. And the specifications of our experimental PC are as follows:

- CPU: Intel(R) Core(TM) i9-12900K
- RAM: DDR4-3200 128GB
- GPU: Nvidia GeForce RTX3090 Ti (24GB memory)

B. PERFORMANCE METRIC

Several state-of-the-art works [28], [30], [31], [37] adopt a performance metric called Nt_{ge} which is proposed in [37]. The Nt_{ge} denotes the number of traces needed to reach a constant Guessing Entropy (GE) of 0. Here, the GE denotes the rank of a real key in a probability vector of the guessing key. That is, the values of GE are in the range of $[0, |k| - 1]$ (or $[1, |k|]$) where $|k|$ is the cryptographic key space. The Nt_{ge} is a well-defined performance metric for deep learning-based side-channel analysis, which reflects the *real world's attack requirements* that GE must reach a constant 0.

However, Nt_{ge} is unsuitable for performance comparisons *in academia* due to the significant deviation and non-convergence problems. The Nt_{ge} changes sensitively to even a little error because the threshold of the GE is 0, *the minimum value of the GE*. To demonstrate this, we show the Nt_{ge} according to the number of iterations (for calculating the average of GE) for reliability in Figure 6 and Table 3. Here, we indicate the threshold as a superscript of Nt_{ge}, i.e., $Nt_{ge}^{th=0}$ denotes the existing Nt_{ge} of [37]. We found that the $Nt_{ge}^{th=0}$ does not converge and the deviation does not decrease even when the number of iterations is increased to improve reliability. Therefore, $Nt_{ge}^{th=0}$ may not have good reliability when comparing the performance of the models.

The problems of $Nt_{ge}^{th=0}$ can be mitigated by loosening the threshold to 1 rather than 0. We have confirmed that by loosening the threshold from 0 to 1, $Nt_{ge}^{th=1}$ converges and shows low deviation as shown in Table 3 and Figure 6. The $Nt_{ge}^{th=1}$ cannot completely reflect the attack requirements in the real world, but we can compare the relative performance more accurately and reliably. Therefore, we adopt $Nt_{ge}^{th=1}$ as a performance metric in this paper. We describe a pseudo-code for calculating $Nt_{ge}^{th=*}$ of R1-SubBytes (AES-128) using probability matrix (predicted from a deep learning model) of N_{atk} measurements in Algorithm 3 where $N_{IV} = |IV|$ and $N_K = |K|$, i.e., 1-byte.

TABLE 3. A summary of $Nt_{ge}^{th=*}$ according to the number of iterations. In contrast to $Nt_{ge}^{th=0}$, $Nt_{ge}^{th=1}$ converges as the number of iterations increases.

$Nt_{ge}^{th=*}$	Iter. per case	Mean	SD†	Time/case(s)‡
$Nt_{ge}^{th=0}$	10	720.54	237.64	1.2
	100	1109.17	198.98	8.1
	1,000	1499.00	228.30	75.0
	10,000	1851.43	204.55	758.8
$Nt_{ge}^{th=1}$	10	**293.49**	**77.04**	1.2
	100	**311.03**	**35.00**	8.1
	1,000	**310.89**	**11.62**	75.0
	10,000	**310.61**	**3.58**	758.8

†: Standard deviation, ‡: Using 5,000 attack traces per case.

C. EVALUATION

To evaluate the proposed ASW-CL, we analyze the performance gains that can be achieved by simply inserting our ASW-CL into state-of-the-art deep learning models. For experiments, we selected two state-of-the-art 1D-CNN models for side-channel analysis from [9] and [37]. We show the structure of the CNNs proposed in [9] and [37] in Table 5 and Table 7 of Appendix A, respectively. Here, the most

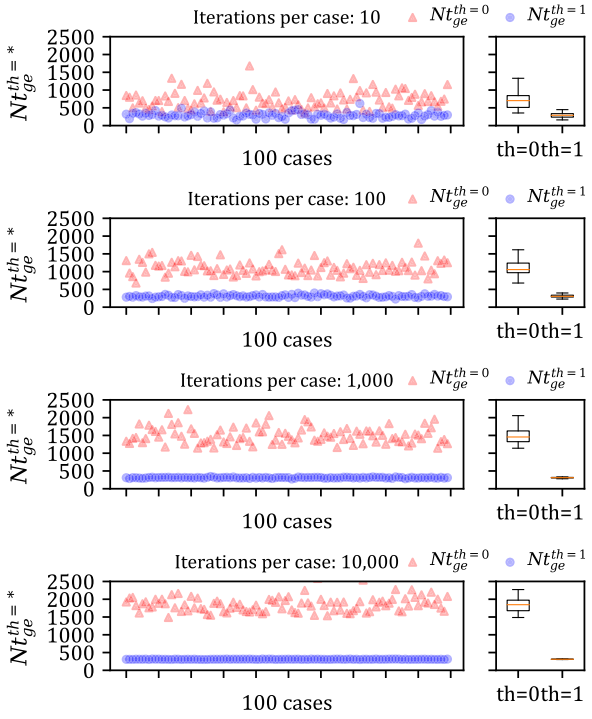
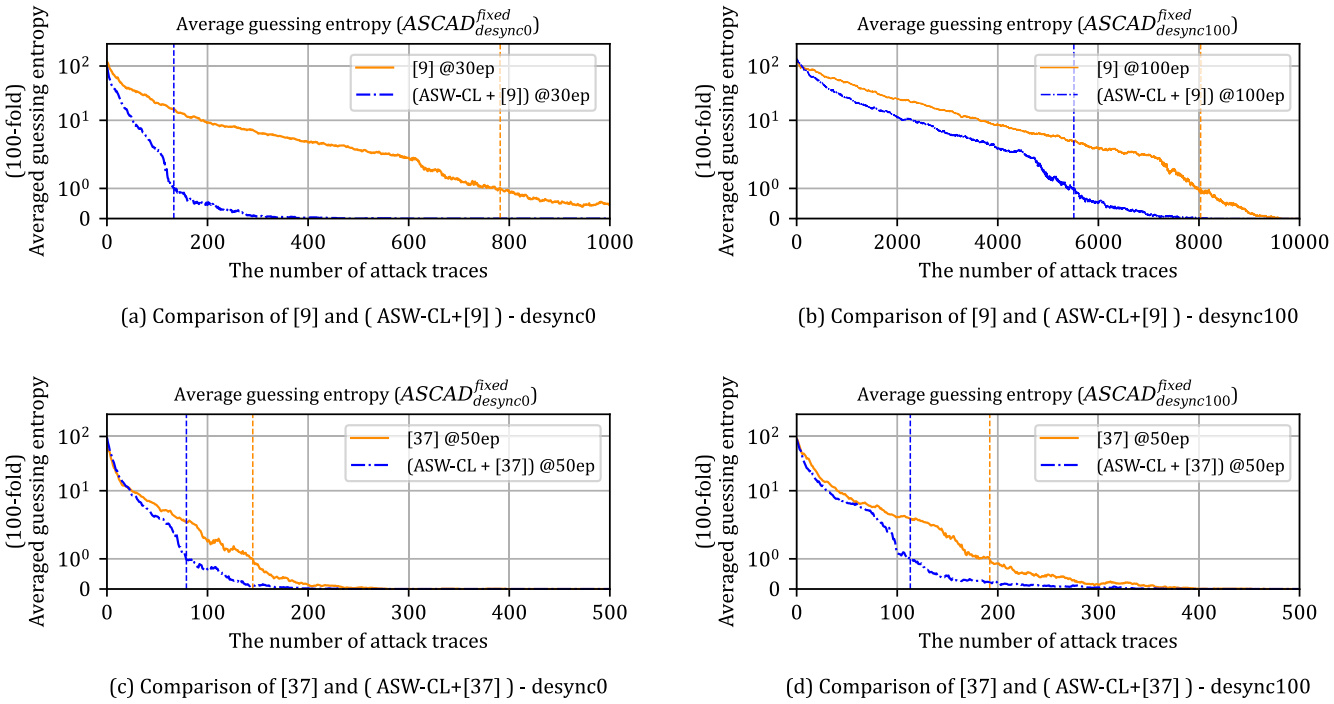


FIGURE 6. Distributions of $Nt_{ge}^{th=0}$ and $Nt_{ge}^{th=1}$ according to the number of iterations (guessing entropy calculation). In contrast to $Nt_{ge}^{th=0}$, $Nt_{ge}^{th=1}$ converges as the number of iterations increases.

Algorithm 3 Pseudo-Code for Calculating $Nt_{ge}^{th=*}$ of AES-128 R1-SubBytes (IV) Using Maximum Log-Likelihood

Input: th: Threshold of $Nt_{ge}^{th=*}$,
$fold$: Repetition number of GE calculation,
$real\_key$: Real key of entire attack traces (1byte),
P_{IV}: Matrix of predicted probabilities ($N_{atk} \times N_{IV}$),
PT: Matrix of target plaintext byte ($1 \times N_{atk}$)
Output: $Nt_{ge}^{th=*}$

$avg\_ge \leftarrow [0, 0, \cdots 0]$ ▷ Len.: N_{atk}
for $f = 1$ to $fold$ **do**
  $key\_prob \leftarrow [0, 0, \cdots 0]$ ▷ Len.: N_{key} ($= |k|$)
  for $t = 1$ to N_{atk} **do**
    $r\_P_{IV}, r\_PT \leftarrow \text{randomly\_select}_t(P_{IV}, PT)$
    for $i = 1$ to t **do**
      for $k = 1$ to N_{key} **do**
        $prob \leftarrow r\_P_{IV}[i][sbox[r\_PT[i] \oplus k]]$
        $key\_prob[k] \leftarrow key\_prob[k] + log(prob)$
      end for
    end for
    $ge \leftarrow \text{calculate\_rank}(real\_key, key\_prob)$
    $avg\_ge[t] \leftarrow avg\_ge[t] + (ge/fold)$
  end for
end for
return $\text{argmin}_n(avg\_ge[n:] \leq th)$



(a) Comparison of [9] and (ASW-CL+[9]) - desync0

(b) Comparison of [9] and (ASW-CL+[9]) - desync100

(c) Comparison of [37] and (ASW-CL+[37]) - desync0

(d) Comparison of [37] and (ASW-CL+[37]) - desync100

FIGURE 7. Comparison of GE evolution changes caused by applying ASW-CL to two state-of-the-art CNN models from [9] and [37] on two datasets $ASCAD^{model}_{desync0}$ and $ASCAD^{model}_{desync100}$. The vertical lines indicate $Nt^{th=1}_{ge}$. The results show that applying ASW-CL can improve the attack performance.

TABLE 4. A summary of the experimental results of Figure 7. The experimental results show that the ASW-CL can improve the attack performance.

Dataset	Ref.†	Model	Complexity‡	Epoch	Time(s)	$Nt^{th=1}_{ge}$	Difference§
$ASCAD^{fixed}_{desync0}$	(a)	CNN proposed in [9]	66,652,544	30ep	273	782	-
		ASW-CL + [9]	66,646,208	30ep	423	**133**	**-649 (83%↓)**
	(c)	CNN proposed in [37]	141,176	50ep	71	145	-
		ASW-CL + [37]	143,352	50ep	103	**79**	**-66 (47%↓)**
$ASCAD^{fixed}_{desync100}$	(b)	CNN proposed in [9]	66,652,544	100ep	882	8,033	-
		ASW-CL + [9]	66,646,208	100ep	1,230	**5,512**	**-2,521 (31%↓)**
	(d)	CNN proposed in [37]	141,176	50ep	71	192	-
		ASW-CL + [37]	143,352	50ep	103	**113**	**-79 (41%↓)**

†: The identifier in Figure 7, ‡: The number of trainable parameters, §: Differences caused by ASW-CL adoption.

important consideration in selecting state-of-the-art models is the applicability of ASW-CL. As mentioned in Section IV, the ASW-CL is only applicable to genuine CNN models where the primitives are not modified. Literature [9] and [37] are state-of-the-art papers that study the performance improvement of models using genuine CNNs (see Section V-E). Then, we implemented two more models by changing the first convolutional layer of each model to ASW-CL. The models with ASW-CL are described in Table 6 and Table 8, respectively. In summary, we evaluate the proposed ASW-CL using two datasets ($ASCAD^{fixed}_{desync0}$, $ASCAD^{fixed}_{desync100}$) and the following four models:

- CNN proposed in [9] (See Table 5)
- **ASW-CL** + CNN proposed in [9] (See Table 6)
- CNN proposed in [37] (See Table 7)
- **ASW-CL** + CNN proposed in [37] (See Table 8)

Following Figure 7 shows the evolution of GE (100-fold) to show the performance improvement caused by the ASW-CL insertion. And Table 4 summarizes the results of Figure 7. All of our results show that the performance can be improved by simply inserting ASW-CL into existing state-of-the-art CNN models. In the case of the CNN proposed in [9], $Nt^{th=1}_{ge}$ was reduced by 649 (83%) for the $ASCAD^{fixed}_{desync0}$ and by 2,521 (31%) for the $ASCAD^{fixed}_{desync100}$ when the ASW-CL was inserted. Likewise, in the case of the CNN proposed in [37], $Nt^{th=1}_{ge}$ was reduced by 66 (47%) for the $ASCAD^{fixed}_{desync0}$ and by 79 (41%) for the $ASCAD^{fixed}_{desync100}$ when the ASW-CL was inserted. Our experiments indicate that the degree of performance improvement decreases as the signal is more jittery. And, the more sophisticated the model, the fewer performance gains are likely to be, as there is not as much scope for performance improvement. Also, the running time increases regardless of whether the number of trainable parameters is increased or decreased (See section V-E). Nevertheless, the proposed ASW-CL can improve the performance and thus can be adopted in deep learning models. The trade-off

between increased time and improved performance should be considered when applying the ASW-CL.

D. HYPERPARAMETERS

The ASW-CL has three hyperparameters: the number of wavelet kernels, the length of the wavelet kernel, and the activation function. To show the impact of the two hyperparameters (the number of wavelet kernels and the activation function), we show the performance variation with (a) the number of wavelet kernels and (b) the type of activation function in Figure 8. In (a) of Figure 8, we found that there is no significant difference in performance depending on the number of wavelet kernels, except when all kernels are classical kernels (same as genuine CNN). And we found that the training performed poorly when using activation functions that limit the scale of the output such as sigmoid and tanh, in (b) of Figure 8. Therefore, we recommend using the ReLU-like functions (ReLU, Leaky ReLU, SELU) as an activation function. On the other hand, the length of the kernel should be determined based on the operating clock of the DUT, the sampling rate, the characteristics of countermeasure, and the amount of jitter, rather than a grid search-based decision.

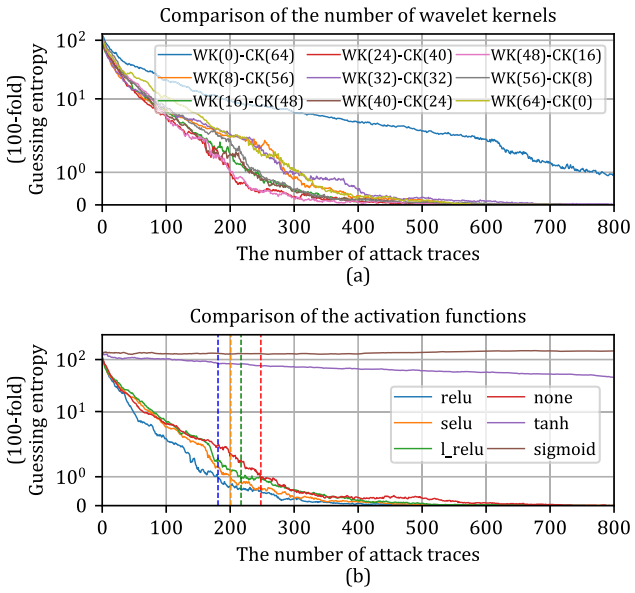


FIGURE 8. Comparison of GE evolution by hyperparameters. (a) shows the variation with the number of wavelet kernels, and (b) shows the variation with the type of activation function. The results show poor performance when no wavelet kernel is used in (a) and when tanh and sigmoid are used in (b).

E. DISCUSSIONS

1) COMPATIBLE DEEP LEARNING MODELS

The proposed ASW-CL is only applicable to CNN-like deep learning models because it is a modification of the existing convolutional layer. And the ASW-CL is based on the signal processing method named CWT. Therefore, ASW-CL can decompose the frequencies on the fly at the front of the model and passes them to the next layer. For this reason, it is important that the frequency components decomposed by ASW-CL can be fully utilized by subsequent feature extraction layers. This means that it is not applicable to some state-of-the-art models such as the Bilinear CNN proposed in [31], which modifies the primitives of the CNN by analyzing and adopting the characteristic of second-order attacks. Therefore, the proposed ASW-CL should be applied to genuine CNN with unmodified primitives.

2) COMPATIBLE WAVELETS

In the field of side-channel analysis, we have experimentally confirmed that the Ricker wavelet can increase SNR the most over wavelets such as Morlet, Daubechies, and more, in general. We consider this as a characteristic of a side-channel signal with a peak at the clock level. Therefore, we have used the Ricker wavelet for all experiments in this paper. However, all wavelets which can be differentiated using the 'autograd' module can be adopted by ASW-CL (in the case of the PyTorch). The wavelet should be selected based on the side-channel signal characteristics of the target device.

3) EFFICIENCY

Our work was carried out in terms of proof of concept, not including sophistication and optimization (fast implementation). That is, compared to the existing models, even though the complexity is slightly reduced, the running time of the ASW-CL-equipped model is increased by about 40%-50%. Therefore, follow-up studies related to sophistication and optimization are needed for performance improvement and reducing running time.

4) OPTIMAL WAVELET SCALE

In our work, we leveraged the neural network training process in order to automatically search and determine the *optimal wavelet scale*, which is a critical parameter of wavelet transform (especially in CWT) and does not have an efficient searching algorithm. Here, we need to pay attention to the word *'optimal'*. In the proposed ASW-CL, the wavelet scale was adjusted, eventually contributing to performance improvement. That is, the *optimal* scale was determined. However, we do not know whether this optimal scale will also be optimal in classical analysis. In other words, a deep learning model can extract information that is helpful only to itself but may not be to others (e.g., classical analysis). Since deep learning is still a black box model, we cannot know how the signals in the specific frequency bands (determined by the model) are combined inside the model. Therefore, follow-up studies are needed not only in terms of performance improvement but also in exploring parameters.

VI. CONCLUSION

In this paper, we proposed a novel feature extraction layer called ASW-CL that takes advantage of wavelet transform and neural network training. The proposed ASW-CL enables an evaluator to perform one-the-fly multi-frequency analysis in a single model without considering the wavelet scale

parameter or significantly increasing complexity. The ASW-CL is a kind of feature extraction layer; it can be combined with a general deep-learning model (especially CNN) for side-channel analysis. And we confirmed that the proposed ASW-CL could improve the performance of the side-channel analysis using the open dataset ASCAD, which is the de facto standard in deep learning-based side-channel analysis.

Our work goes beyond the performance improvement of deep learning models and includes the novel part of automatic parameter search (wavelet scale of CWT). However, this work was a proof-of-concept on the novel ideas and the resulting performance improvement of deep learning-based side-channel analysis. Therefore, as mentioned in Section V-E above, follow-up studies are needed. We expect that research similar to our work will be conducted to determine parameters based on the gradient for other specific algorithms.

APPENDIX A

We describe the structure of the CNN proposed in [9], and its combined structure with ASW-CL in Table 5 and Table 6, respectively. In the same context, we describe the structure of the CNN proposed in [37], and its combined structure with ASW-CL in Table 7 and Table 8, respectively. In addition, we provide details about the training parameters for Table 5 and Table 6 in Table 9, and for Table 7 and Table 8 in Table 10, respectively.

TABLE 5. A structure of the CNN proposed in [9].

Layer	#Kernel #Nodes Pool. size	Act. / Pool.
Input	700	-
Conv. 1D	64 (len. 11)	ReLU
Pool. 1D	(2, 2)	Avg.
Conv. 1D	128 (len. 11)	ReLU
Pool. 1D	(2, 2)	Avg.
Conv. 1D	256 (len. 11)	ReLU
Pool. 1D	(2, 2)	Avg.
Conv. 1D	512 (len. 11)	ReLU
Pool. 1D	(2, 2)	Avg.
Conv. 1D	512 (len. 11)	ReLU
Pool. 1D	(2, 2)	Avg.
F.C.	10,752	-
F.C.	4,096	ReLU
F.C.	4,096	ReLU
F.C.	256	Softmax

TABLE 6. A structure of the ASW-CL + CNN proposed in [9].

Layer	#Kernel #Nodes Pool. size	Act. / Pool.
Input	700	-
ASW-CL	**64 (32/32) (len. 100)**	**ReLU**
Pool. 1D	(2, 2)	Avg.
Conv. 1D	128 (len. 11)	ReLU
Pool. 1D	(2, 2)	Avg.
Conv. 1D	256 (len. 11)	ReLU
Pool. 1D	(2, 2)	Avg.
Conv. 1D	512 (len. 11)	ReLU
Pool. 1D	(2, 2)	Avg.
Conv. 1D	512 (len. 11)	ReLU
Pool. 1D	(2, 2)	Avg.
F.C.	10,752	-
F.C.	4,096	ReLU
F.C.	4,096	ReLU
F.C.	256	Softmax

TABLE 7. A structure of the CNN proposed in [37].

Layer	#Kernel #Nodes Pool. size	Act. / Pool.
Input	700	-
Conv. 1D	32 (len. 1)	SELU
Batch Norm.	-	-
Pool. 1D	(2, 2)	Avg.
Conv. 1D	64 (len. 50)	SELU
Batch Norm.	-	-
Pool. 1D	(2, 2)	Avg.
Conv. 1D	128 (len. 3)	SELU
Batch Norm.	-	-
Pool. 1D	(2, 2)	Avg.
F.C.	384	-
F.C.	20	SELU
F.C.	20	SELU
F.C.	256	Softmax

TABLE 8. A structure of the ASW-CL + CNN proposed in [37].

Layer	#Kernel #Nodes Pool. size	Act. / Pool.
Input	700	-
ASW-CL	**32 (16/16) (len. 100)**	**SELU**
Batch Norm.	-	-
Pool. 1D	(2, 2)	Avg.
Conv. 1D	64 (len. 50)	SELU
Batch Norm.	-	-
Pool. 1D	(2, 2)	Avg.
Conv. 1D	128 (len. 3)	SELU
Batch Norm.	-	-
Pool. 1D	(2, 2)	Avg.
F.C.	384	-
F.C.	20	SELU
F.C.	20	SELU
F.C.	256	Softmax

TABLE 9. Training parameters of the (ASW-CL +) CNN proposed in [9].

Training parameters	Values
Optimizer	RMSProb
Learning rate	0.00001
Loss function	Cross Entropy (categorical)
Label	$sbox[p[3] \oplus k[3]]$ (One-hot encoding)

TABLE 10. Training parameters of the (ASW-CL +) CNN proposed in [37].

Training parameters	Values
Optimizer	Adam
Learning rate	0.005 (We do not use one-cycle policy)
Loss function	Cross Entropy (categorical)
Label	$sbox[p[3] \oplus k[3]]$ (One-hot encoding)

REFERENCES

[1] P. C. Kocher, "Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems," in *Proc. Int. Crytology Conf. (CRYPTO)*, Santa Barbara, CA, USA, 1996, pp. 104–113, doi: 10.1007/3-540-68697-5_9.

[2] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Int. Crytology Conf. (CRYPTO)*, Santa Barbara, CA, USA, 1999, pp. 388–397, doi: 10.1007/3-540-48405-1_25.

[3] P. Socha, V. Mikovsk, and M. Novotn, "A comprehensive survey on the non-invasive passive side-channel analysis," *Sensors*, vol. 22, no. 21, pp. 1–37, Oct. 2022, doi: 10.3390/s22218096.

[4] T. Le, J. Clediere, C. Serviere, and J. L. Lacoume, "How can signal processing benefit side channel attacks," in *Proc. IEEE Workshop Signal Process. Appl. Public Secur. Forensics*, Washington, DC, USA, Apr. 2007, pp. 1–7.

[5] S. Jin, P. Johansson, H. Kim, and S. Hong, "Enhancing time-frequency analysis with zero-mean preprocessing," *Sensors*, vol. 22, no. 7, pp. 1–18, 2022, doi: 10.3390/s22072477.

[6] C. E. Heil and D. F. Walnut, "Continuous and discrete wavelet transforms," *SIAM Rev.*, vol. 31, no. 4, pp. 628–666, Dec. 1989, doi: 10.1137/1031129.

[7] N. Debande, Y. Souissi, M. A. E. Aabid, S. Guilley, and J.-L. Danger, "Wavelet transform based pre-processing for side channel analysis," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchitecture Workshops*, Dec. 2012, pp. 32–38, doi: 10.1109/MICROW.2012.15.

[8] D. Bae and J. Ha, "Implementation of disassembler on microcontroller using side-channel power consumption leakage," *Sensors*, vol. 22, no. 15, pp. 1–17, Aug. 2022, doi: 10.3390/s22155900.

[9] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Deep learning for side-channel analysis and introduction to ASCAD database," *J. Cryptograph. Eng.*, vol. 10, no. 2, pp. 163–188, Jun. 2020, doi: 10.1007/s13389-019-00220-8.

[10] C. O'Flynn and Z. D. Chen, "Side channel power analysis of an AES-256 bootloader," in *Proc. IEEE 28th Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2015, pp. 750–755, doi: 10.1109/CCECE.2015.7129369.

[11] R. Mayer-Sommer, "Smartly analyzing the simplicity and the power of simple power analysis on smartcards," in *Proc. Int. Work. Cryptograph. Hardw. Embedded Syst. (CHES)*, Worcester, MA, USA, 2000, pp. 78–92, doi: 10.1007/3-540-44499-8_6.

[12] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, Boston, MA, USA, 2004, pp. 16–29, doi: 10.1007/978-3-540-28632-5_2.

[13] B. Timon, "Non-profiled deep learning-based side-channel attacks with sensitivity analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 2, pp. 107–131, Feb. 2019, doi: 10.13154/tches.v2019.i2.107-131.

[14] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)* Berlin, Germany, 2002, pp. 13–28, doi: 10.1007/3-540-36400-5_3.

[15] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: A first study," *J. Cryptograph. Eng.*, vol. 1, no. 4, pp. 293–302, Dec. 2011, doi: 10.1007/s13389-011-0023-x.

[16] E. Ghaderpour, S. D. Pagiatakis, and Q. K. Hassan, "A survey on change detection and time series analysis with applications," *Appl. Sci.*, vol. 11, no. 13, p. 6141, Jul. 2021, doi: 10.3390/app11136141.

[17] J. Park, X. Xu, Y. Jin, D. Forte, and M. Tehranipoor, "Power-based side-channel instruction-level disassembler," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6, doi: 10.1109/DAC.2018.8465848.

[18] A. Garg and N. Karimian, "Leveraging deep CNN and transfer learning for side-channel attack," in *Proc. 22nd Int. Symp. Quality Electron. Design (ISQED)*, Apr. 2021, pp. 91–96, doi: 10.1109/ISQED51717.2021.9424305.

[19] E. Ghaderpour and S. Ghaderpour, "Least-squares spectral and wavelet analyses of V455 andromedae time series: The life after the super-outburst," *Publications Astronomical Soc. Pacific*, vol. 132, no. 1017, Oct. 2020, Art. no. 114504, doi: 10.1088/1538-3873/abaf04.

[20] S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina, "SoK: Deep learning-based physical side-channel analysis," *ACM Comput. Surv.*, vol. 55, no. 11, pp. 1–35, Nov. 2023, doi: 10.1145/3569577.

[21] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jul. 1989, doi: 10.1016/0893-6080(89)90020-8.

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[23] S. Jin, S. Kim, H. Kim, and S. Hong, "Recent advances in deep learning-based side-channel analysis," *ETRI J.*, vol. 42, no. 2, pp. 292–304, Apr. 2020, doi: 10.4218/etrij.2019-0163.

[24] T. S. Messerges, "Securing the AES finalists against power analysis attacks," in *Proc. 7th Int. Workshop Fast Softw. Encryption (FSE)*, Leuven, Belgium, 2002, pp. 150–164, doi: 10.1007/3-540-44706-7_11.

[25] J. Coron and I. Kizhvatov, "An efficient method for random delay generation in embedded software," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, Lausanne, Switzerland, 2009, pp. 156–170, doi: 10.1007/978-3-642-04138-9_12.

[26] N. Veyrat-Charvillon, M. Medwed, S. Kerckhof, and F. X. Standaert, "Shuffling against side-channel attacks: A comprehensive study with cautionary note," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT)*, Beijing, China, 2012, pp. 740–757, doi: 10.1007/978-3-642-34961-4_44.

[27] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *Proc. Int. Conf. Secur., Privacy, Appl. Cryptogr. Eng. (SPACE)*, Hyderabad, India, 2016, pp. 3–26, doi: 10.1007/978-3-319-49445-6_1.

[28] L. Wouters, V. Arribas, B. Gierlichs, and B. Preneel, "Revisiting a methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, pp. 147–168, Jun. 2020, doi: 10.13154/tches.v2020.i3.147-168.

[29] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise. Unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 3, pp. 148–179, May 2019, doi: 10.13154/tches.v2019.i3.148-179.

[30] J. Rijsdijk, L. Wu, G. Perin, and S. Picek, "Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, pp. 677–707, Jul. 2021, doi: 10.46586/tches.v2021.i3.677-707.

[31] P. Cao, C. Zhang, X. Lu, D. Gu, and S. Xu, "Improving deep learning based second-order side-channel analysis with bilinear CNN," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3863–3876, 2022, doi: 10.1109/TIFS.2022.3216959.

[32] S. Fujieda, K. Takayama, and T. Hachisuka, "Wavelet convolutional neural networks," 2018, *arXiv:1805.08620*.

[33] P. Liu, H. Zhang, W. Lian, and W. Zuo, "Multi-level wavelet convolutional neural networks," *IEEE Access*, vol. 7, pp. 74973–74985, 2019, doi: 10.1109/ACCESS.2019.2921451.

[34] J.-W. Liu, F.-L. Zuo, Y.-X. Guo, T.-Y. Li, and J.-M. Chen, "Research on improved wavelet convolutional wavelet neural networks," *Int. J. Speech Technol.*, vol. 51, no. 6, pp. 4106–4126, Jun. 2021, doi: 10.1007/s10489-020-02015-5.

[35] D. Paraskevopoulos, C. Spandonidis, and F. Giannopoulos, "Hybrid Wavelet–CNN fault diagnosis method for ships' power systems," *Signals*, vol. 4, no. 1, pp. 150–166, Feb. 2023, doi: 10.3390/signals4010008.

[36] Z. Martinasek and V. Zeman, "Innovative method of the power analysis," *Radioengineering*, vol. 22, no. 2, pp. 586–594, 2013.

[37] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, no. 1, pp. 1–36, 2020, doi: 10.13154/tches.v2020.i1.1-36.

[38] L. Chang, Y. Wei, S. He, and X. Pan, "Research on side-channel analysis based on deep learning with different sample data," *Appl. Sci.*, vol. 12, no. 16, p. 8246, Aug. 2022, doi: 10.3390/app12168246.

[39] Q. Lei, Z. Yang, Q. Wang, Y. Ding, Z. Ma, and A. Wang, "Autoencoder assist: An efficient profiling attack on high-dimensional datasets," in *Proc. Int. Conf. Inf. Commun. Secur. (ICICS)*, Canterbury, U.K., 2022, pp. 324–341, doi: 10.1007/978-3-031-15777-6_18.

[40] K. Kneževic, J. Fulir, D. Jakobovic, S. Picek, and M. Đurasevic, "NeuroSCA: Evolving activation functions for side-channel analysis," *IEEE Access*, vol. 11, pp. 284–299, 2023, doi: 10.1109/ACCESS.2022.3232064.

DAEHYEON BAE received the B.S. and M.S. degrees in information security from Hoseo University, South Korea, in 2021 and 2022, respectively. He is currently pursuing the Ph.D. degree with the School of Cybersecurity, Korea University, Seoul, South Korea. His research interests include side-channel attacks, hardware security, and deep learning-based side-channel analysis.

DONGJUN PARK received the B.S. degree in information security from Sejong University, Seoul, South Korea, in 2018, and the M.S. degree in information security from Korea University, Seoul, in 2020, where he is currently pursuing the Ph.D. degree. Since 2018, he has been a Research Assistant with the Institute of Cyber Security and Privacy (ICSP), School of Cybersecurity, Korea University. His research interests include cryptography, hardware security, and side-channel attacks.

GYUSANG KIM received the B.S. degree in mathematics from Yonsei University, Seoul, South Korea, in 2020. He is currently pursuing the joint M.S. and Ph.D. degrees in information security with Korea University, Seoul. His research interests include cryptography, post-quantum cryptography, and side-channel attacks.

MINSIG CHOI received the B.S. degree in information security from Hongik University, Seoul, South Korea, in 2023. He is currently pursuing the M.S. degree in information security with Korea University, Seoul. Since 2023, he has been a Research Assistant with the Institute of Cyber Security and Privacy (ICSP), School of Cybersecurity, Korea University. His research interests include cryptography and side-channel attacks.

NAYEON LEE received the B.S. and M.S. degrees in information security from Korea University, South Korea, in 2022 and 2023, respectively. Her research interests include side-channel attacks and deep learning-based side-channel analysis.

HEESEOK KIM (Member, IEEE) received the B.S. degree in mathematics from Yonsei University, Seoul, South Korea, in 2006, and the M.S. and Ph.D. degrees in engineering and information security from Korea University, Seoul, in 2008 and 2011, respectively. He was a Postdoctoral Researcher with the University of Bristol, U.K., from 2011 to 2012. From 2013 to 2016, he was a Senior Researcher with the Korea Institute of Science and Technology Information (KISTI). Since 2016, he has been with Korea University. His research interests include side-channel attacks, cryptography, and network security.

SEOKHIE HONG (Member, IEEE) received the M.S. and Ph.D. degrees in mathematics from Korea University, in 1997 and 2001, respectively. From 2000 to 2004, he was with Security Technologies Inc. From 2004 to 2005, he was a Postdoctoral Researcher with the COSIC, KU Leuven, Belgium. He joined the Graduate School of Cybersecurity, Korea University. His research interests include cryptography, public and symmetric key cryptosystems, hash functions, and message authentication codes.