electronics

MDPI

Article

Side-Channel Profiling Attack Based on CNNs' Backbone Structure Variant

Weifeng Liu [1,2], Wenchang Li [3,4,*], Xiaodong Cao [1,2], Yihao Fu [5], Xiang Li [1,2], Jian Liu [6], Aidong Chen [5], Yanlong Zhang [7], Shuo Wang [7] and Jing Zhou [7]

1 Artificial Intelligence and High-Speed Circuits Laboratory, Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China; weifengliu@semi.ac.cn (W.L.); xdcao@semi.ac.cn (X.C.); lixiang24@semi.ac.cn (X.L.)
2 University of Chinese Academy of Sciences, Beijing 100049, China
3 Key Laboratory of Solid-State Optoelectronic Information Technology, Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China
4 College of Microelectronics, University of Chinese Academy of Sciences, Beijing 100049, China
5 Multi-Agent Systems Research Center, School of Robotics, Beijing Union University, Beijing 100101, China; dysfuyihao@gmail.com (Y.F.); aidongise@gmail.com (A.C.)
6 State Key Laboratory of Semiconductor Physics and Chip Technologies, Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China; liujian@semi.ac.cn
7 Beijing Institute of Microelectronics Technology, Beijing 100076, China; zhangylsun@126.com (Y.Z.); wangshuo@mxtronics.com (S.W.); zhoujenny0915@126.com (J.Z.)
* Correspondence: liwc@semi.ac.cn

Abstract: This paper focuses on sorting out two backbone network variants' development routes for CNNs and transformers based on deep learning methods for side-channel analysis. Combining parametric and quantitative structural analyses, it is preferred that improvement research be carried out on the structure of the continuation CNN variants. Firstly, the comparative reproduction of Zaid's Efficient CNN and Wouters' Simplified Efficient CNN will be carried out; second, ResNet will be designed and implemented based on residual structure and the improved design of feature-coded CNNs; finally, the four methods in DPA_ contest v4.1, AES_RD, AES_HD, and ASCAD public datasets will be used for extensive analysis experiments to explore the effects of six preprocessing roles at three scales and further explore the impact of data enhancement by dataset noise, offset, and amplitude scaling. The experimental results show that ResNet based on the residual structure and the coded feature CNN proposed in this paper exhibit better performance advantages than the previous methods. Preferring a preprocessing method for each dataset can continue to reduce the mean rank; continuing to overlay data enhancement methods can make the model easier to converge and increase the model's generalization ability. To advance the research in the branch of CNNs and transformer variants in the SCA field, the model methods obtained from the above experiments and the datasets obtained from the processing have been made publicly available on GitHub.

Keywords: side-channel analysis; profiling attacks; deep learning; backbone networks

check for updates

Academic Editor: Dimitra I. Kaklamani

Received: 29 March 2025
Revised: 8 May 2025
Accepted: 9 May 2025
Published: 15 May 2025

Citation: Liu, W.; Li, W.; Cao, X.; Fu, Y.; Li, X.; Liu, J.; Chen, A.; Zhang, Y.; Wang, S.; Zhou, J. Side-Channel Profiling Attack Based on CNNs' Backbone Structure Variant. *Electronics* **2025**, *14*, 2006. https://doi.org/10.3390/electronics14102006

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the development of the hardware security field and cryptanalysis techniques, the new side-channel analysis (SCA) methods have seriously threatened the security of most cryptographic devices. Machine learning (ML) and deep learning (DL) techniques applied in side-channel analysis have overwhelming advantages. The analysis session

applied in SCA mainly performs hypothesis–guess–classification operations, which obtains key-guessing results.

The application of ML-SCA and DL-SCA started in 2016 when Maghrebi et al. [1] combined Random Forest (RF), Auto-Encoder (Auto-Encoder, AE), Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM) methodological processes. The application's goals were twofold: first, to complete the analysis process using as little curve acquisition as possible, and second, to still be able to implement effective analysis against guarded cryptographic algorithms.

To achieve these two goals, researchers have made more attempts on MLP [2], FFN [1,3–5], RNN [1,6,7], and CNN [1,7–10]. Among them, CNN and its variants [2,11–16] applied to perform side-channel analysis are the most widespread and effective because CNN is suitable for performing cryptographic guessing types of classification tasks. The transformer structure [17–19] has also been successively researched in the last two years; however, as a large network modeling structure, it requires multi-sample or even full-sample datasets, which puts new demands on the dataset.

The main contributions of this work are summarised as follows:

- Combining the evolution of two backbone networks, CNNs, and transformers in the field of side-channel analysis, supplementing the establishment of a general expression for the transformer structure, quantitatively comparing the model structure parameters and training hyperparameters, and refining a clearer form of side-by-side comparison.
- Propose a deep learning analysis method for the feature-encoded CNN structure—improve both the design and execution of the main model structure and add a one-step feature coding and decoding process so the network model can better understand the target leakage features and the method shows a good generalization ability on the comparison benchmarks of public datasets.
- Design to conduct complete ablation and control experiments; design to conduct {2 scales * 3 algorithms * 1 enhancement} ablation comparison experiments to complement previous unfinished work. The comparison of guessed entropy results with Nt_{GE} {1, 3, 5, 10} results is used to side-by-side corroborate the positive effects of preprocessing and data enhancement measures, while an optimal method is derived for all four datasets.

The rest of this paper is structured as follows: in Section 2, the basis of previous research work on the CNN backbone network and transformer and the structural features and performance limitations of the SOTA method are targeted to provide a theoretical basis for the improvement of this paper. Section 3 introduces the derivation of the basic principles of the deep learning method for key guessing against the dataset. Meanwhile, combined with the parameterized model SOTA method structural parameters and training hyperparameters, the feature encoding CNN network, preprocessing method, and the selection of the loss function are proposed in this paper. In Section 4, the dataset objects used for the experiments in this paper are introduced, and the computation of the two main evaluation indexes, namely, the model accuracy and the guessing entropy, are deduced. The configuration of the experimental platform, the experimental data, and the summary analysis of the results are explained. Finally, the full paper is summarised in Section 5.

2. Related Work

The application of DLSCA takes key guessing as the main computational goal, as a task to classify the corresponding probability outputs with key byte guessing, in line with typical tasks, such as machine decision making, computer vision, natural language processing, etc. The main vein of technological development also starts from machine learning algorithms, such as multilayer perceptual machine (MLP) to convolutional neural

network (CNN), and then develops to use the transformer as the main architecture of the model, with structural improvement, hyper-parameter tuning or loss function optimization to innovate for better analysis accuracy, efficiency and other metric improvements.

1. CNN backbone network and variants

The CNN model has shown good performance advantages since it has been applied to the side-channel analysis task, and more improvements and variants have emerged, as shown in Table 1, mainly including the following. In 2018, Kim et al. [9] proposed a sample-level CNN, while adding artificial noise to the input signal, which was found to be beneficial to the performance of the neural network. Along with the proposal of ASCAD, Benadjila et al. [2] also proposed and applied CNN_best. In 2019, Zaid et al. [11] proposed Efficient CNN, a CNN architecture used to build Efficient CNNs in terms of attack efficiency and network complexity, while using techniques, including weight visualisation, gradient visualisation, and heatmap visualisation to clearly explain each hyperparameter in the feature selection phase. In 2020, Woutert et al. [12] critically corrected and improved the structure proposed by Zaid by reducing the number of parameters by an average of 52% based on an understanding of the inner workings of the model while maintaining similar performance, as well as demonstrating that the size of the convolutional filters does not strictly correlate with the amount of unalignment in the traces. Additionally, increasing the size of the filters and the convolutional number of times can actually improve the performance of the network. In 2022, Cao et al. [14] applied the bilinear CNN to an analytical task, an innovation that also improved on the starting point from work in computer vision by introducing a bilinear convolutional neural network (B-CNN) to perform an effective contour attack against widely used masking countermeasures. The authors of this paper have also further improved the CNN_best variant, specifically by adding an attention mechanism module to it.

Table 1. Comparison of CNN models and their variants.

Model Variant	Structural Features	Performance Limitations	Time
CNNs [9]	Based on VGG-1D, small convolution kernels, and deep stacking structures are emphasized for efficiency	Small sample size, sensitive to overfitting	2018
CNN_best [2]	Optimized based on VGG-16, suitable for analyzing synchronized and desynchronized trace data	Relies on large-scale data, high training complexity	2018
Efficient CNN [11]	Efficient and lightweight CNN, hyperparameter optimization and visualization tool design, low complexity	Limited support for complex masking countermeasures	2019
Revisiting EffCNN [12]	Compared to EffCNN, the first convolutional layer can be omitted by appropriately preprocessing the input data	Dependent on the full sample dataset	2020

Table 1. *Cont.*

Model Variant	Structural Features	Performance Limitations	Time
B-CNN [14]	Bilinear CNN architecture, with two CNN paths extracting features of masked and unmasked values, respectively	Outer product operation increases complexity, limiting performance in certain scenarios	2022
CNN_best + CBAM [15]	Addition of CBAM attention mechanism module in the CNN_best architecture	Small sample, not applicable to real scenarios for the time being	2024

2. Transformer backbone network and variants

The transformer architecture was successfully applied to cryptanalysis tasks within the last two years, as shown in Table 2, mainly by Hajra et al. [17], who proposed the TransNet structure at the 2022 Unclassified Conference by proposing a new translation-invariant transformer model with linear time and memory complexity, which performs well in successfully combating implementations that are protected by both masking and random delay countermeasures. It performs better in terms of performance, but suffers from quadratic time and memory complexity. In the following year, experiments were carried out with an optimized and improved EstraNet [18], which significantly improved the secondary time and memory cost of the transformer model using an attention layer (enhanced relative position information coding) and a novel normalization method, and was very effective against countermeasures, such as random delays and clock jitter. In 2024, Fan et al. [19] applied this TransNet structure with certain hyperparameter tuning applied to the side-channel analysis of the FESH cryptographic algorithm by adding the BlurPool fuzzy downsampling technique, which, combined with the normalization operation, reduced the number of parameters and improved the model performance.

Table 2. Comparison of the transformer model and its variants.

Model Variant	Structural Characteristics	Performance Limitations	Time
TransNet [17]	Problems characterized by high quadratic time and memory complexity	Challenges associated with problems exhibiting high quadratic time and memory complexity	2022
EstraNet [18]	Linear complexity, telative position information, layer-centering normalisation	Limited performance for long-time trajectory processing; model effects validated only on specific datasets (e.g., ASCAD), generalisability needs more validation	2023
Improved TransNet [19]	A transient-based model with fuzzy downsampling (BlurPool) combined with the normalization	BlurPool introduces a high dependency on the blurring characteristics, which may have an impact on the ability to extract detailed signals.	2024

In summary, the motivation for the innovation in DLSCA in this paper comes from going beyond the two existing technology routes of the CNN variant (SOTA) and transformer variant (SOTA) baseline, where the benchmark pursues GE reduction.

3. Methodology

3.1. Fundamental Principle

To achieve improved analysis efficiency and implement higher-order analysis, the introduction of machine learning methods and deep learning algorithm model applications aims to implement a guessing model $\hat{g}$ for key k, to implement the estimation of the conditional probability function calculation, and to construct a mapping relationship from energy profile $\vec{l}$ and plaintext p inputs, to the different labels P or outputs. Firstly, the model $\hat{g}_k$ is built, which can be expressed as follows:

$$\vec{g}_k: (\vec{l}, p) \mapsto P[\vec{L} = \vec{l} \mid P, K = (p, k)] \tag{1}$$

where, due to the presence of an encryption–decryption sensitive operation $\varphi(\cdot)$, denoted as $\varphi(P, K)$, it can be rewritten as follows:

$$\vec{g}_k: (\vec{l}, p) \mapsto P[\vec{L} = \vec{l} \mid \varphi(P, K) = z] \tag{2}$$

The modeling (training) and attack (validation) target datasets obtained after collection and processing generally cover the following:

$$\begin{cases} D_{profiling} = \left\{ \vec{l}_i, p_i, k_i \right\}_{i=1,\ldots,N_p} = \left\{ \vec{l}_i, z_i \right\}_{i=1,\ldots,N_p} \\ D_{train} = \left\{ \vec{l}_i, p_i, k_i \right\}_{i=1,\ldots,N_p} \\ D_{test} = \left\{ \vec{l}_i, p_i, k_i \right\}_{i=1,\ldots,N_p} \\ D_{attack} = \left\{ \vec{l}_i, p_i \right\}_{i=1,\ldots,N_a} \end{cases} \tag{3}$$

For the computation of model probability estimates for (1) and (2), also called prediction problems in deep learning, the most common is the classification problem, i.e., building, as follows:

$$\vec{g}_k: (\vec{l}, p) \mapsto P[P, K = (p, k) \mid \vec{L} = \vec{l}] \tag{4}$$

Predictive and classification models are often linked and the two can be derived from each other, expressed as

$$P[(P, K) = (p, k) \mid \vec{L}] \times f_{\vec{L}} = P[\vec{L} \mid P, K = (p, k)] \times f_{(P, K)}(p, k) \tag{5}$$

Combined with the advantages of deep neural networks in solving classification problems, the tendency is to directly construct approximate models as follows:

$$\vec{g}_k : (\vec{l}, p) \mapsto P\left[P, K = (p, k) \,\middle|\, \vec{L} = \vec{l}\right]_{k \in \mathcal{K}} \tag{6}$$

Applied to the guess classification of the new curve leakage $\vec{l}$ candidate key obtained by inputting a new p observation collection in the attack set D_{attack}, t the key discrimination

classification can then be performed by processing $\vec{y} = g_{\vec{L},P}(\vec{l}, p)$ and the key candidate key $\hat{k} = \text{argmax}_k \vec{y}[k]$. The key discrimination classification of the curve plaintexts of the whole attack set $(\vec{l}_i, p_i)_{i=1,\ldots,N_a}$ can be performed by adopting the strategy of great likelihood estimation by introducing the key score vector $\vec{d}_{N_a}[k]$, which can be computed as follows:

$$\vec{d}_{N_a}[k] = \prod_{i=1}^{N_a} \vec{y}_i[k] \tag{7}$$

$$\vec{y}_i[k] = g_{\vec{L},P}\left(\vec{l}_i, p_i\right)[k] \tag{8}$$

Then, the score vector calculation can be expanded as

$$\vec{d}_{N_a}[k] = \prod_{i=1}^{N_a} P\left[(P,K) = (p_i,k) \middle| \vec{L} = \vec{l}_i\right] = \prod_{i=1}^{N_a} \frac{P\left[(P,K) = (p_i,k) \middle| \vec{L} = \vec{l}_i\right]}{f_{\vec{L}}(\vec{l}_i)} \times f_{(P,K)}(p_i,k) \tag{9}$$

where $\vec{y}_i$ represents the output of the model and the kth coordinate of the score vector is the score corresponding to the key candidate key. The model outputs are accumulated to obtain the scores of the key candidates and form a vector of great likelihood estimates.

Due to the presence of the labeling of the sensitive operation $\varphi(\cdot)$, the classification model can then be further labeled as follows:

$$\vec{g}_{\vec{L},P} : (\vec{l}, p) \mapsto P[\varphi(p, K) = z \mid \vec{L} = \vec{l}]_{z \in \text{Im}(\varphi)} \tag{10}$$

where A is the set of two-dimensional vectors integrating the processed operation-specific vectors. Therefore, output C of model B can be indexed by D to create a second output for the key candidate key as follows:

$$\vec{y}[k] = y'[\varphi(p, k)],\ k \in K \tag{11}$$

Combined with the above computational representation, it can be further clarified that in the context of modeling class side-channel analysis, the application of DL-SCA is uses multiple curves l for modeling and training. After the set of attack curves is obtained as a result, the attack curves and the key guessing probabilities can form $\left\{\vec{l}_i, y\right\}_{i=0}^{T_a-1}$, in which the key guessing probabilities find the following target guessing key:

$$\hat{k} = \text{argmax}_k \hat{\delta}_k \tag{12}$$

$$\hat{\delta}_k = \sum_{i=0}^{Ta-1} \log\ s_i[F(yi, k)] \tag{13}$$

$$y = f(l; \theta^*),\ y \in \mathbb{R}^{|\mathcal{Z}|} \tag{14}$$

where $y[i], i = 0, \ldots, |\mathcal{Z}| - 1$ represents the guessed probability of the intermediate variable $Z = i$ and θ^* is the hyperparameters of the deep learning network model used for training.

Previous research [4,12,14,20] provided a list type of parameter statements. To be able to describe a deep learning neural network algorithm model more concisely and clearly, this paper suggests specifying its structural parameters (number of layers, number of nodes or neurons and activation function, etc.) and hyper-parameters (number of training rounds, batch size, optimizer, learning rate, etc.), denoted as follows:

$$\hat{g} = \begin{cases} \text{Network}\ (n_{\text{layer}},\ n_{\text{units}},\ \text{ACT},\ ./n_{\text{blocks}},\ n_{\text{dense}}\ ,\ n_{\text{filter}},\ \text{ACT},\ \text{et al.}) \\ \theta^*{}_{\text{Network}}(n_{\text{epoch}},\ \text{batch_size, Optimizer, learning_rate, dropout,}\ \text{et al.}) \end{cases} \tag{15}$$

3.2. Existing Methodology

3.2.1. CNN Class Network Model

In this section, we summarise and collate the existing CNN models and their variants improved by previous authors, so as to further compare and sort out the optimization ideas. The parametric expression of the SOTA method in the form of Equation (15) leads to a description of the CNN variant model as shown in Table 3.

Table 3. CNN models and their variants' structural parameters and hyperparameters.

	Structural Parameters and Hyperparameters
1 [2]	CNN_base $\{5[1(64, 3), 5(2, 2), \text{ReLU}], 2, \text{SOFT})\}$ $\theta^*_{\text{CNN_base}}(>130, 100, \text{RMSProp}, 10^{-5})$
2 [2]	CNN_best $\{5[1(64\,\vert 128\vert 256\vert 512, 11), 52, 2, \text{ReLU}], 2, \text{SOFT}\}$ $\theta^*_{\text{CNN_best}}[100(75), 200, \text{RMSProp}, 10^{-5}]$
3 [11]	EffCNN$\{5[2(64, 3*3), 3(2, 2), \text{SeLU}], 1, \text{SOFT}\}$ $\theta^*_{\text{EffCNN}}(50, 256, \text{Adam}, 10^{-3})$
4 [12]	SimplifiedEffCNN$\{2/3/4[2(64/128/256, 5*5), (2,2), \text{SeLU}], 1, \text{SOFT}\}$ $\theta^*_{\text{SimplifiedEffCNN}}(50, 256, \text{Adam}, 10^{-3})$
5 [16]	DilatedCNN$\{5[3(32\vert 64\vert 128, 1\text{k}1\vert 25\vert 3), (2, 2), \text{SeLU}], 3, \text{SOFT}\}$ $\theta^*_{\text{DilatedCNN}}(75, 256, \text{Adam}, 10^{-3})$
6 [14]	BCNN$\{3[3(4, 100), (2, 10), \text{SeLU}], 20, \text{SOFT}\}$ $\theta^*_{\text{BCNN}}(50, 600, \text{Adam}, 5\times10^{-4}-5\times10^{-3})$

3.2.2. TransNet Network Model

Analogously to MLPs and CNNs, a general expression for the transformer can be established as follows:

$$\vec{g}_{\text{Transformer}}: \vec{l} \mapsto \hat{g}\left(\vec{l}\right) = s \circ \varphi^{n_4} \circ [\alpha \circ \mu^{n_3}_{Q,K,V} \circ \rho^{n_2} \circ \varepsilon(\vec{l}_n)] = \vec{y} \tag{16}$$

The parametric expression of the SOTA method, also in the form of Equation (15), yields a description of the transformer variant model as shown in Table 4.

Table 4. Transformer model variant structural parameters and hyperparameters.

	Structural Parameters and Hyperparameters
1 [17]	TransNet$(1(\text{ReLU}(500, 11)), 6(\text{PE}, \text{MHA}, 10, 50, 1000), 500, 1, \text{SOFT})$ $\theta^*_{\text{TransNet}}(391, 256, \text{Adam}, 2.5*10^{-4}, 0.1)$
2 [18]	EstraNet$(1(\text{ReLU}\ (500, 11)), 6(\text{PF}, \text{SA}, 4, 32, 256), 128, \text{SOFTA}, 1, \text{SOFT})$ $\theta^*_{\text{EstraNet}}(391, 256, \text{Adam}, 2.5*10^{-4}, 0.1)$
3 [19]	Improved TransNet$\{10[\text{ReLU}(60, 3)], 2(\text{PE}, 2, 32, 256), \text{SOFT}\}$ $\theta^*_{\text{Improved TransNet}}(30000, 256, \text{Adam}, 2.5*10^{-4}, 0.004)$

3.3. Proposed Structural Framework

In order to achieve a more efficient side-channel analysis for specific target leakage models, this section proposes a new variant of the CNN method based on Simplified EffCNN using the idea of generative networks, exploring whether it is possible to migrate and apply it to the analysis of the real-life scenarios introduced in this paper. The main structure of the model is shown in Figure 1.

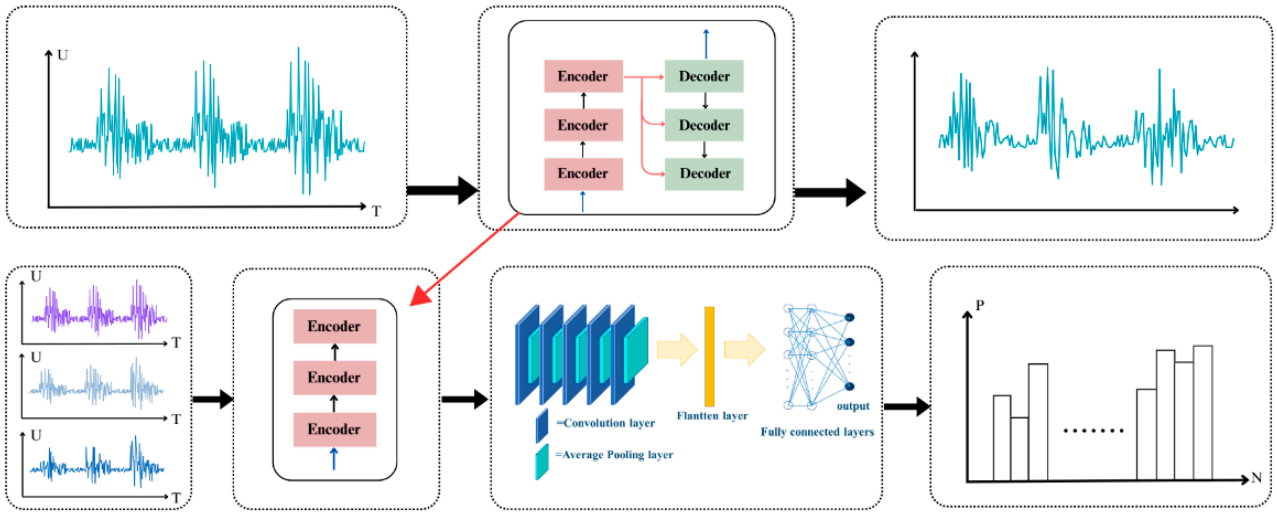


Figure 1. Bi-CNN Codec analysis model structure.

The design idea of the proposed method is characterized by 2 main aspects:

(1) Structural design aspect—To achieve the consistency of input and output, the first set of coder–decoder is constructed so that the network model understands the target leakage features, the feature head encoder is trained first, and then the encoder parameters are fixed for the second step of the network. To achieve the focus on the target features, a feature head is added in front of the CNN backbone network encoder.

(2) Analysis of the implementation aspects—The first step allows the model to generate the point traces again through the point traces, which enables the model to learn the features of the point traces. This is part of the idea borrowed from Stable Diffusion. The second step is to take the encoder that has been learned in the first step and provide it to the main CNN model.

The model structure parameters and training hyperparameters of the proposed Encoder Feature CNN (EFCNN) method are summarised in comparison with the SOTA method as shown. To be able to optimize the continuity design of the SOTA method, as well as exploring and resolving the effect of the full sample size of the data, detailed experiments will be carried out in this paper for evaluation.

3.4. Preprocessing

To be able to train the neural network to ensure its fast convergence, pre-processing of the input data is generally chosen. Wouters et al. [12] explored scaling between 0 and 1, scaling between −1 and 1, and data normalization pre-processing practices based on zero-mean-unit-variance to validate that removing the first layer of the convolution achieves the same results with reduced model parameters. In this paper, six preprocessing control groups are set up for the above three preprocessing function implementations at the feature scale and spatial scale, respectively, against the previous practices: feature_scaling_0_1, feature_scaling_m1_1, feature_standardization, horizontal_scaling_0_1, horizontal_scaling_m1_1, and horizontal_standardization.

3.5. Loss Function

In training, a loss function calculation is introduced to evaluate the degree of inconsistency between the predicted and true values of the training model. Cross-entropy loss

(CE-LOSS) can be used for outputting the predicted label y_{ij} or the predicted probability $\hat{y}_{ij}$ and the true label or the true value deviation until convergence as shown in Equation (17)

$$\mathrm{Loss}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\log(\hat{y}_{ij}) \tag{17}$$

4. Performance Evaluation

In this section, a comparative evaluation will be carried out using the experimental setup of 3 (methods) * 4 (datasets) * 7 (preprocessing) * 3 (data enhancement). The four public datasets used in the experiments are first described, followed by an overview of the evaluation metrics, and the results of the advanced and proposed methods on the datasets are analyzed and discussed. Next, an ablation study is provided. Finally, quantitative conclusions are drawn about the excellent analytical generalization performance of CNNs and transformer variant structures across datasets.

4.1. Datasets

In order to be consistent with previous studies, the experimental subjects chosen in this paper are the DPA_v4.1 [21], AES_RD [22], AES_HD [23,24], and ASCAD_f [25] datasets, and the detailed information of specific concern is shown in Table 5.

Table 5. Benchmarks for comparison of self-collected datasets.

	Targeted Chip Devices	Cryptographic	Type of Analysis	Name of Dataset	Traces (Features)	Time
1	SASEBO-W (FPGA) (Software)	AES-256 (RSM)	Electromagnetic	DPAcontest_v4.1 [21]	100,000 5000 (4000)	2014
2	8 bit Atmel AVR (Software)	AES-128 (Masked)	power consumption	AES_RD [22]	50,000 (3500)	2009
3	SASEBO-GII (FPGA)	AES-128	power consumption Electromagnetic	AES_HD [23,24]	50,000 (1250)	2018
4	ATMega (Software)	AES-128	power consumption Electromagnetic	ASCADf [25]	60,000 (100,000)	2018

4.2. Evaluation Method

In the application of the deep learning method, combined with the division of the dataset, the experiments establish the rank function to rank the candidate guessing key scores, as shown in Equation (18):

$$\mathrm{rank}(\vec{g}_k, D_{train}, D_{test}, n) = \left| \left\{ k \in K \middle| \vec{d}_n[k] > \vec{d}_n[k*] \right\} \right|, k* \in K \tag{18}$$

If the experiment is performed on target datasets for key ranking computation, for the overall rank, the expectation can be solved to be used as the total rank as shown in Equation (19):

$$\mathrm{Rank}_{N_{train},n}\left(\vec{g}_k\right) = E[\mathrm{rank}(\vec{g}_k, D_{train}, D_{test}, n)] \tag{19}$$

Since the guessing accuracy corresponds to the correctness of the key extracted candidates, researchers majoring in DL-SCA algorithms still adopt the guessing accuracy metric.

The calculation of the model's training and testing classification accuracy on the dataset itself can be shown in Equation (20):

$$\text{Acc.} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{20}$$

where TP stands for true case, TN for true negative case, FP for false positive case, and FN for false negative case. Such a calculation expression is still relatively abstract, and the concrete application can be expressed by the formula:

$$\text{acc}\left(\vec{g}_k,\ D_{\text{train}},\ D_{\text{test}}\right) = \frac{\left|\left\{\vec{l}_i, p_i, k* \in D_{\text{test}} \middle| \hat{k} = \text{argmax}_k \vec{y}_i[k]\right\}\right|}{|D_{\text{test}}|} \tag{21}$$

If the experiment is performed on target datasets to validate the classification guessing accuracy results, for the overall accuracy, the expectation can be solved to serve as the total ACC as shown in Equation (22):

$$\text{ACC}_{N_{\text{train}}}\left(\vec{g}_k\right) = \text{E}[\text{acc}(\vec{g}_k, D_{\text{train}}, D_{\text{test}})] \tag{22}$$

For each key guess k_i, a probability of guessing is assigned to its correctness, which can also be used as a weight for the overall key guess. Guess entropy is defined as the mean ranks of the correct key k^* in vector $\vec{g}_k$ of experimental key guesses that are repeatedly performed N times in an attack on the Q-bar power consumption curve. More precisely, the GE metric characterizes the average number of power analysis attacks required to obtain the correct key k^*, as shown in Equation (23)

$$\text{rank}(K|X) = \textstyle\sum_{i|i \neq k^*} P[\text{pos}_{k_i} < \text{pos}_{k^*}] \tag{23}$$

$$\text{GE} = \frac{1}{N}\sum_{q=1}^{N} \text{rank}(K|X = X_q) \tag{24}$$

where rank represents the position of the correct key in the ranked probability histogram, N is the number of experiments, k_* is the correct key, and $P\left[\text{pos}_{k_i} < \text{pos}_{k_*}\right]$ represents the probability that other keys are more likely than the correct key.

In most of the side-channel analysis scenarios, the key file is the key quantity used in the power SCA process, the key guessing accuracy is the metric that the method focuses on, and the guessing entropy GE is and will be the basis for evaluating the efficiency of the target device facing power SCA.

4.3. Experimental Setup

All the experimental methods in this paper are implemented in Python 3.7.12 using the Tensorflow 2.10.1 framework. The models are trained in distributed data parallelism, can be run on single/multiple machines, and use half-precision tensors to reduce training time. The experiments are run on an arithmetic server equipped with 4-card GeForce RTX 4090 GPUs. The training model hyperparameters are as follows:

$$\hat{g} = \begin{cases} \text{EFCNN}\{2/3/4[\text{Encode}\,2 + 2(64/128/256, 5*5), (2,2), \text{SeLU}], 1, \text{SOFT}\} \\ \theta^*{}_{\text{EFCNN}}(50, 256, \text{Adam}, 10^{-3}) \end{cases} \tag{25}$$

4.4. Comparison with State-of-the-Art Methods

The state-of-the-art methods compared in this paper are Efficient CNN by Zaid et al. [11] and Simplified Efficient CNN by Wouters et al. [12]. Based on the structural fea-

tures, the experiments and results are named zaid and noConv1 in the legend, respectively. A residual structure-based ResNet network (named improved_resnet) and the proposed feature-coded CNN network (named autoencoder_classifier) are used to compare and analyse the four methods to guess the entropy results on four datasets and the number of attack curves, $Nt_{GE}\{1,3,5,10\}$ required to reach four threshold constants, GE.

Firstly, the focus is visual analysis to compare the effects of four methods * seven preprocessing operations * and whether data enhancement operations are performed on the guessed entropy results. The ablation comparison results are presented in two result plots for each dataset; the first one contains 42 guessed entropy curves for the first three methods, and the latter one includes 14 guessed entropy curves for the methods proposed in this paper, as shown in Figures 2–9.

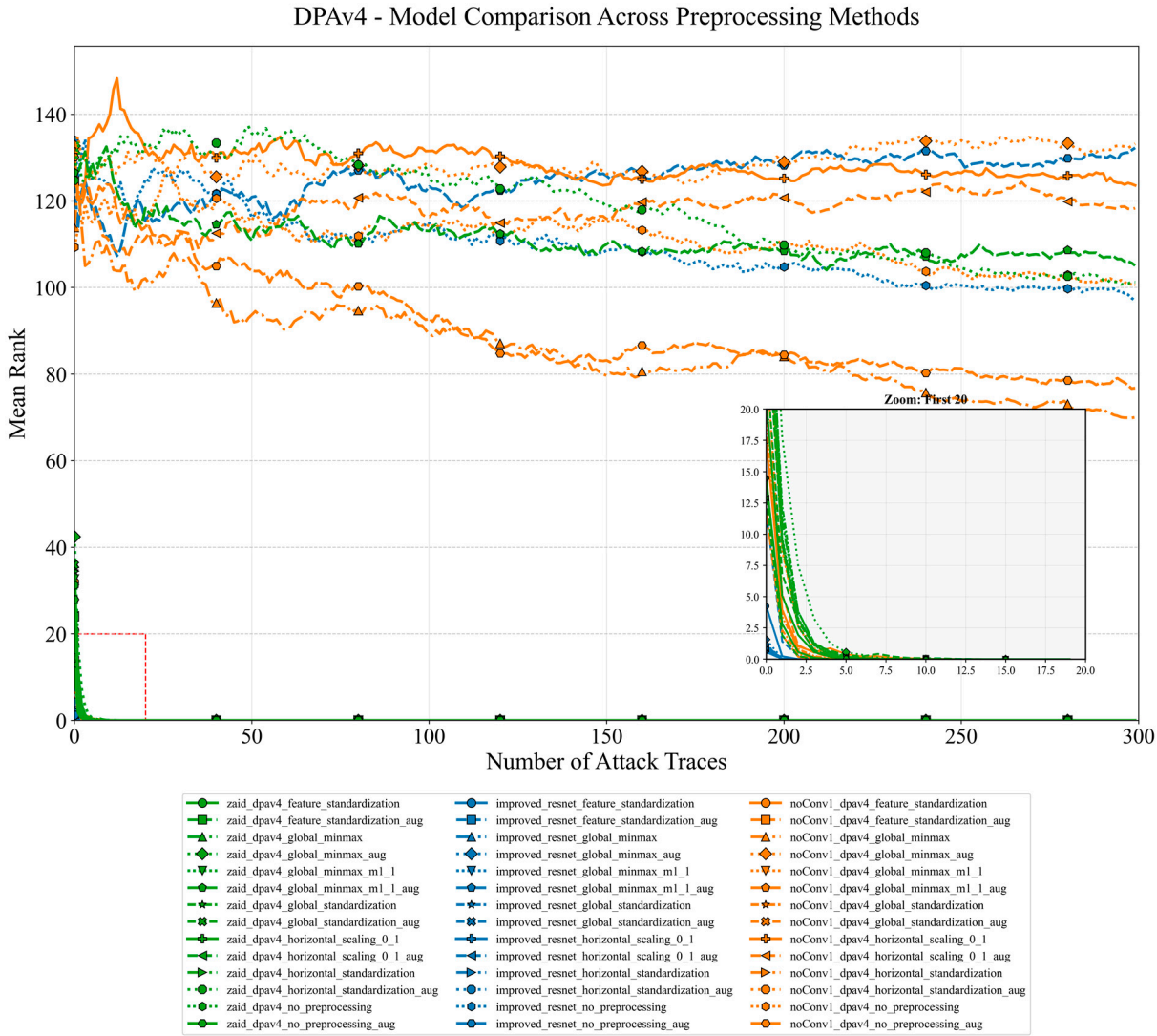


Figure 2. Variation in the average rank of the correct key hypothesis for three methods on the DPA_v4 dataset as the number of traces used for the attack increases.

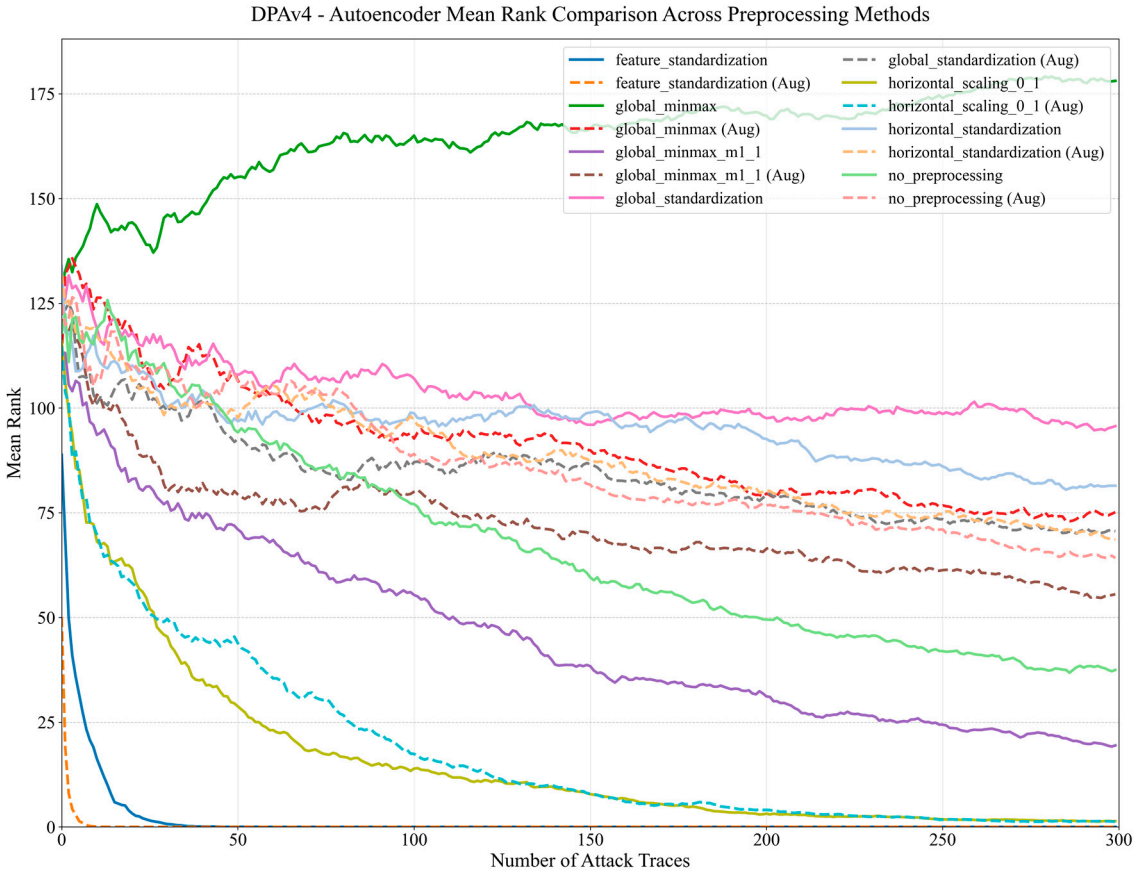


Figure 3. Variation in the average rank of the correct key hypothesis for feature encoding CNN on the DPA_v4 dataset as the number of traces used for the attack increases.

Figure 2 demonstrates that nearly three-quarters of the number of the three methods with added preprocessing and data enhancement operations achieve the target classification results on the DPA_v4 dataset. Among them, Simplified Efficient CNN fails more. This validation result is surprising; it may be because the number of its parameters is too small, resulting in insufficient sensitivity to the feature recognition of data preprocessing and data enhancement. A comparison of Figure 3 shows that the ResNet network based on residual structure (named improved_resnet) and the feature encoding CNN network (named autoencoder_classifier) proposed in this paper have the best overall results. The effective method achieves the number of curves used to achieve the average rank of the correct key assumption in the course of responding to the 300 validation attacks is less than 10, demonstrating that preprocessing and data augmentation does have a positive effect on the optimization of high-parametric quantitative models, with feature normalization having the most significant effect.

Figure 4 illustrates the results of the target classification achieved by the three methods, adding preprocessing and data enhancement operations on the AES_RD dataset. Among them, most of the preprocessing and data enhancement operations fail for the ResNet network based on residual structure (named improved_resnet), and only local normalization and global normalization (with data enhancement) achieve effective classification. The autoencoder_classifier fails to achieve the target guessing results well. The reason is that this dataset is a guarded AES-128 leaked dataset, which serves as a control group for the generic model, and no specialized parameter adaptation was performed, but the ResNet network based on the residual structure (named improved_resnet) is partially effective. Parameter adaptation for this type of parameter will be specifically compared in the next work.

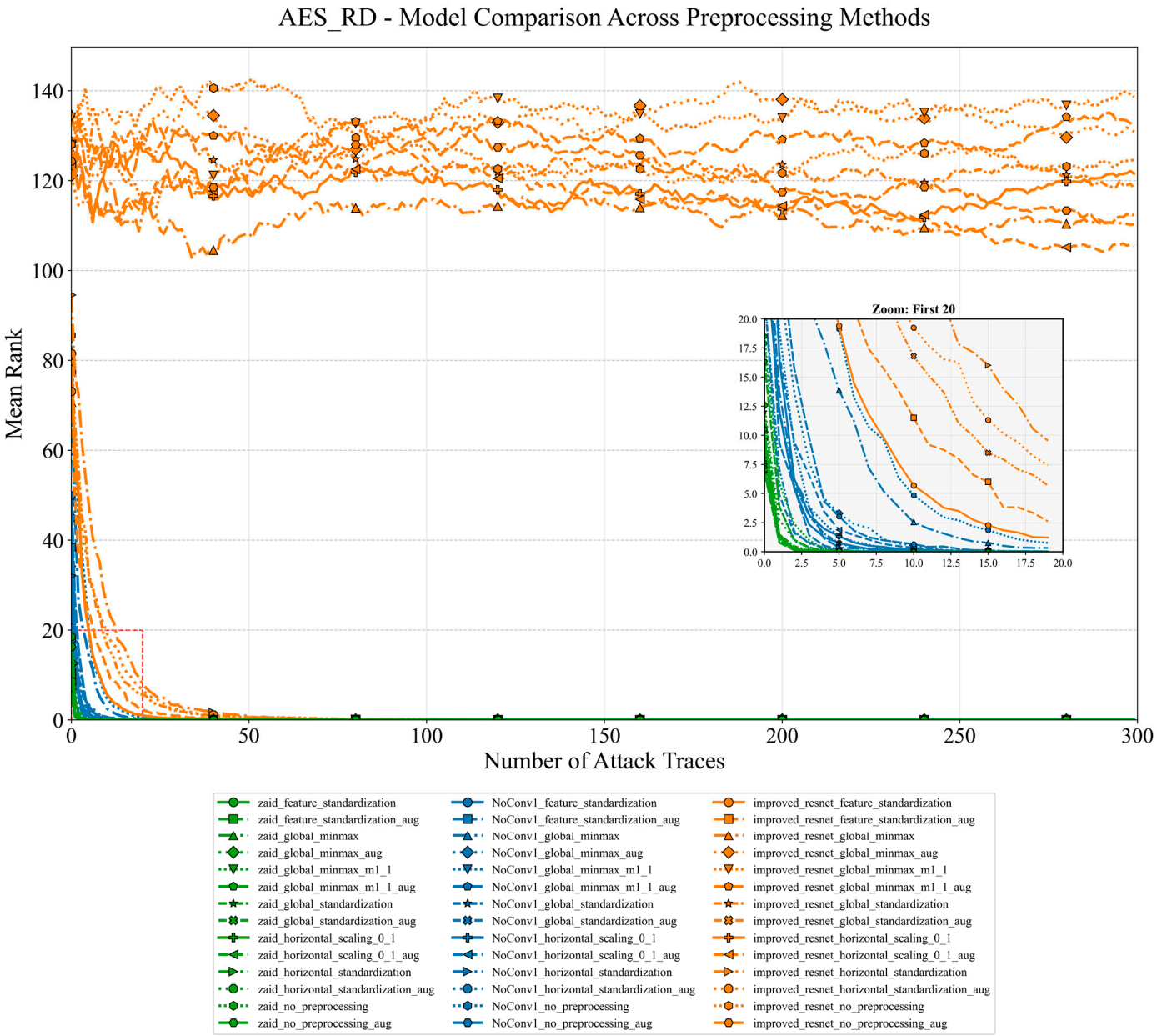


Figure 4. Variation in the average rank of the correct key hypothesis for three methods on the AES_RD dataset as the number of traces used for the attack increases.

As shown in Figure 6, approximately half of the target classification results on the AES_HD dataset are not achieved by the three methods, even with the application of preprocessing and data enhancement techniques. This outcome is primarily attributed to the high noise level inherent in the dataset. Although the target guessing entropy values are not fully reached, the improved_resnet method demonstrates performance comparable to Zaid and noConv1, thereby validating the effectiveness of the proposed structural improvements. These results indicate that further targeted parameter tuning for the autoencoder_classifier is necessary to enhance its robustness against noisy environments.

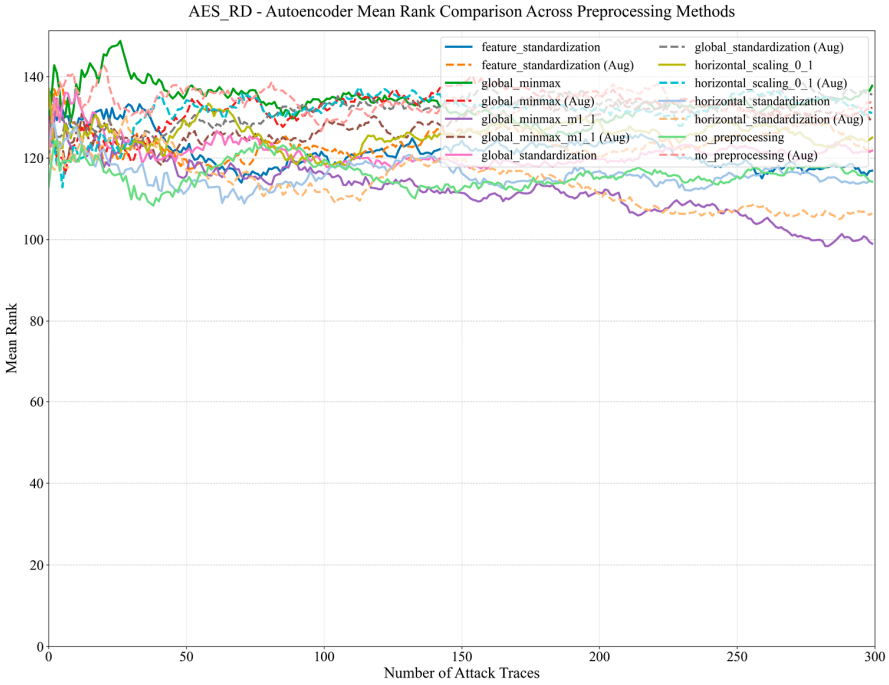


Figure 5. Variation in the average rank of the correct key hypothesis for feature encoding CNN on the AES_RD dataset as the number of traces used for the attack increases.

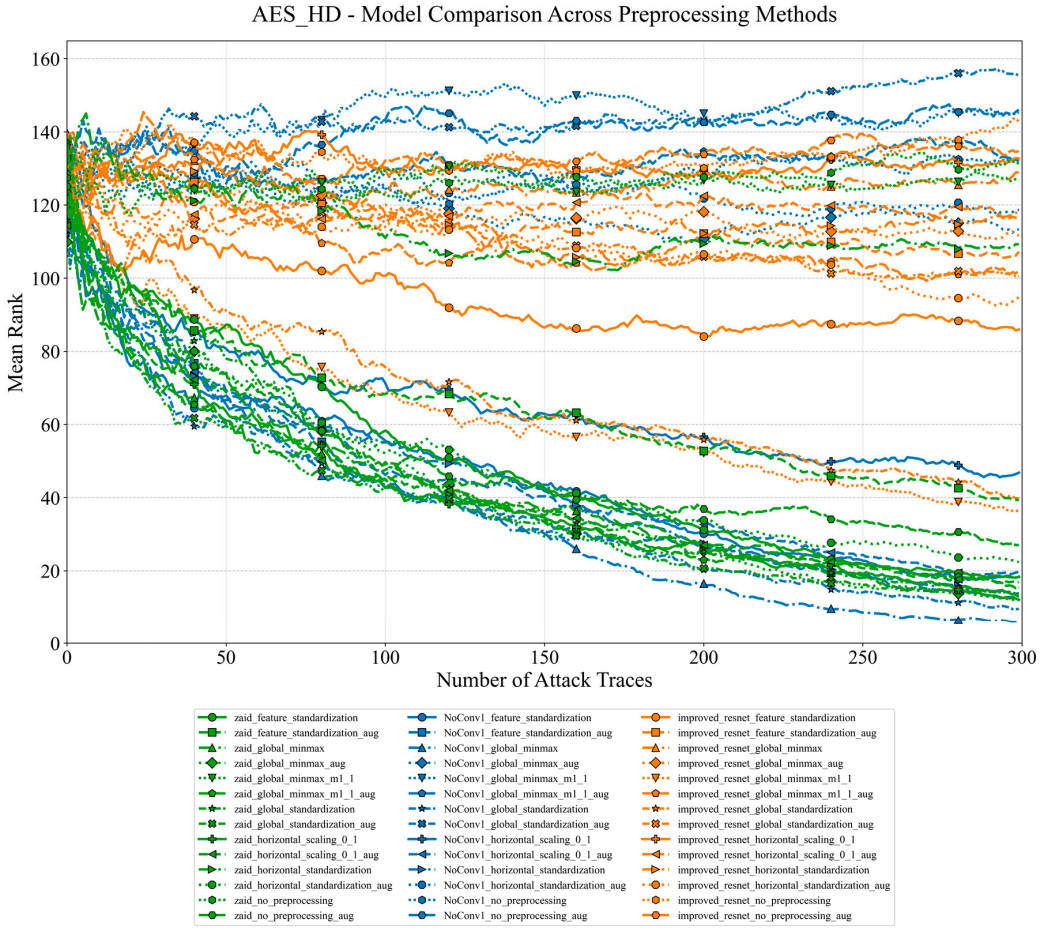


Figure 6. Variation in the average rank of the correct key hypothesis for three methods on the AES_HD dataset as the number of traces used for the attack increases.

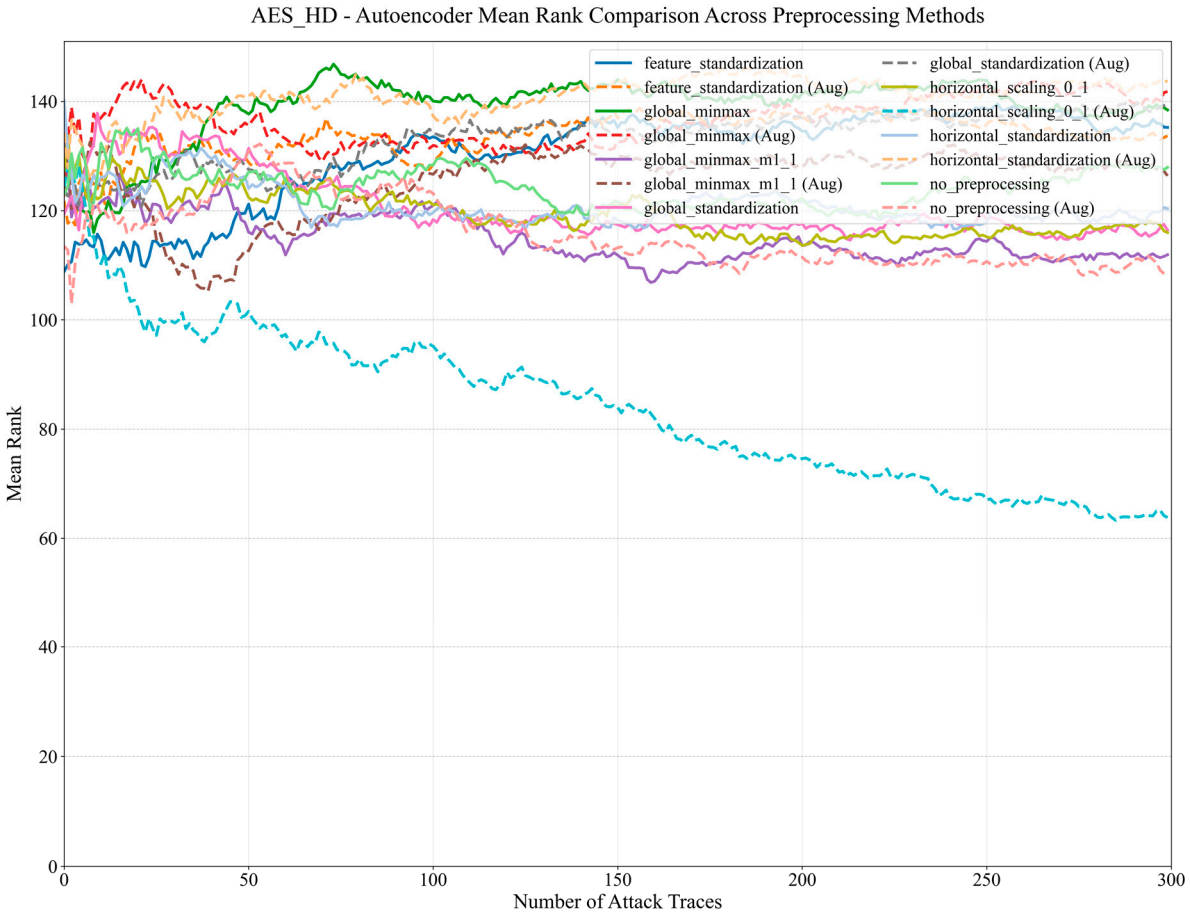


Figure 7. Variation in the average rank of the correct key hypothesis for feature encoding CNN on the AES_HD dataset as the number of traces used for the attack increases.

The results presented in Figure 8 illustrate the target classification performance of the three methods with preprocessing and data enhancement applied to the ASCAD dataset. Among them, the improved_resnet consistently outperforms the other methods across various settings. This finding provides valuable guidance for future tuning of the autoencoder_classifier, particularly in enhancing its effectiveness against masking countermeasures in general-purpose target implementations.

Secondly, the comparison results of the four modeling methods combined with seven preprocessing operations and the application of data enhancement, evaluated under the NtGE thresholds {1, 3, 5, 10}, are presented in Figures 10–15. The ideal target for each threshold setting is to achieve a value of 1. Experimental results demonstrate that on the DPA_v4 and ASCAD datasets, the proposed methods show clear advantages, with a significant reduction in the number of traces required to meet the thresholds. In particular, feature normalization and data enhancement consistently contribute the most to performance improvements across these datasets. In contrast, for the AES_RD and AES_HD datasets, which involve strong masking countermeasures and high noise levels, respectively, the methods proposed in this paper have not yet achieved optimal results. This indicates that further refinement of model parameters and architecture may be necessary to address more challenging protected or noisy scenarios.

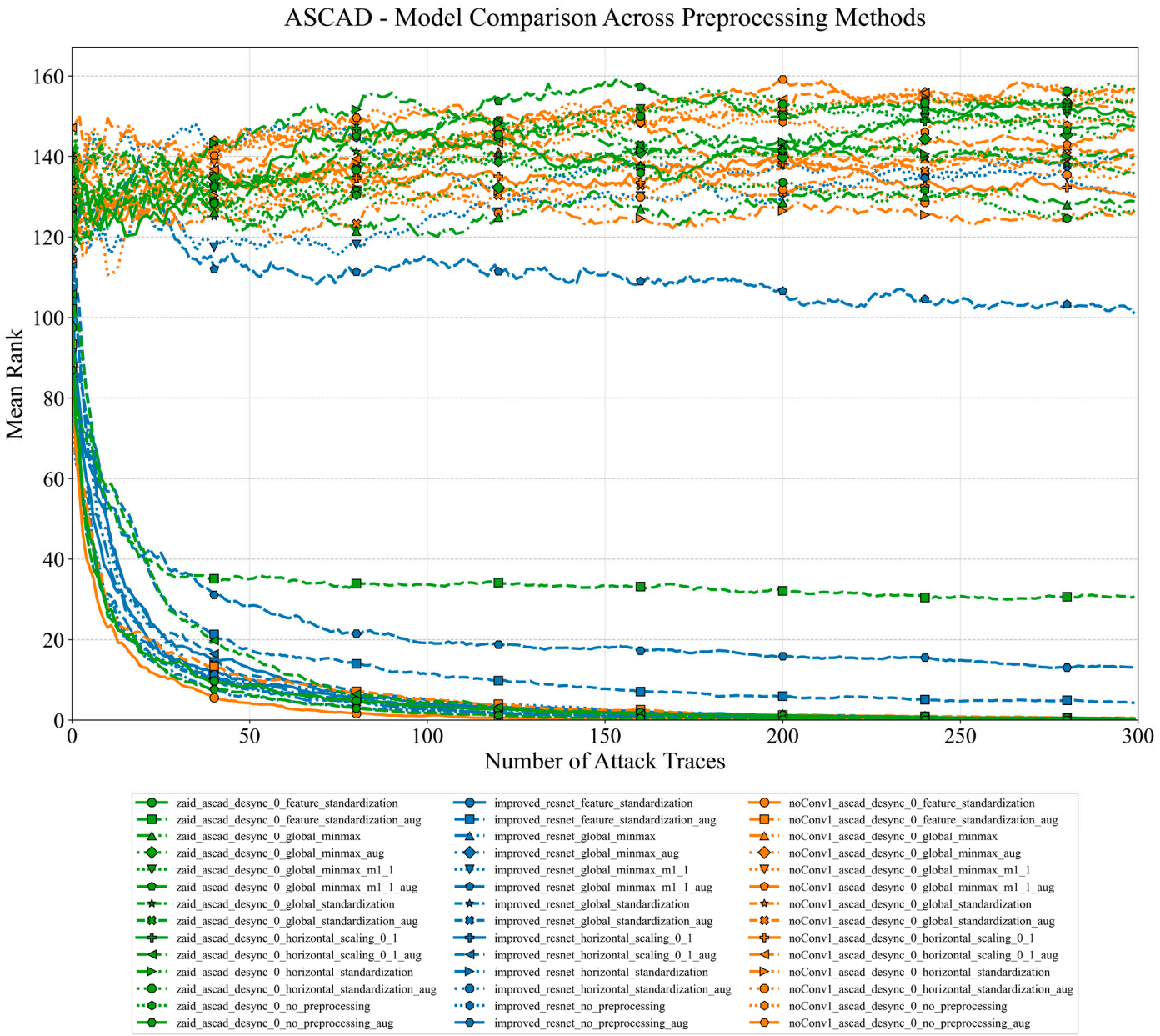


Figure 8. Variation in the average rank of the correct key hypothesis for three methods on the ASCAD dataset as the number of traces used for the attack increases.

Across the four evaluated datasets, the proposed EFCNN exhibits superior performance on DPA_v4.1 and ASCAD_f, achieving GE = 0 with fewer than 10 traces in most settings, compared to 30–50 traces typically required by Efficient CNN and Simplified Efficient CNN. This highlights EFCNN's strong capability in extracting informative features from unprotected or moderately noisy data. However, on the AES_RD dataset—which involves first-order masking countermeasures—and the AES_HD dataset characterized by high environmental noise, EFCNN fails to consistently achieve successful attacks (GE does not converge within 200 traces under most configurations). In contrast, the improved ResNet model demonstrates greater robustness in these two challenging scenarios, successfully achieving classification under specific preprocessing settings, such as global normalization. This suggests that while EFCNN excels in clean and standard leakage settings, improved ResNet may be a more suitable architecture for scenarios involving strong masking or high noise levels. The complementary behavior of the two models provides a practical path forward for generalized side-channel analysis under diverse conditions.

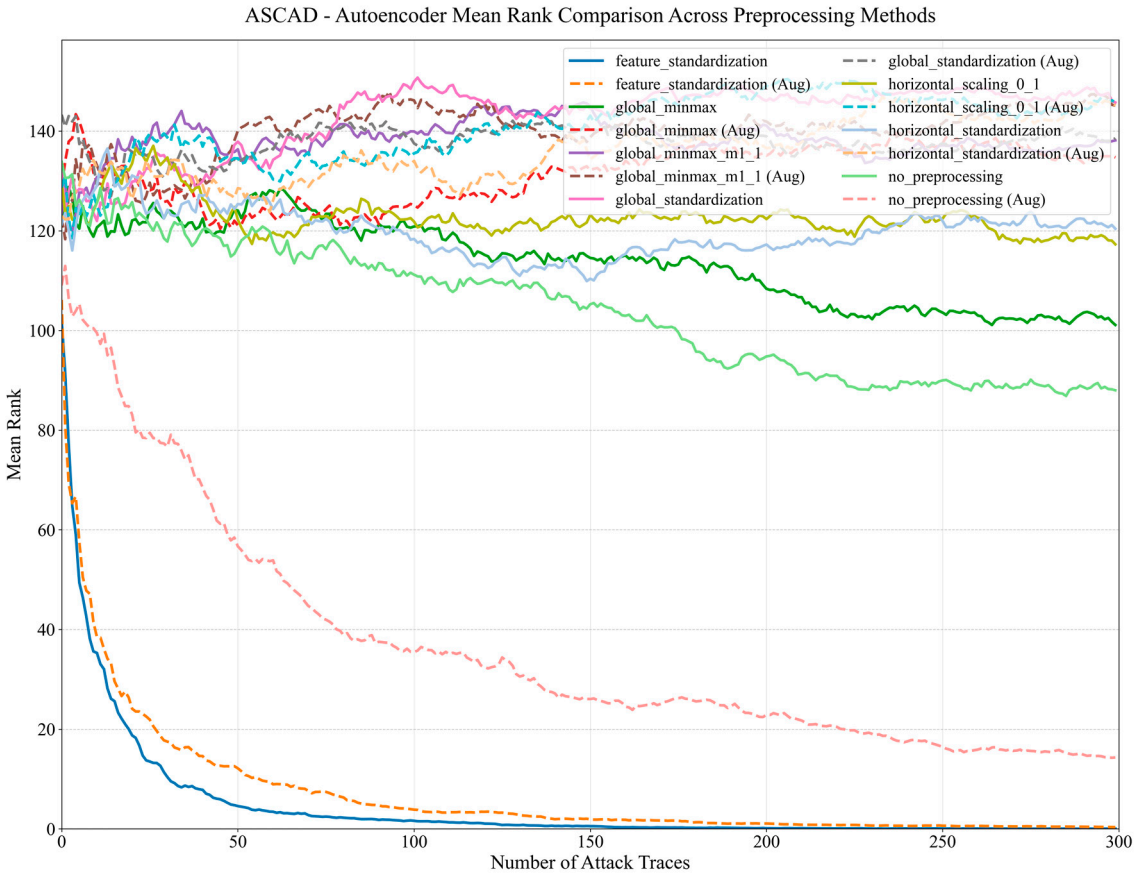


Figure 9. Variation in the average rank of the correct key hypothesis for feature encoding CNN on the ASCAD dataset as the number of traces used for the attack increases.

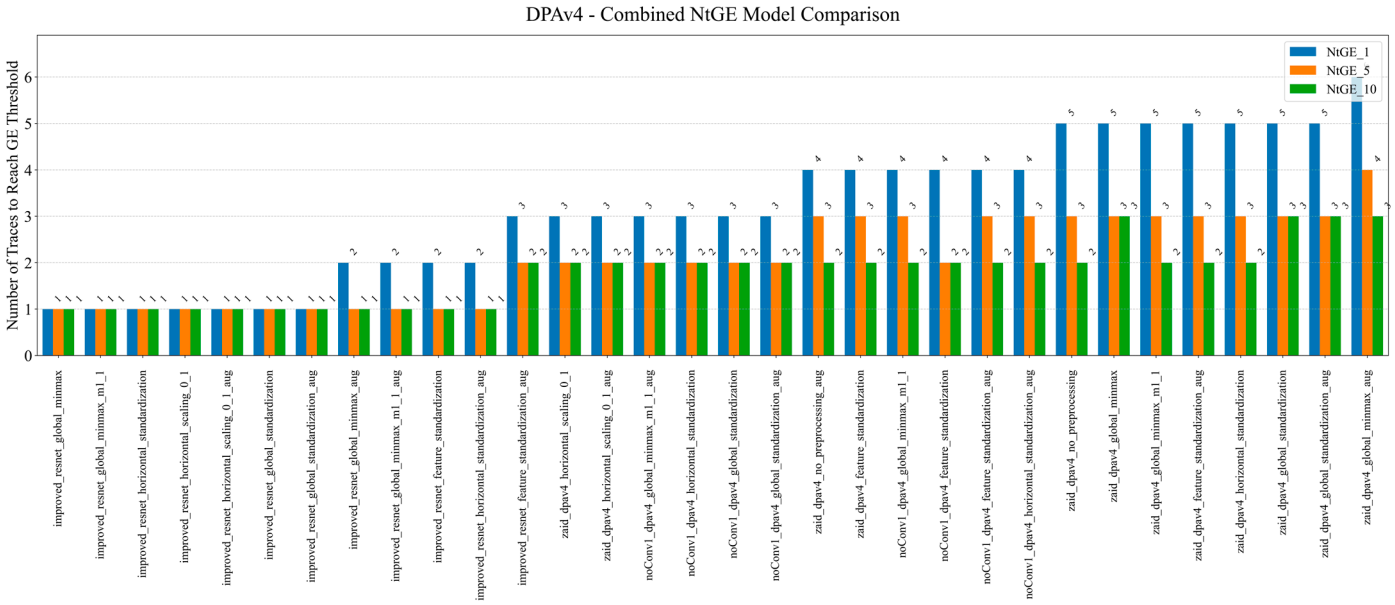


Figure 10. Number of traces required for three methods to reach the Nt_{GE} threshold on the DPA_v4 dataset.

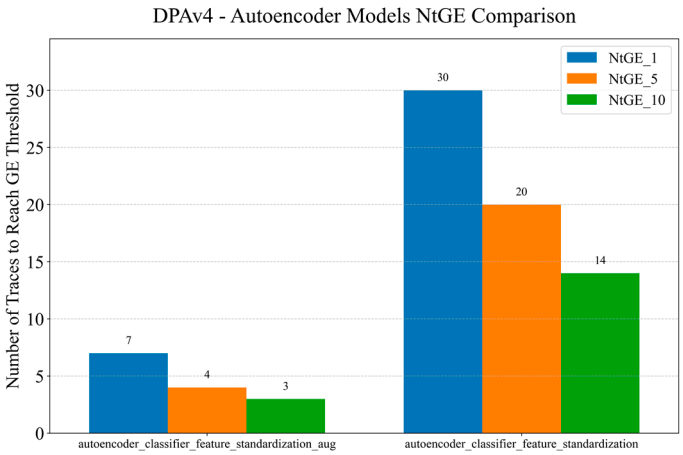


Figure 11. Number of traces required for feature encoding CNN to reach the Nt_{GE} threshold on the DPA_v4 dataset.

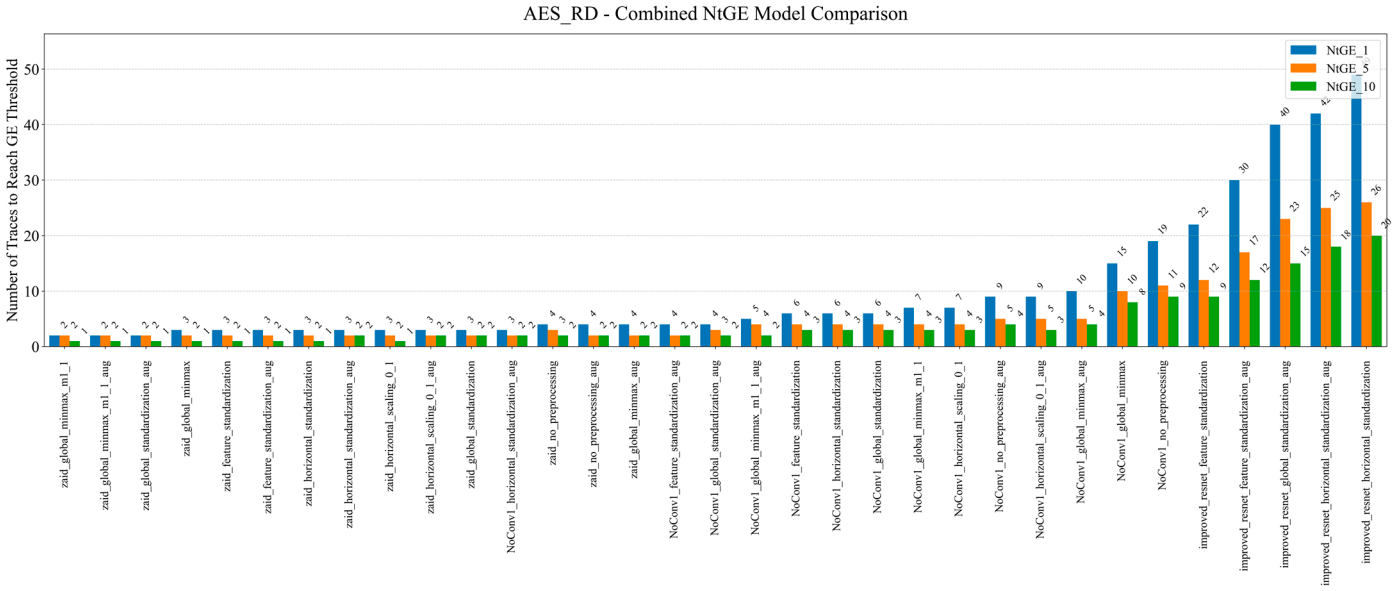


Figure 12. Number of traces required for three methods to reach the Nt_{GE} threshold on the AES_RD dataset.

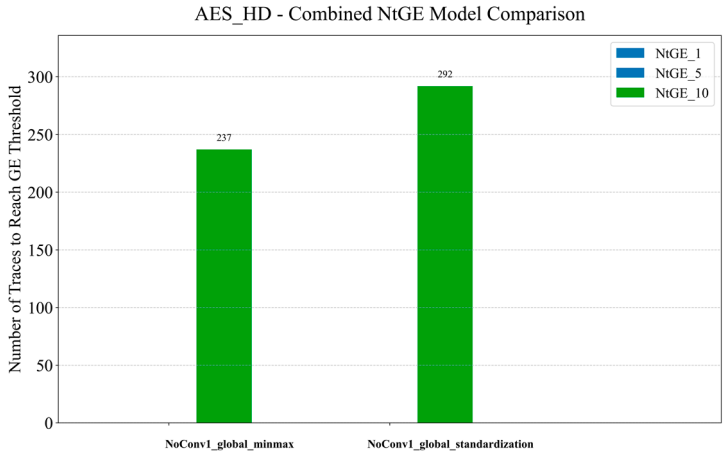


Figure 13. Number of traces required for three methods to reach the Nt_{GE} threshold on the AES_HD dataset.

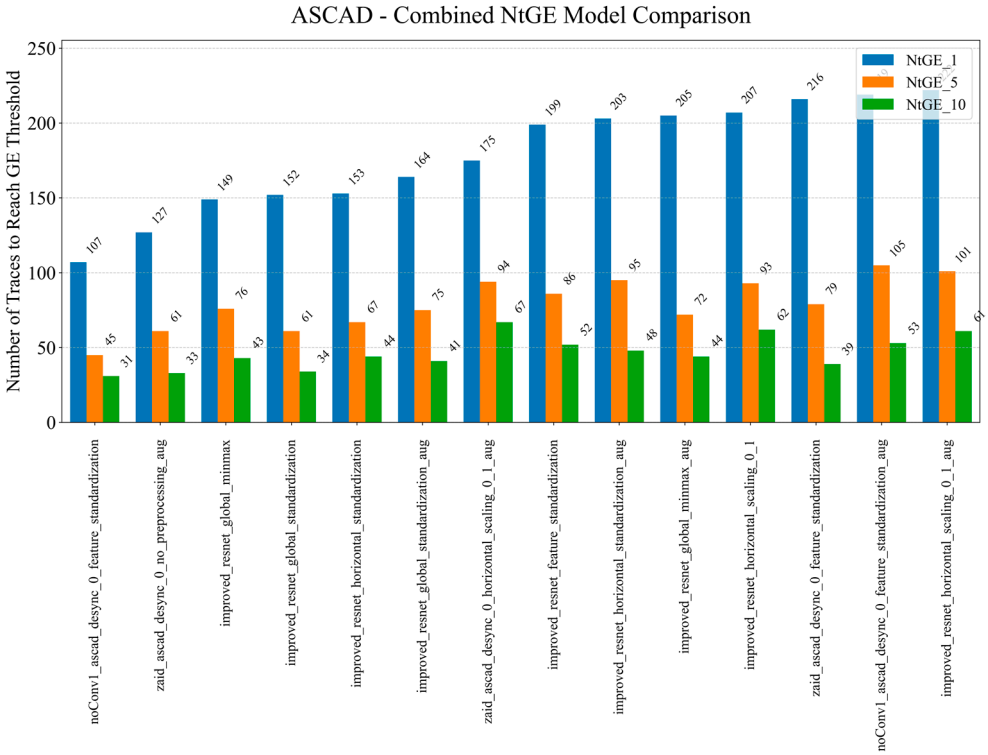


Figure 14. Number of traces required for three methods to reach the Nt_{GE} threshold on the ASCAD dataset.

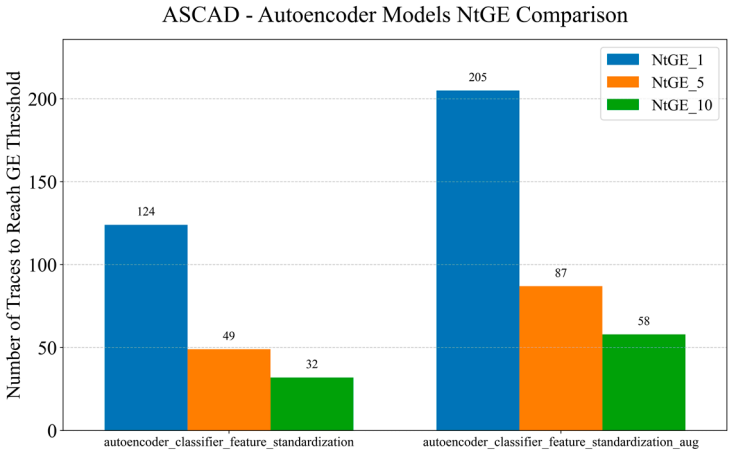


Figure 15. Number of traces required for feature encoding CNN to reach the Nt_{GE} threshold on the ASCAD dataset.

4.5. Cross-Dataset Evaluation

Through the extensive experiments conducted in this paper, an optimal model approach with preprocessing and data augmentation for key guessing is matched from zaid, noConv1, and improved_resnet for each dataset, as shown in Figure 16. The dataset adaptation of autoencoder_classifier is also preferred and explored, as shown in Figure 17. It facilitates the selection and comparison of subsequent experiments, and provides a research basis and a way to compare the generalization of the model.

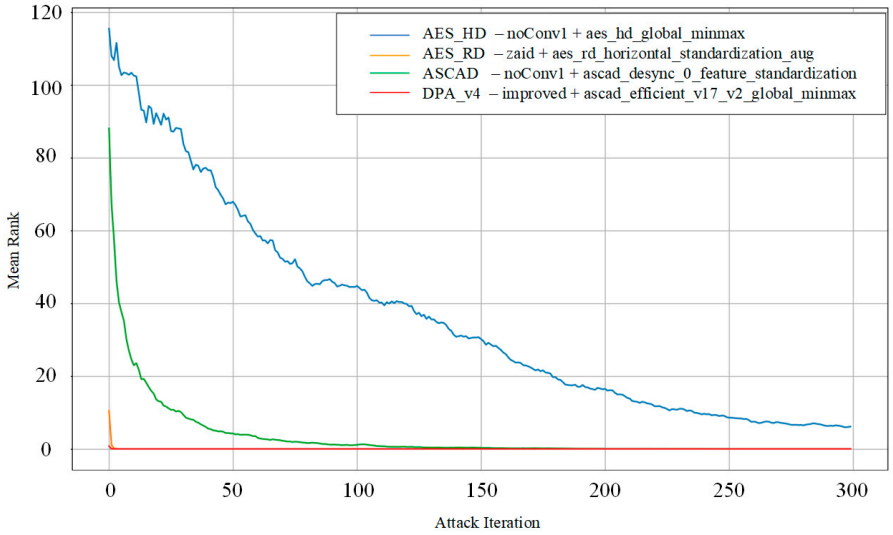


Figure 16. The average ranking of the correct key assumptions of the optimal models of the three methods on the four data task sets varies with the increase in the number of attack traces.

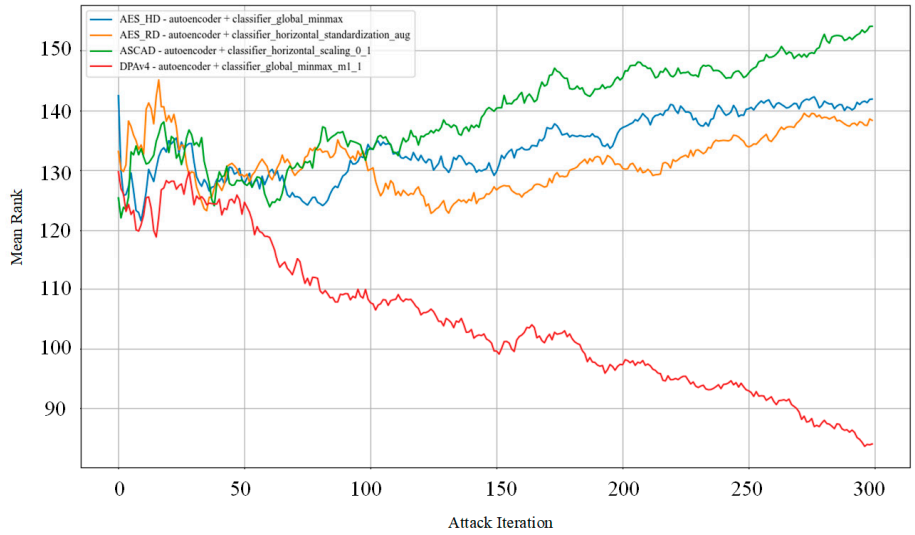


Figure 17. The average ranking of the correct key assumptions of the EFCNN optimal model on the four data task sets varies with the increase in the number of attack traces.

Although the proposed EFCNN structure demonstrates superior performance on target datasets, several potential directions for further experimental validation and improvement are identified:

(1) Cross-Dataset Generalization: Future work will explore the transferability of the proposed model by training on one dataset and evaluating on different datasets without fine tuning to validate its generalization capabilities under different device environments.

(2) Low-Trace Regime Evaluation: Considering realistic side-channel attack scenarios, where only limited traces are available, experiments under extremely reduced training data conditions (e.g., 10%, 20%, 30% of original traces) will be designed to evaluate the robustness and learning efficiency of the model.

(3) Feature Encoder Ablation Studies: Further controlled ablation experiments will be conducted to quantify the individual contribution of the Feature Head Encoder in enhancing classification accuracy and reducing guessing entropy.

(4) Noise Robustness Testing: Robustness to varying levels of artificial noise will be systematically assessed to better simulate real-world environmental disturbances and validate the resilience of the model against noisy measurements.

These directions are expected to form the basis for future extended work, potentially leading to the development of a more unified and generalized framework for side-channel analysis.

5. Conclusions

This paper focuses on side-channel analysis methods based on CNN backbone network variants, systematically reviews the development routes of two typical backbone networks, CNN and transformer, and proposes two improved CNN variant structures—ResNet based on residual structures and the improved feature-coded CNN (EFCNN). Experimental results demonstrate that EFCNN achieves superior performance compared to existing methods on standard datasets, such as DPA_v4 and ASCAD, particularly in clean or moderately noisy environments, while improved ResNet shows better robustness against complex masking countermeasures and high-noise scenarios. The EFCNN enhances the model's ability to understand target leakage features by introducing a feature header encoder inspired by generative network concepts. Experiments show that EFCNN significantly reduces guessing entropy and achieves faster convergence and higher classification accuracy on the selected datasets.

By comparing six preprocessing methods and three data enhancement techniques, this paper finds that feature normalization and global normalization provide the most significant model performance improvement. The preprocessing methods can continue to make part of the trained model mean rank reduced, effectively improving the generalization ability of the model. This paper is the first to achieve a systematic comparison of multiple CNN variants and preprocessing methods on four mainstream datasets at the same time, with more comprehensive coverage of methods, more in-depth model comparisons, and a wider range of datasets, which provides a comprehensive benchmark for model evaluation of side-channel analysis. Experimental results show that the improved ResNet and EFCNN outperform the traditional methods in most scenarios, especially on the DPA_v4 dataset.

Quantifying the impact of different preprocessing and enhancement techniques provides a key reference for data standardization and model optimization for future research. The research results not only validate the effectiveness of deep learning models in side-channel analysis but also promote the development of the field in the direction of greater efficiency and versatility, pointing out possible technical paths for subsequent research.

In order to promote the development of the field, the model code involved in the experiments in this paper has been publicly shared. Future work will focus on dataset full-element analysis and optimization of the transformer structure for side-channel analysis, exploring more efficient analysis methods and designing dedicated or generic models for specific protection measures. These research results will provide continuous support for technological advances in the field of side-channel analysis and drive the field toward smarter and more automated development.

Author Contributions: Conceptualization, W.L. (Weifeng Liu) and W.L. (Wenchang Li); methodology, W.L. (Weifeng Liu) and Y.F.; software, W.L. (Weifeng Liu) and Y.F.; validation, W.L. (Weifeng Liu) and W.L. (Wenchang Li) and Y.F.; writing—original draft preparation, W.L. (Weifeng Liu) and X.L.; writing—review and editing, W.L. (Weifeng Liu), W.L. (Wenchang Li), X.C., J.L., A.C., Y.Z., S.W. and J.Z.; supervision, W.L. (Weifeng Liu) and W.L. (Wenchang Li); funding acquisition, W.L. (Wenchang Li), A.C., Y.Z., S.W. and J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare that this research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Maghrebi, H.; Portigliatti, T.; Prouff, E. Breaking Cryptographic Implementations Using Deep Learning Techniques. In Proceedings of the International Conference on Security, Privacy, and Applied Cryptography Engineering, Hyderabad, India, 14–18 December 2016; pp. 3–26. [CrossRef]
2. Benadjila, R.; Prouff, E.; Strullu, R.; Cagli, E.; Dumas, C. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptogr. Eng.* **2020**, *10*, 163–188. [CrossRef]
3. Martinasek, Z.; Hajny, J.; Malina, L. Optimization of Power Analysis Using Neural Network. In Proceedings of the International Conference on Smart Card Research and Advanced Applications, Paris, France, 5–7 November 2014; pp. 94–107. [CrossRef]
4. Picek, S.; Samiotis, I.P.; Heuser, A.; Kim, J.; Bhasin, S.; Legay, A. On the Performance of Deep Learning for Side-channel Analysis. *IACR Cryptol. ePrint Arch.* **2018**, *2018*, 4.
5. Zeman, V.; Martinasek, Z. Innovative Method of the Power Analysis. *Radioengineering* **2013**, *22*, 586–594.
6. Xiangjun, L.; Chi, Z.; Pei, C.; Dawu, G.; Haining, L. Pay Attention to Raw Traces: A Deep Learning Architecture for End-to-End Profiling Attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2021**, *2021*, 235–274. [CrossRef]
7. Maghrebi, H. Deep Learning based Side Channel Attacks in Practice. *IACR Cryptol. ePrint Arch.* **2019**, *2019*, 578.
8. Cagli, E.; Dumas, C.; Prouff, E. Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures. In Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems, Taipei, Taiwan, 25–28 September 2017; pp. 45–68. [CrossRef]
9. Kim, J.; Picek, S.; Heuser, A.; Bhasin, S.; Hanjalic, A. Make Some Noise. Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2019**, *2019*, 148–179. [CrossRef]
10. Yoo-Seung, W.; Xiaolu, H.; Dirmanto, J.; Jakub, B.; Shivam, B. Back to the Basics: Seamless Integration of Side-Channel Pre-Processing in Deep Neural Networks. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 3215–3227. [CrossRef]
11. Gabriel Zaid, L.B.; Habrard, A.; Venelli, A. Methodology for Efficient CNN Architectures in Profiling Attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2019**, *2020*, 1–36. [CrossRef]
12. Wouters, L.; Arribas, V.; Gierlichs, B.; Preneel, B. Revisiting a Methodology for Efficient CNN Architectures in Profiling Attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2020**, *2020*, 147–168. [CrossRef]
13. Yuanyuan, Z.; François-Xavier, S. Deep learning mitigates but does not annihilate the need of aligned traces and a generalized ResNet model for side-channel attacks. *J. Cryptogr. Eng.* **2020**, *10*, 85–95. [CrossRef]
14. Pei, C.; Chi, Z.; Xiangjun, L.; Dawu, G.; Sen, X. Improving Deep Learning Based Second-Order Side-Channel Analysis with Bilinear CNN. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 3863–3876. [CrossRef]
15. Chen, A.; Li, X.; Yang, N.; Liu, W.; Zhang, Y.; Wang, S.; Zhou, J. Domain-Adaptive Power Profiling Analysis Strategy for the Metaverse. *Int. J. Netw. Manag.* **2024**, *35*. [CrossRef]
16. Paguada, S.; Armendariz, I. The Forgotten Hyperparameter: Introducing Dilated Convolution for Boosting CNN-Based Side-Channel Attacks. In Proceedings of the Applied Cryptography and Network Security Workshops: ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, 19–22 October 2020; pp. 217–236. [CrossRef]
17. Hajra, S.; Saha, S.; Alam, M.; Mukhopadhyay, D. TransNet: Shift Invariant Transformer Network for Side Channel Analysis. In Proceedings of the International Conference on Cryptology in Africa, Fes, Morocco, 18–20 July 2022; pp. 371–396. [CrossRef]
18. Hajra, S.; Chowdhury, S.; Mukhopadhyay, D. EstraNet: An Efficient Shift-Invariant Transformer Network for Side-Channel Analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2023**, *2024*, 336–374. [CrossRef]
19. Fan, J.; Du, Z. Side-Channel Attacks Against the FESH Algorithm. *Front. Comput. Intell. Syst.* **2024**, *7*, 42–48. [CrossRef]
20. Kubota, T.; Yoshida, K.; Shiozaki, M.; Fujino, T. Deep learning side-channel attack against hardware implementations of AES. *Microprocess. Microsyst.* **2021**, *87*, 103383. [CrossRef]
21. DPA Contest v4.1. Available online: https://dpacontest.telecom-paris.fr/v4/rsm_traces.php (accessed on 12 March 2012).
22. Jean-S'ebastien, C.; Ilya, K. AES_RD: Randomdelays-Traces. Available online: https://github.com/ikizhvatov/randomdelays-traces (accessed on 14 April 2021).
23. Shivam Bhasin, D.J.; Picek, S. AES HD Dataset-500 000 Traces. Github Repository. Available online: https://github.com/AISyLab/AES_HD (accessed on 2 December 2020).

24. Shivam Bhasin, D.J.; Picek, S. AES_HD. Github Repository. Available online: https://github.com/AESHD/AES_HD_Dataset (accessed on 13 July 2018).
25. Benadjila, R.; Prouff, E.; Junwei, W. ASCAD (ANSSI SCA Database). Available online: https://github.com/ANSSI-FR/ASCAD (accessed on 9 June 2021).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.