Alkadhim Journal for Computer Science, Vol. 2, No. 1 (2024)

Open Access ISSN: 3007-1437 (Online), ISSN: 3007-1429 (Print)

IRAQI Academic Scientific Journals

Alkadhim Journal for Computer Science (KJCS)

Journal Homepage: https://alkadhum-col.edu.iq/JKCEAS

KJCS ALKADHIM JOURNAL FOR COMPUTER SCIENCE

Adaptive Optimization of Deep Learning Models on AES based Large Side Channel Attack Data

[1]Ammar Abdulhassan Muhammed [*], [2]Israa Akram Alzuabidi,[1]Amjed Abbas Ahmed,[3]Rabiu Aliyu Abdulkadir

[1] Department of Computer Techniques Engineering, Imam Al-Kadhum College (IKC), Baghdad, Iraq

[2]Continuing Education Unit, Collage of Arts, University of Baghdad, Baghdad, Iraq

[3]Department of Electrical Engineering, Aliko Dangote University of Science and Technology, Wudil, Kano, Nigeria

Article information

Article history:
Received: February, 04, 2024
Accepted: March, 05, 2024
Available online: March, 14, 2024

Keywords:
Side-channel attacks,
Deep learning,
Hyperparameters,
Profiling models,
Optimizers.

Corresponding Author:
Ammar Abdulhassan Muhammed
amar.abdalhasan@alkadhum-col.edu.iq

DOI:

Abstract

Deep learning-based side-channel analysis is an efficient and suitable technique for profiling side-channel attacks. In order to obtain the better performance, it is highly necessary to analyze an in-depth training stage in which the optimization of relevant hyperparameters should be a vital process. During the training phase, hyperparameters that are connected to the architecture of the neural network are often selected; however, hyperparameters that impact the training process are to be effectively analyzed. This was represented by an optimized hyperparammeter that consists of considerable impact on attacking behaviour, which is the primary focus of our research. Our research has shown that even while the popular optimizers Adam and RMSprop are capable of delivering satisfactory outcomes, they are also tend to being overfit. Hence, it is necessary to use condensed training periods, simple profiling models, and explicit regularization in order to avoid this problem. On the other hand, the performance of optimizers of the SGD type is only satisfactory when momentum is used which results in slower convergence and less overfit. In conclusion, the research results provide a better use of Adagrad in the cases of longer training datasets or big profiling models.

1. Introduction

As it is possible for attackers to obtain access to physical equipment, the equipment itself must be equipped with cryptographic algorithms [1, 2] in order to provide security not just against physical attacks but also against mathematical cryptanalysis. A security system [3, 4] can be physically compromised by an attack known as a side-channel attack. This kind of attack uses data from cryptographic equipment known as side channels, like time of operation, electromagnetic radiation, temperature, sound, or power use. The use of side-channel attack (SCA) on genuine gadgets, like public transportation and mobile card, has resulted in a greater number of successful hacking attempts [5]. Side-channel attacks [6] can be of profile or non- profiled, depending on the environment in which the attacker is operating. Profiled attack is an example of side- channel attack that is carried out by utilizing a

profiling gadget which is similar to attack targeted gadgets with unchanged concealed keys, just like in Attack of Template and Stochastic Attack [7]. Another instance of side-channel attack is brute-force attack that are carried out using gadgets that is same to attack targeted devices. Attackers use profiling tools to provide a description of the leakage that is occurring on the target device, and then they investigate the device using the information that they have obtained. On the other hand, a non-profiled attack is one of the components that make up a side-channel attack in a scenario where there are no profiling devices present. In order to determine secret keys, attackers just need to collect side-channel data from the device they are targeting and then use statistical techniques to the measurements they have got from the target. These kinds of attacks don't make use of any pre-calculated templates; rather, they rely only on exploiting the side-channel data that is obtained from the devices that are being targeted in order to retrieve the secret key. Examples of attacks that are not profiled are Differential Power Analysis (DPA) [8] and Correlation Power Analysis (CPA) [9, 10].

Recent study has focused on deep learning and side- channel attacks, both of which have shown exceptional performance in a variety of diverse fields [11]. The vast majority of studies have placed an emphasis on scenario profiling for attacks. In a manner similar to that of classic profiling attacks, an attacker trains a network of neural by feeding it data collected through side channels from a profiling gadget. After then, clandestine key is extracted through making use of a neural network that has been trained to classify the side-channel data that has been collected from the target device. According to recent research, side-channel attacks that are powered by deep learning can be utilized to get beyond protections meant for disguising and concealing information [12]. When it comes to profiling deep learning-based side-channel attacks, one of the limitations is that the neural network must be built using training data for which the correct label must be known in order for the attack to be successful. As a direct consequence of this, this tactic has never been used in any research on profiling attacks until both training data and labels have been made accessible.

Timon proposed a differential deep learning analysis, which is side-channel analysis that uses deep learning with context other than profiling, concentrating over our study in profiling atmosphere [13]. The method proposed by Timon makes use of the metrics that are generated by deep learning as a differentiator in place of metrics of convention like correlations amongst side-channel data and inter- mediate variables like projected powers consumption or radiation of electromagnetic nature. As a differentiator, adversaries will leverage deep learning metrics similar to gradiant, accuracy and loss. Strategy makes merits of fact, like measures created containing proper intermediate keys which are simpler in training compared to value produced containing erroneous intermediate key, and that different training threshold are capable in discerning difference between the two values. According to our knowledge, the research conducted by Timon was primarily a person to represent where deep learning could get utilized within non-profile attacks.

Traditional unprofiled side-channel attacks [14], like DPA and CPA such as CPA or DPA, are able to provide fixed results after just one run. Differential deep learning analysis, on the other hand, can produce these findings after only one run. Because the training of deep learning networks is influenced by arbitrary features like weight at initial and hyperparameters, findings of analysis need to be accumulated over course of numerous iterations before they can be studied. As an instance, coefficient of correlation amongst evaluation with artificial intermediate measures stayed same up to point in time when data changed. However, outputs of training differs from behavior to performance despite the fact that training data and their labels are same. This is because random components like hyperparameters along with weighted beginning measures contribute to variation. With non- profiling circumstances, in which an adversary could not acquire access to appropriate labellings, this was further difficult to construct hyperparameters. It is very necessary to reduce the amount of time spent on attacks in order to acquire outcomes for additional hyper-parameters or in reducing probability factor via recurrent attackings. hence, with comparison to more convention attack, ones that are more rapid provide better results.

1.1. Aim

The purpose is to improve the strength of cryptographic systems, and this will be done by making use of the versatility and learning power of deep learning models in order to effectively reduce or eliminate the threats posed by side-channel attacks.

- Designing dynamic optimization methods for deep learning models that enable them to keep pace with evolving attack strategies and security loopholes.

- The integration of adaptive deep learning solutions into cryptographic systems and hardware architectures is to be developed for smooth functioning and compatibility.

1.2. Problem Statement

The deep learning approaches we use today may not be well-suited for the constantly changing domain of side-channel attacks, where attackers always innovate their methods to take advantage of weaknesses. Thus, the research problem is how to design adaptive optimization algorithms for deep learning models that can effectively counteract the adverse effects of side-channel attacks. This involves:

- The design of neural network architectures that adjust their parameters and structure in a responsive fashion to new attack patterns. The generation of algorithms that would work continuously to monitor and analyze the side-channel data to find out the emerging strategies of attacks.

- The work on the optimization algorithms that are integrated to tune model parameters in real time is aimed at suppressing known vulnerabilities. Also, efforts have been made in research to improve deep learning model generalization capabilities against different types of side-channel attacks on various hardware platforms.

1.3. Outline

In this article, the research work is organized as follows,

Section 1 explains the introduction, aim and problem statement in brief. Section 2 details the literature survey related to our research work. Section 3 provides design of our proposed method of CNN architecture with adaptive optimization model. Section 4 presents the results of our research work. Finally, Section 5 summarizes the research work with a detailed conclusion.

2. LITERATURE REVIEW

We can categorize corpus of works used to describe machine learning methods according to degree of complication of applied methods and treatment of hyper- parameter phase. In fact, primary studies examined more basic machine learning methods as Multilayer Perceptron, Naive Bayes, Support Vector Machines and Naive Forest [14]. The effectiveness of attacks and how those tactics stack up against the template attack and its variations were the key points of attention. The SCA community began to focus heavily on deep learning methods starting in 2016 [15]. Convolutional neural networks and multilayer perceptron 2 are the two most investigated methods. Both of those methods attained maximum performance, making it even conceivable to defeat implementations that are fortified with defenses [16]. In recent times, fraternity of research has begun in broadening deep learning approach to profile SCA. For example, have a look at the autoencoders used in [17] to pre-process the trace data.

From the standpoint of the hyperparameter tuning phase, the machine learning proficiency of the SCA community can also be evaluated. In fact, early studies [18] does not specify if they perform tuning of hyper-parameter or what finalized hyperparameters are selected. Then, several works take into account different machine learning approaches to carry out tuning of hyperparameter by randomized or grid searching. The authors of [19] undertake an experimental assessment of various hyper- parameters concerning CNNs using ASCAD repository. To create deep learning model ensembles, Perin et al. used a random search inside pre-specified ranges [20]. A number of studies seek to systematically assess the impact of different hyperparameters. For the number of layers, neurons, and activation functions, L. Weissbart [21] took into consideration multilayer perceptrons and hyperparameter tuning. For Multilayer Perceptron (MLP) and CNN architectures, Li et al. looked at the weight initialization role [22]. Picek and Perin [23] investigated impact of optimizers selection pertaining to side-channel analysis dependent on deep learning. Such papers explain effects of certain hyperparameters in particular contexts, but they do not include instructions on how to construct whole neural network structures. These publications also address the relative importance of various hyperparameters, which can aid in future hyperparameter tuning improvements.

Finally, a number of studies attempt to provide a mechanism for creating neural networks. A strategy for choosing hyperparameters linked to sizes (number of learnable features, such as biases and weights) of CNNs layers was put out by the authors in [24]. The quantity of filters, kernel sizes, strides, and neurons in fully linked layers are all included in this. To the best of our knowledge, this is the first study to attempt a systematic response to the topic of how to create neural network topologies that function well. Unfortunately, a true technique is still a long way off. Although writers provide CNN designs achieving highest behavior, they presume extensive dataset knowledge too and solely take into account CNNs. Wouters et al. [25] enhanced work out of Zaid et al. [26], in which they explain many misunderstanding in primary tasks, and representing the way in reaching identical attack behavior having important tiny architecture of neural network. Furthermore, it is a bit complicated in extending these techniques over to newest repository.

With the improvements in hardware and learning methods, deep learning has gained traction in a number of applications outside computer vision and natural language processing and voice recognition. Using deep learning, researchers have been able to classify side-channel data [27, 28]. There has been much study into the possibility of using deep learning to categorize measurements in the context of profiling, when an attacker uses a profiling equipment. While the attacker explains the profiling device leaking, the target device's measurements are analyzed. Maghrebi found that profiling attacks based on deep learning could be assessed apart from the masking reaction [18]. Without the need to preprocess the input, Cagli et al. [29] found that an attacker utilizing convolutional neural networks could potentially recover the secret key using side-channel attacks based on deep learning key. The results of deep learning investigations can be compared with one another with the use of open datasets [30]. In order to break public-key cryptosystems [31], researchers have employed deep learning in tandem with block ciphers.

The first deep learning-based side-channel analysis [32, 33] is differential deep learning analysis (DDLA). When an attacker doesn't have access to a profiled device, this method can be utilized instead. Deep learning attacks-based profiling of side-channel attacks is challenging to accomplish because side-channel readings cannot be classified without a template device. These issues are solved by DDLA's label guessing. One common kind of covert study, known as a correlation power analysis, demonstrates that intermediate values generated with the right key are strongly correlated with observations. An accurately labeled neural network will outperform an inaccurately labeled one in every deep learning investigation. DDLA classifies various deep learning metrics differently. The validity of the key can be ascertained by the attacker with its unique identifier.

3. CNN Architecture with Adaptive Optimization Model

For the purpose of the side-channel profiling study, we take into consideration a typical scenario. An intelligent attacker has access to a device (clone device) that has the information necessary to implement the secret key. An assortment of N profiling traces $\boldsymbol{X_1}, \ldots, \boldsymbol{X_N}$, each corresponding to the processing of plaintext or ciphertext $\boldsymbol{T_p}$, can be retrieved by the attacker. These traces can be used to profile the processing of the plaintext or ciphertext. Using this data, develops a leaky model that calls $\boldsymbol{Y}(\boldsymbol{T_p}, \boldsymbol{k}^*)$. The data are subsequently used by the attacker in the construction of a profiling model f, thus the term "profiling phase" for this step of the process. It is possible to initiate the attack on a separate device by using the mapping function. During the attack phase, sometimes known simply as the attack phase, the attacker will gather further Q traces $\boldsymbol{X_1}, \ldots, \boldsymbol{X_Q}$ from the target device in order to discover the unknowable secret key $\boldsymbol{k_a^*}$.

In order to calculate effectiveness of attacks, we require computing measures. Success rate (SR) and guessing entropy (GE) are the two-evaluation metrics that are used in SCA the majority of the time. Following the completion of a side-channel analysis, GE reveals the usual number of key candidates that an adversary is need to examine in order to deduce the location of the secret key. Let Q stand for the total number of measurements that were made during the attack phase, and let $\boldsymbol{g} = [\boldsymbol{g_1}, \boldsymbol{g_2}, \ldots, \boldsymbol{g}|\boldsymbol{K}|]$ stand for the key guessing vector that the attack created. $|\boldsymbol{K}|$ indicates key-space size. Entropy of guessing was calculated by taking average value of where $\boldsymbol{k_a^*}$ and g are located on several trials. Estimation was done by us for entropy based on the average of 100 separate experiments. Since we only evaluate entropy of guessing for a one key byte, term that more accurately describes what we do is called partial guessing entropy. Despite this, we talk about each of these ideas interchangeably.

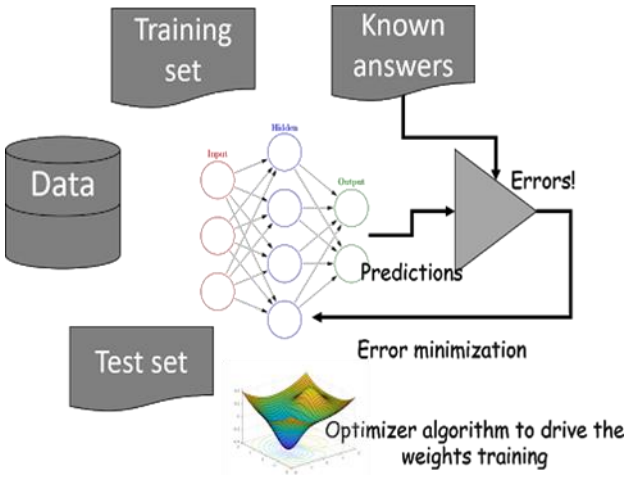


Figure (1): Optimizer Based CNN Model.

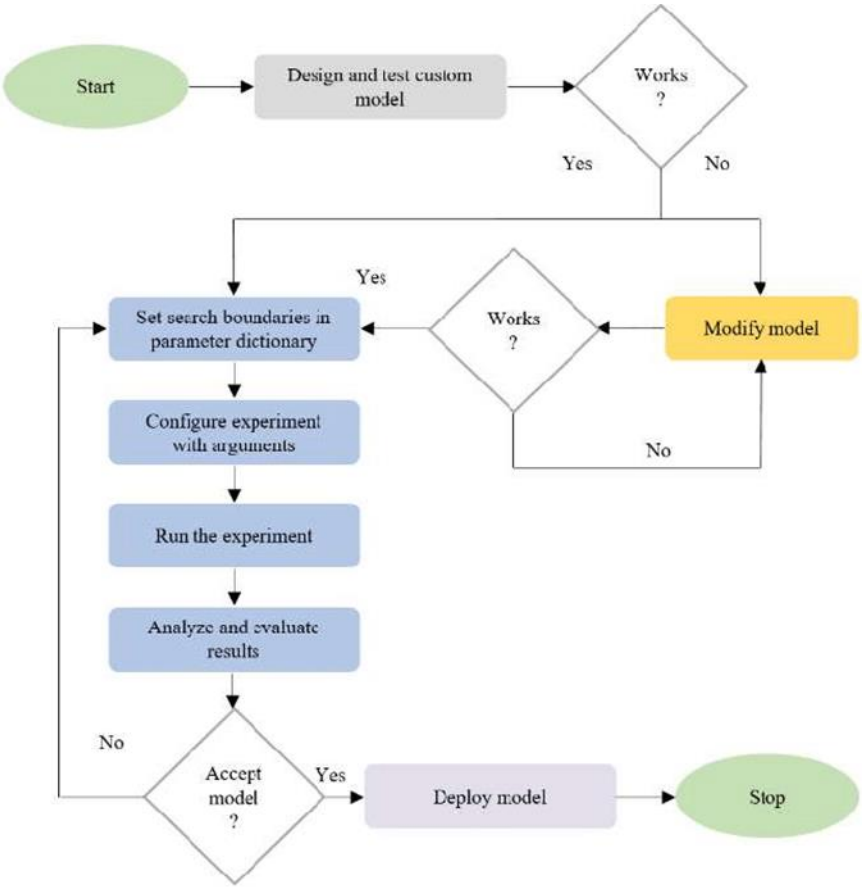


Figure (2): Flowchart for adaptively optimizing the CNN model's hyperparameters

Since supervised machine learning follows the same technique as the supervised learning paradigm, which can be illustrated, it is possible for us in using regulated machine learning concerning SCA. To be additional exact, phase corresponding to testing is called the attack phase, whereas the one that corresponds to training is called the profiling phase. The following three components are crucial to the algorithms used in machine learning: A machine learning method is comprised of three distinct parts: a model, a loss function (the majority of the objective functions used in machine learning are, as a matter of convention, expected to be lowered), and a loss function optimization procedure. In order to develop a successful profiling model, that is to say one that generalizes well to data that has not been seen before, we provided training to parameters θ set a way that losses are minimized. In

the last step, a variety of hyperparameters λ are included into the development of the learning model. During our examinations, we take into consideration both convolutional neural networks and multiple layer perceptron.

Before optimization can begin with the intention of lowering a loss function or cost function, that function must first be specified. This function is also referred to as an objective function. It can rely on the optimization approach that we choose as to whether we achieve high accuracy in a matter of hours or days. Figure 1 shows the optimizer-based CNN model. After making some adjustments, the modified CNN is fine-tuned using the adaptive optimization for its parameters and hyperparameters, such as (i) hidden neuron in dense layers, (ii) separable-convolutional dropouts, (iii) dense layer dropouts, (iv) optimizer functions (v) non-linear activations [33]. Figure 2 shows a flowchart for adaptively optimizing the CNN model's hyperparameters. Models that have been pretrained are instantiated using the weights that were extracted from ImageNet.

3.1. Machine learning classifiers

We take into account the multilayer perceptron as well as the convolutional neural networks, which are both varieties of neural networks that are typical procedures in the process of profiling SCA.

3.2. Multilayer perceptron

A multiple layer perceptron, often called MLP, is a kind of forward-feed neural network that is responsible for translating different sets of inputs into other sets of usable outputs. The fact that each layer in the directed graph of an MLP is fully connected to the one behind it is what gives the layers their names-fully-connected layers and dense layers, respectively. Because input and output are considered to be two separate levels, an MLP consists of at least three and no more than five substrates of non-linearly nodes of activating.

3.2.1. Convolutional neural networks

Feed-forward network of neural, also known as convolutional neural networks (CNNs), generally containing 3 distinct kinds of substrate. These layers are convolution, assembling, and fully concatenated, respectively. Substrate of convolution calculates neurons output of neurons tied to certain areas over input by performing a calculation known as dot product amongst weight of each neuron and very tiny area that is directly related to the volume of the input. Pooling brings about a reduction in the number of recovered features as a result of downsampling along the spatial dimensions. Either the class scores or the hidden activations can be computed by the entirely-joined substrate (much as in a multiple layer perceptron).

3.3. Neural networks hyperparameter ranges

Hyper-parameter variation that was studied by us, concerning architecture of MLP are shown in Table I, and range that we investigate for CNN architectures are presented in Table II.

Table (1): The Search Space for The Hyperparameters of The Multilayer Perceptron.

Hyperparameter	Min		Max	Step
Learning rate	0.0001		0.01	0.0001
Mini-batch	400		1000	100
Dense (fully-connected) layers	1		10	1
Neurons (for dense layers)	100		1000	10
Activation function (all layers)	ReLU, Tanh, ELU, or SELU			

Table (2): The search space for the hyperparameters of the convolutional neural network (l stands for the convolution index of substrate).

Hyperparameter	Min	Max	Step
Learning rate	0.0001	0.01	0.0001
Mini-batch	400	1000	100

Hyperparameter	Min	Max	Step
Learning rate	0.0001	0.01	0.0001
Convolution layers	1	4	1
Convolution filters	$4 * l$	$8 * l$	1
Convolution kernel size	1	40	1
Convolution stride	1	4	1
Dense (fully-connected) layers	1	10	1
Neurons (for dense layers)	100	1000	10
Activation function (all layers)	ReLU, Tanh, ELU, or SELU		

z-score normalization has been used in our research for the purposes of standardizing test side channel traces, training and validations:

$$\boldsymbol{t_i = \frac{t_i - \mu_{T_{train}}}{\sigma_{T_{train}}}} \quad \textbf{(1)}$$

In which T_{train} is a collection of training trace with t_i are lone traces taken out of validation, training or testing set. Bear in mind that every trace has been normalized by using the data from the training set. In none of the possible configurations of neural networks is there a batch normalization layer to be found.

3.4. Adaptive Optimizers

3.4.1. RMSprop

The adaptive learning rate approach RMSprop is going to be used in an effort to rectify the situation with Adagrad's rapidly declining learning rates, which are similar to Adadelta's. RMSprop takes the learning rate and divides it by the average squared gradient, which then decreases at an exponential rate.

3.4.2. Adam

The Adam (Adaptive Moment Estimation) method is still another approach that is used to ascertain the adaptive learning frequency pertaining to every parameter. Such methodology mixes merits that are offered by the Adagrad and RMSprop methodologies into a single solution. It does this in a manner that is analogous to the momentum technique by storing an exponentially decaying average of past squared gradients in addition to an earlier exponentially decaying average of gradients.

4. RESULTS

We need to be certain that a particular neural network architecture can behave differently for a number of optimizers, or that selecting one optimizer can influence the value of another hyperparameter. For this purpose, we define very big MLPs and very huge CNNs. Entire models of neural network, whether CNN or MLP, were trained using a batches size around 400, a learning frequency of 0.001, the starting biases and weight that are provided by default by the Keras library. The definitions that follow are applicable to the large-scale MLP and CNN models:

A huge MLP with ten concealed layers, every containing one thousand neurons. A substantial CNN consisting of 4 substrate of convolution and four entirely joined layers, each having a total of 1000 neurons (10, 20, 40, and 80 filters, stride 2 and kernel size 4 in all four layers of convolution).

Layers at output concerning CNN and MLP each consist of 9 neurons, which are modeled after the Hamming Weight leakage algorithm and activated using the Softmax function. Unless it is specifically specified differently, the ReLU activation function is present in each layer of all neural networks. In addition, conclusions are presented for the ELU activation function applicable to profiling and adaptive optimizers models. Such network of neural are trained together of ASCAD (ANSSI SCA Database) repository, having both random and fixed keys included in data of training. Following the execution of study ten turns, having identical framework each time, for each data set and neural network, we then take the average of generalization outcomes out of such ten runs.

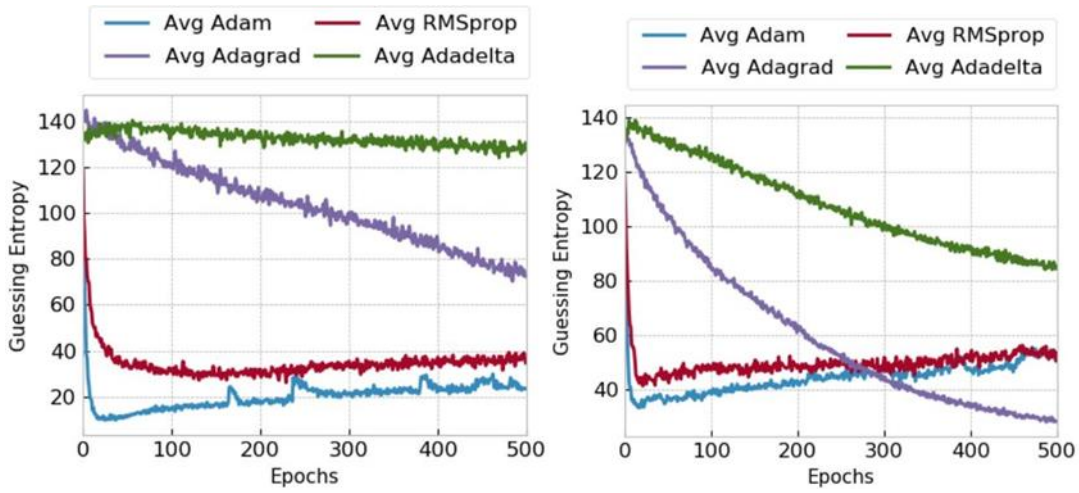
4.1. Adaptive Gradient Descent Methods

We examined the empirical generalizability of adaptive methods. Their research suggests that adaptive optimizers could cause overfitting in models that have too many parameters. Adaptive optimizers like Adam and RMSprop make more progress at the outset of training but often lose effectiveness as they amass a larger number of training epochs. The precision of the data exceeds that of the model. Regularizing overfitting requires precision.

We look at Adam and RMSprop, two adaptive optimizers for profiled SCA, and demonstrate that they overfit, whereas Adagrad and Adadelta do not. For more complex models and longer training times, Adagrad and Adadelta function better. ASCAD receives results for both small and big models in terms of fixed key averaged guessing entropy, as shown in Figs. 3a and 3b. Figures 4a and 4b depict the development of GE for small and large models during ASCAD random key training, respectively.

Both datasets show a broad pattern of guessing entropy rise during adaptive optimizer training that is comparable to one another, showing that this is a natural optimizer behavior that is independent of the dataset under attack. Adam and RMSprop are rapidly becoming more similar to one another. Once the model can generalize, the entropy of correctly guessing the most promising choices drops below 20.

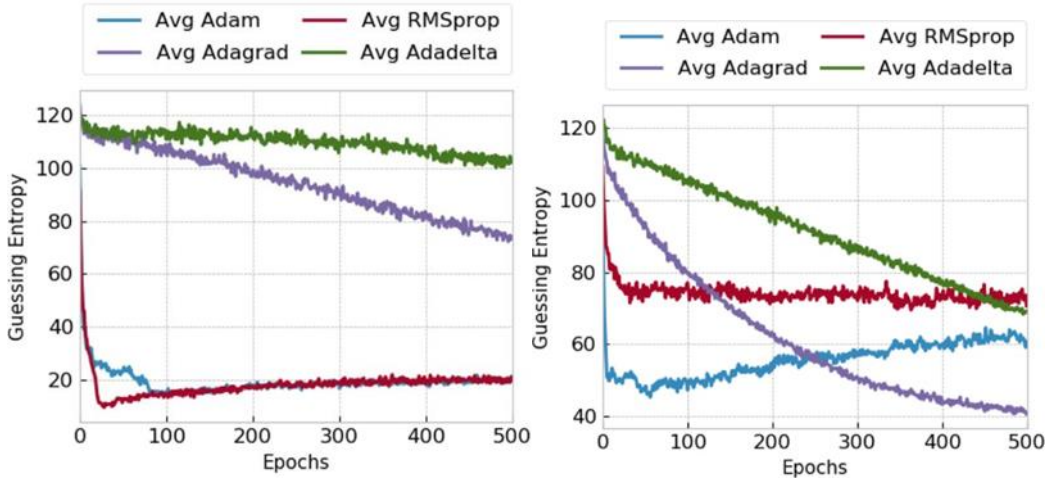
By continuing to process epochs after the guessing entropy has reached its lowest (when it has achieved its maximum generalization), we can reduce the model's generalization. Since the Adam optimizer quickly overfits, interrupting it before it completes its task could be useful. Adam's optimizer can profile SCA without much training. For more complex models and data sets, Adam is superior than RMSprop.



(a) Small Models (b) Large Models

Figure (3):Adaptive optimizer and models' size for the ASCAD data set with fixed keys.

Adagrad can reduce the guessing entropy if the model can be extended. Adagrad, unlike Adam and RMSprop, benefits from a longer training period when used for optimization since the guessing entropy does not decrease with more epochs. Adagrad can need more training epochs than usual since generalization cannot occur until much later in the process. In deep neural network architectures, Adadelta generalizes less than other methods. Just with Adagrad, generalization retains its accuracy throughout the course of several epochs. Generalization can occur even if training begins late. When dealing with bigger models (which suggests that the dataset is challenging and asks for a considerable amount of profiled model capacity) and when it is unclear how to tune other hyperparameters, our findings demonstrate that Adagrad is the most efficient optimizer to utilize. We found that for Adam and RMSprop on the ASCAD random keys dataset, MLPs converge more quickly (with lower guessing entropy) than CNNs. This was yet another discovery of ours. Results for Adagrad and Adadelta using MLPs and CNNs were quite similar.

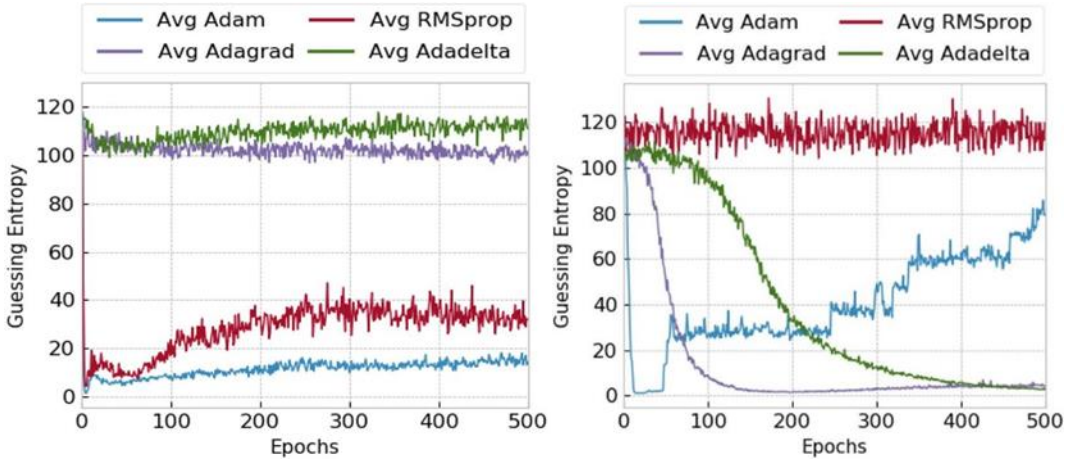


(a) Small Models (b) Large Models

Figure (4):Adaptive optimizer and models' size for the ASCAD data set with random keys.

4.2. Adaptive Optimizers on Large MLP and CNN

In this example, we show that different kinds of activation functions and adaptive optimizers react in a variety of different ways. Adam and RMSprop often have worse performance when compared to tiny models of MLP, and this is particularly true if ELU is employed as function f activation, as shown in Figure 5b. Even though these two optimizers are especially prone to overfitting due to the fact that GE grows if the quantity of training epochs is too big, they tend to perform relatively better for activation function of ReLU is shown in Figure 5a, that are often utilized with state-of-the-art neural design of network. This is despite the fact that Figure 5a shows that this activation function is frequently employed. We observed that for Adedelta and Adagrad, a big MLP works pretty good when activation function of ELU was utilized for concealed substrate. This is depicted in Figure 5b, with such particular scenario, network needs additional train epochs (about 400 in as shown in Figure 5b) in order to congregate to an entropy of guessing that is low. Even for very high numbers of epochs (such as 500 epochs), large MLP models, and ELU, Adagrad and Adadelta have advantage of low guessing entropy. This is true even for ELU. The results of the investigations into 4 different adaptive optimizers concerning huge MLP are shown, respectively, in Figures 5a and 5b.

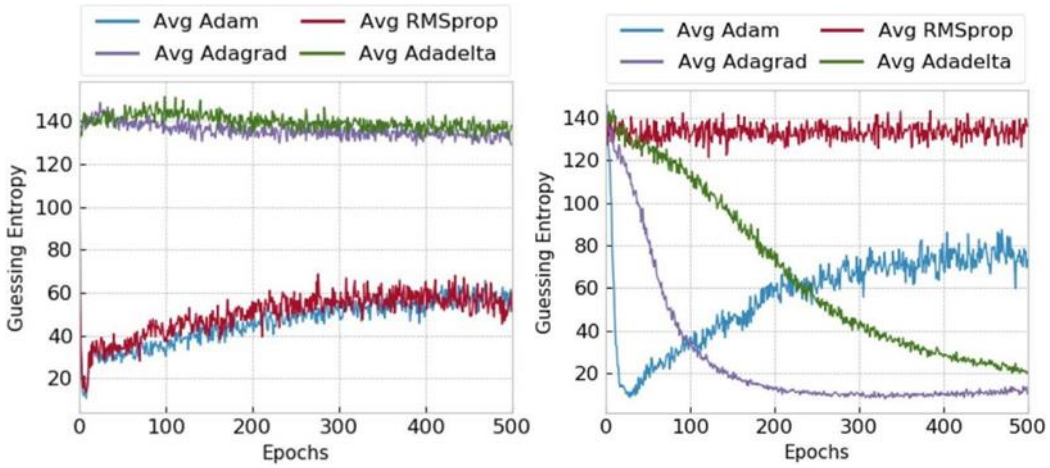


a) ReLU (b) ELU

Figure (5): Adaptive optimizers applied to huge MLP prototypes for use in ASCAD with random keys.

The similar pattern of performance are seen for ASCAD unchanged keys information set as well. Figures 5a and 5b, respectively, demonstrate outcomes that were obtained using a large models of MLP that used ELU and ReLU function of activation. According to findings of research presented in this part, the performance of large MLP

models is improved by using Adagrad and Adadelta as optimizers in conjunction with an ELU activation function. Even after giving great consideration to the activation function to use, it was not possible to guarantee a consistent convergence of guessing entropy throughout the training process for the Adam and RMSprop cases. Once again, we show that Adam and RMSprop have the potential to give higher performance by including additional artifacts of regularization, like early pausing. If set of training contains few evaluations, as is case with ASCAD unchanged data set of key, RMSprop and Adam often produce a small interval of generalization. This is also situation with ASCAD unchaged key information set. As evident through Figure 6a, a lower guess entropy pertaining to RMSprop and Adam only lasts for a short period of time- less than 10 epochs-and then it begins to steadily increase. As it can be observed in the case of Figure 6b, interval where entropy of guessing is lower covers a greater range when larger training sets are used. This dataset was contributed by ASCAD random data sets keys. With this particular scenario, the guessing entropy is low, and it maintains that state for a minimum of fifty processing epochs.



(a) ReLU (b) ELU

Figure (6): Adaptive optimizers applied to huge MLP prototypes for use in ASCAD with fixed keys.

5. Conclusions

The optimizer method that was used during the training phase of SCA profiling has a considerable impact over outcomes that were obtained from attacks. With such investigation, findings from eight various optimizers was provided by us, which we have categorized into groups for SGD and adaptive optimization respectively. It was observed by us, that RMSprop and Adam optimizers perform good if network of neural systems to be optimized is on smaller side and the training time is kept to a minimum. Both the Adagrad and the Adadelta adaptive algorithms perform quite well when large models are taken into consideration. In addition to this, we confirmed that the activation function for hidden layers is the sole factor that has a role in the selection of the adaptive optimizer. In our further study, we want to investigate the performance of a number of optimizers, as well as the identity value leaking model. In addition, we limit our attention to only two datasets for the duration of this research. In order to ensure the accuracy of our findings, we plan on extending the scope of the research to include other datasets that are available to the general public. Then, we are going to look at twice descent SCA's performance. Even though it was discovered by us, that elongated training do not always result in greater behavior, we were still attracted in discovering that greater models do not always perform in a good way and longer phases of profiling contribute to better performance. Both of these hypotheses are now under investigation by our team.

Conflict of Interest: The authors declare no conflicts of interest.

References

[1] H. Wang and E. Dubrova, "Tandem deep learning side-channel attack on FPGA implementation of AES," SN Computer Science, vol. 2, pp. 1-12, 2021.

[2] K. Nomikos, A. Papadimitriou, M. Psarakis, A. Pikrakis, and V. Beroulle, "Evaluation of Hiding-based Countermeasures against Deep Learning Side Channel Attacks with Pre-trained Networks," in IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2022, pp. 1-6: IEEE.

[3] M. K. Hasan et al., "Lightweight cryptographic algorithms for guessing attack protection in complex internet of things applications," Complexity, vol. 2021, pp. 1-13, 2021.

[4] A. A. Ahmed, M. K. Hasan, S. Islam, A. H. M. Aman, and N. Safie, "Design of Convolutional Neural Networks Architecture for Non-Profiled Side-Channel Attack Detection," Elektronika Ir Elektrotechnika, vol. 29, no. 4, pp. 76-81, 2023.

[5] C. Dhasarathan et al., "COVID-19 health data analysis and personal data preserving: A homomorphic privacy enforcement approach," Computer communications, vol. 199, pp. 87-97, 2023.

[6] C. Wang et al., "Portability of Deep-Learning Side-Channel Attacks against Software Discrepancies," in Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2023, pp. 227-238.

[7] A. A. Ahmed, R. A. Salim, and M. K. Hasan, "Deep Learning Method for Power Side-Channel Analysis on Chip Leakages," Elektronika ir Elektrotechnika, vol. 29, no. 6, pp. 50-57, 2023.

[8] V.-P. Hoang, N.-T. Do, and V. S. Doan, "Performance Analysis of Deep Learning Based Non-profiled Side Channel Attacks Using Significant Hamming Weight Labeling," Mobile Networks Applications, pp. 1-10, 2023.

[9] A. A. Ahmed and M. K. Hasan, "Design and Implementation of Side Channel Attack Based on Deep Learning LSTM," in IEEE Region 10 Symposium (TENSYMP), 2023, pp. 1-6: IEEE.

[10] A. Ito, K. Saito, R. Ueno, and N. Homma, "Imbalanced data problems in deep learning-based side-channel attacks: Analysis and solution," IEEE Transactions on Information Forensics Security, vol. 16, pp. 3790-3802, 2021.

[11] A. A. Ahmed et al., "Detection of Crucial Power Side Channel Data Leakage in Neural Networks," in 33rd International Telecommunication Networks and Applications Conference, 2023, pp. 57-62: IEEE.

[12] S. Bhasin, A. Mendelson, and M. Nandi, "Security, Privacy, and Applied Cryptography Engineering: ," in 9th International Conference, SPACE 2019,, Gandhinagar, India, 2019, vol. 11947: Springer Nature.

[13] A. A. Ahmed, M. K. Hasan, N. S. Nafi, A. H. Aman, S. Islam, and S. A. Fadhil, "Design of Lightweight Cryptography based Deep Learning Model for Side Channel Attacks," in 2023 33rd International Telecommunication Networks and Applications Conference, 2023, pp. 325-328: IEEE.

[14] Y. Ou and L. Li, "Side-channel analysis attacks based on deep learning network," Frontiers of Computer Science, vol. 16, pp. 1-11, 2022.

[15] D. Kwon, S. Hong, and H. Kim, "Optimizing implementations of non-profiled deep learning-based side-channel attacks," IEEE Access, vol. 10, pp. 5957-5967, 2022.

[16] A. A. Ahmed, M. K. Hasan, N. S. Nafi, A. H. Aman, S. Islam, and M. S. Nahi, "Optimization Technique for Deep Learning Methodology on Power Side Channel Attacks," in 2023 33rd International Telecommunication Networks and Applications Conference, 2023, pp. 80-83: IEEE.

[17] L. Zhang, X. Xing, J. Fan, Z. Wang, and S. Wang, "Multilabel deep learning-based side-channel attack," IEEE Transactions on Computer-Aided Design of Integrated Circuits Systems, vol. 40, no. 6, pp. 1207-1216, 2020.

[18] H. Maghrebi, "Deep learning based side channel attacks in practice," Cryptology ePrint Archive, 2019.

[19] N. T. Do, V. P. Hoang, V. S. Doan, and C. K. Pham, "On the performance of non-profiled side channel attacks based on deep learning techniques," IET Information Security, vol. 17, no. 3, pp. 377-393, 2023.

[20] D. Thapar, M. Alam, and D. Mukhopadhyay, "Transca: Cross-family profiled side-channel attacks using transfer learning on deep neural networks," Cryptology ePrint Archive, 2020.

[21] L. Weissbart, "Performance analysis of multilayer perceptron in profiling side-channel analysis," in Applied Cryptography and Network Security Workshops: ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, 2020, pp. 198-216: Springer.

[22] Li, Huimin, Marina Krček, and Guilherme Perin. "A comparison of weight initializers in deep learning-based side-channel analysis." Applied Cryptography and Network Security Workshops: ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, October 19–22, 2020, Proceedings 18. Springer International Publishing, 2020.

[23] G. Perin and S. Picek, "On the Influence of Optimizers in Deep Learning-Based Side-Channel Analysis," in Selected Areas in Cryptography, Cham, 2021, pp. 615-636: Springer International Publishing.

[24] H. Wang, M. Brisfors, S. Forsmark, and E. Dubrova, "How diversity affects deep-learning side-channel attacks," in IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC), Helsinki, Finland, 2019, pp. 1-7: IEEE.

[25] Wouters, Lennert, et al. "Revisiting a methodology for efficient CNN architectures in profiling attacks." IACR Transactions on Cryptographic Hardware and Embedded Systems (2020): 147-168.

[26] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Efficiency through diversity in ensemble models applied to side-channel attacks:–a case study on public-key algorithms–," IACR Transactions on Cryptographic Hardware Embedded Systems, vol. 2021, no. 3, pp. 60-96, 2021.

[27] T. Kubota, K. Yoshida, M. Shiozaki, and T. Fujino, "Deep learning side-channel attack against hardware implementations of AES," Microprocess. Microsyst., vol. 87, p. 103383, 2021.

[28] S. Gupta et al., "A Novel Approach Toward the Prevention of the Side Channel Attacks for Enhancing the Network Security," no. Pre-print, 2022.

[29] E. Cagli, C. Dumas, and E. Prouff, "Enhancing dimensionality reduction methods for side-channel attacks," in Smart Card Research and Advanced Applications: 14th International Conference, CARDIS 2015, Bochum, Germany, 2016, pp. 15-33: Springer.

[30] M. Panoff, H. Yu, H. Shan, and Y. Jin, "A Review and Comparison of AI-enhanced Side Channel Analysis," ACM Journal on Emerging Technologies in Computing Systems, vol. 18, no. 3, pp. 1-20, 2022.

[31] Y. Zhou and F.-X. Standaert, "S-box pooling: towards more efficient side-channel security evaluations," in International Conference on Applied Cryptography and Network Security, Rome, Italy, 2022, pp. 146-164: Springer.

[32] S. Swaminathan, Ł. Chmielewski, G. Perin, and S. Picek, "Deep learning-based side-channel analysis against aes inner rounds," in International Conference on Applied Cryptography and Network Security, Rome, Italy, 2022, pp. 165-182: Springer.

[33] L. Chang, Y. Wei, S. He, and X. Pan, "Research on Side-Channel Analysis Based on Deep Learning with Different Sample Data," Applied Sciences, vol. 12, no. 16, p. 8246, 2022.