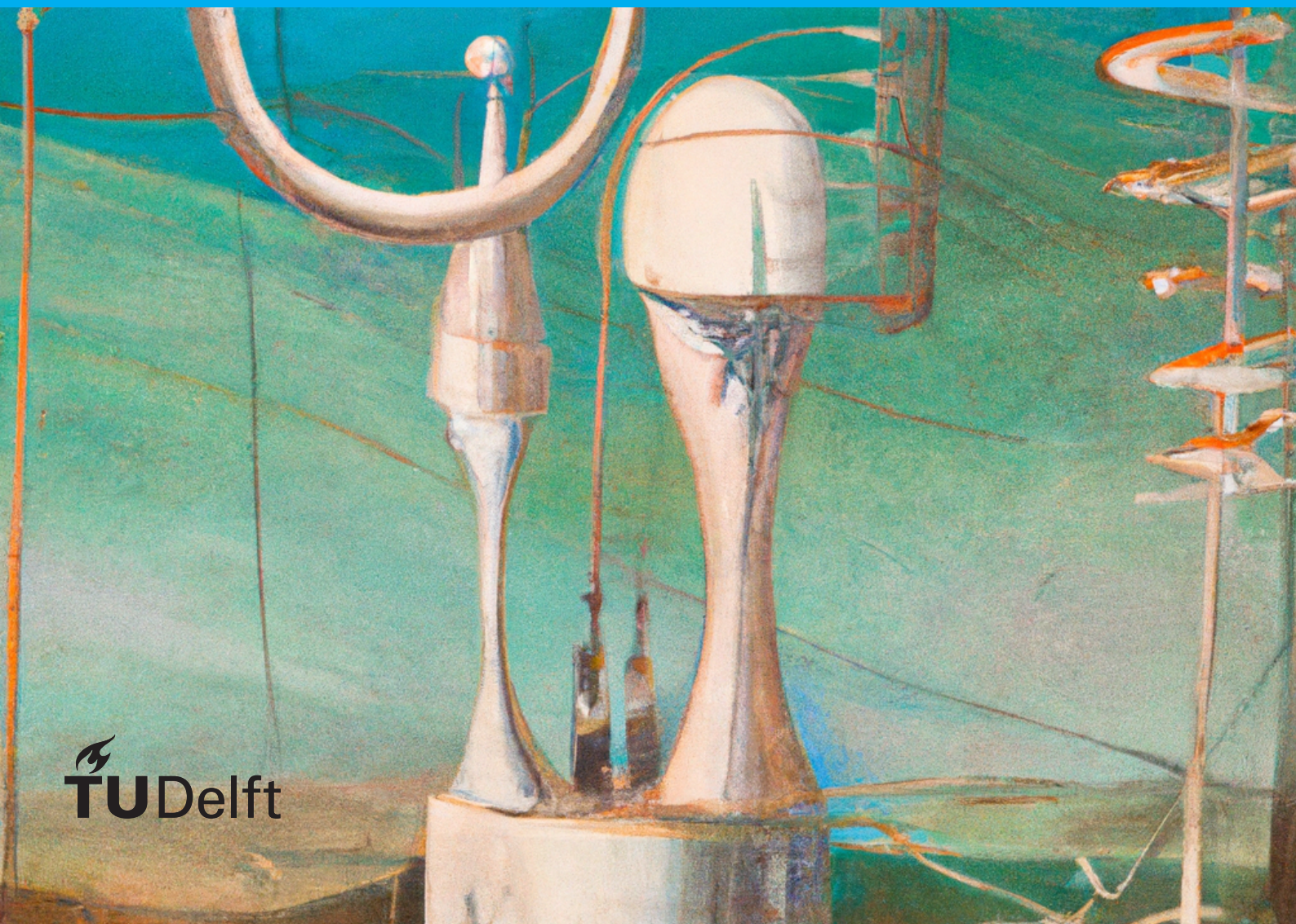
Unsupervised Deep Learning-based SCA

Non-profiled Multi-output Regression in Side-Channel Analysis

Ioana Savu



TU Delft

Unsupervised Deep Learning-based SCA

Non-profiled Multi-output Regression in Side-Channel Analysis

by

Ioana Savu

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday, July 11, 2023, at 15:30 PM.

Student number:	4842375	
Project duration:	November 14, 2022 – July 11, 2023	
Thesis committee:	Dr. S. Picek,	TU Delft, supervisor
	Prof. dr. ir. Inald Lagendijk,	TU Delft
	Prof. dr. Leila Batina,	Radboud University

This thesis is confidential and cannot be made public until July 11, 2023.

TU Delft

Abstract

Side-channel attacks (SCA) play a crucial role in assessing the security of the implementation of cryptographic algorithms. Still, traditional profiled attacks require a nearly identical reference device to the target, limiting their practicality. This thesis focuses on non-profiled SCA, which provides a realistic alternative when the attacker lacks access to a profiling device. Specifically, we investigate non-profiled deep learning-based SCA techniques. Our evaluation first explores existing unsupervised deep learning-based side-channel analysis approaches: differential deep learning analysis (DDLA) and multi-output regression (MOR). We show that using a validation set for key distinguishing, rather than the training set, improves the overall success rate across various datasets.

In the context of multi-output regression SCA, we comprehensively evaluate different loss functions. The thesis proposes a novel approach that outperforms existing methods by employing the normalized Z-score MSE (Z-MSE) loss function. Additionally, we introduce two key distinguishing methods, one based on the smallest Z-MSE loss and the other on the highest Pearson correlation between actual and predicted labels during validation. The experimental results show the efficacy of the novel approach in breaking traces protected by countermeasures. Notably, even with high levels of desynchronization (250) for ASCADf traces, the attack succeeds within a limited number of epochs (15).

Moreover, we demonstrate that the performance of the novel approach can be further enhanced through ensembles. This leads to a reduction in the number of traces required to break the key for most datasets and a decrease in the average guessing entropy. Data augmentation also proves beneficial in some instances, resulting in improved success rates.

Preface

This thesis concludes my five years of studying at TU Delft, where I have been fortunate to work with many inspiring and supportive people. I am grateful for their guidance, encouragement, and friendship.

First and foremost, I would like to thank my supervisor, Stjepan Picek, for his advice, feedback, and mentorship throughout this project. He has been a constant source of motivation, and I have learned a lot from his expertise and experience. I am genuinely grateful for his trust and support.

I would like to thank Marina Krcek and Guilherme Perin for their helpful suggestions, constructive criticism, and guidance. I also want to express my gratitude to Lichao Wu, even though we never met in person.

Thanks to all my Cybersecurity colleagues and friends for making my life enjoyable and memorable during my studies at TU Delft. Special thanks to Dan Andreescu, Ion Babalau, Bob Brockbernd, Marin Duroyon, Andrei Geadau, Dan Plamadeala, Konrad Ponichtera, Mariana Samardzic, Natalia Struharova, Wessel Thomas, and Bram Verboom for the coffee breaks, brainstorming ideas, sharing frustrations, or celebrating achievements.

Last but not least, I want to thank my family for their unconditional love, support, and encouragement throughout my studies. They have always believed in me and supported me in following my dreams.

Ioana Savu
Delft, July 2023

Contents

Chapter 1

Introduction

Embedded devices, also known as Internet of Things (IoT) devices, play an important role in various domains, such as smart homes, smart cities, healthcare, transportation, and industry. However, designers and developers may face security challenges due to the compact size and limited computational power of these systems. IoT connects multiple devices with different operating systems, processing power, and functionalities. The diversity of the network and the widespread use of IoT devices need enhanced security and privacy protection. Therefore, designers and developers require robust cryptographic techniques that meet higher standards while remaining practical for implementation on limited devices [43].

The Advanced Encryption Standard (AES) [19] is one of the most widely used secret key ciphers in embedded devices. It is notoriously difficult to crack with cryptanalytic methods. The best-known classical attacks on AES are based on differential, linear, algebraic, and meet-in-the-middle techniques, but they can only break a reduced number of rounds and require a large amount of data and time [8]. However, side-channel analysis (SCA) has demonstrated vulnerabilities in its software implementation. SCA attacks exploit unintended leakage from the system or hardware, which can expose sensitive information about the encryption algorithm. By analyzing timing, electromagnetic radiation, power consumption, or sound emissions, an attacker can retrieve the secret keys used in encryption algorithms.

Kotcher introduced the concept of breaking cryptographic algorithms through side-channel attacks in 1996 [32]. The initial focus was measuring the computation time algorithms like Diffie-Hellman or RSA require for modular factorization. This approach aimed to discover secret keys and ultimately compromise the security of such systems. The first side-channel attacks included non-profiled Simple Power Analysis (SPA) and Differential Power Analysis (DPA) on the Data Encryption Standard (DES), which preceded AES [33]. Subsequently, researchers introduced Correlation Power Analysis (CPA) to break AES [11].

Moreover, profiled attacks, more powerful than SPA and DPA, operate under the assumption that the attacker possesses a cloned device identical to the victim's and can acquire a model of the device. This model is then used to break the encryption algorithm. One such attack is the Template Attack (TA) [14].

In response to the emergence of side-channel analysis (SCA) on AES, various algorithmic-level remedies have been developed. One commonly employed countermeasure is masking, which introduces randomization to the relationship between sensitive intermediate values and the corresponding side-channel leakages. Another significant defense mechanism involves incorporating desynchronization by introducing random delays into the power usage during the process. The rise of ML-based SCA techniques has prompted the development of countermeasures. Deep Learning, in particular, has proven to be highly effective in breaking implementations protected by masking or desynchronization [56].

Profiled side-channel attacks play a crucial role in assessing the security of cryptographic algorithm implementations, particularly in worst-case scenarios. However, these attacks rely on having an identical or closely matched reference device to the target device. This requirement can be limiting in real-world

settings. In contrast, non-profiled side-channel attacks offer increased realism and practicality in certain scenarios. They are applicable when the attacker lacks access to a profiling device or cannot make assumptions about the implementation or key of the target device. Non-profiled attacks need the development of novel techniques and metrics to exploit side-channel information without prior knowledge or training [67, 17]. Exploring non-profiled attacks can lead to a deeper understanding of the underlying mechanisms of side-channel leakage and the development of potential mitigation strategies [41].

In 2019, Benjamin Timon proposed the differential deep learning analysis (DDLA), the first side-channel attack technique that uses deep learning in a non-profiled approach [67]. The technique recovers the secret key using the phenomenon of deep learning metrics. This approach trains one neural network for each key candidate and uses them as distinguishers: the one showing the largest accuracy during training corresponds to the correct key candidate. However, the proposed technique made it difficult to observe the results from the intermediate process since the neural networks had to be retrained repeatedly as the cost of learning increased with the key size.

Researchers proposed a recent novel non-profiled side-channel attack utilizing multi-output regression neural networks (MOR) [17]. This technique allows for the simultaneous estimation of all key hypotheses during a single training process, leading to a substantial improvement in the computational performance of non-profiled deep learning-based side-channel attacks. They introduced two MOR variants: MOR-MLP based on multi-layer perceptron and MOR-CNN based on convolutional neural networks. These models specifically target masking and desynchronization-protected schemes.

However, neither DDLA's nor MOR's capabilities of generalization and performance have been evaluated against multiple datasets. Moreover, none (DDLA) or very few experiments (MOR) are conducted for hyperparameter tuning of these networks. To that extent, in this work, the following questions will be answered:

1. How can we review and quantify the performance of the available deep learning-based non-profiled side-channel analysis methodologies?
2. Can we derive a new methodology for deep learning-based non-profiled side-channel analysis that outperforms the state-of-the-art work?

The rest of this thesis is structured as follows: Chapter 2 gives a detailed overview of the fundamental background information needed to understand this research. Next, Chapter 3 looks at the previous work done by the community that is relevant to this study. Following that, Chapter 4 explores the first research question, while Chapter 5 investigates and answers the second research question. Finally, Chapter 6 wraps up the thesis by providing concluding remarks and summarizing the main findings.

Chapter 2

Background

This chapter provides the theoretical background for this research. We start by defining the basic concepts of Machine Learning (ML), a subfield of Artificial Intelligence (AI) that focuses on enabling computers to learn from data and make predictions. Subsequently, we look into Deep Learning (DL) algorithms, a Machine Learning technique that uses neural networks to extract features from data and provide high-level representations. Moreover, we discuss the Advanced Encryption Standard (AES), a widely used symmetric key encryption algorithm. Lastly, we present these concepts within the field of side-channel analysis.

2.1 Machine Learning

Machine learning is a topic of study focused on understanding and developing *learning* methods that use data to enhance performance on a particular set of tasks [47]. In recent years, there has been an impressive rise in ML applications that have achieved significant success [20]. This surge marks an essential milestone in the field of technology. Notable examples of such applications include the creation of advanced data mining programs that can effectively detect fraudulent credit card transactions [13] and the introduction of self-driving cars that undergo thorough training to ensure safe navigation on public roads [1].

ML algorithms aim to build models out of input data, also called *training data*, that are used subsequently to make decisions. The input instances are often represented by vectors $x = (x_1, x_2, ..., x_n)$ composed of n features or *points of interest* (POI). When classifying, the task is to define a function $f(x) = y$ where y is the prediction output of x.

Mitchell [47] described ML as follows: ”A computer program is said to learn from experience E concerning some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E“. The tasks T describe how the features x should be processed. The most relevant machine learning tasks are *classification* and *regression*, as further explained in 2.1.1. In 2.1.2, we present different types of performance metrics P that aim to quantify the achievement of the ML algorithm. Finally, 2.1.3 describes the types of experience E that an ML algorithm has during the learning process.

2.1.1 ML Tasks

Classification

In a classification task, the algorithm requires the input vectors $x = (x_1, ..., x_N)$, a finite number m of K categorical possible labels y and aims to find the function $f : \mathbb{R}^n \rightarrow \{1, 2, ..., m\}$ such that $f(x) = y$. In ML, the classification task is reduced to finding the most *probable* class that the input belongs to

[66]. That is, finding the *a posteriori probability* $p(y_j/x)$. The algorithm shall then output the class j that maximize the a posteriori probability: $\max_{1 \leq j \leq |K|} p(y_j/x)$.

Regression

Different from classification, the output y in a regression task can take continuous values and is not based on a probability distribution function as before. The ML task, in this case, is to find the function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Then, the algorithm finds a numerical value for each input vector x. When talking about regression, the most famous instance is linear regression. Linear regression assumes that the function f is represented as $f(x) = w^T * x + w_0$, where w is called the *weight vector* and its dimension is N, the same as for the input x.

2.1.2 Performance Metrics

To quantify the performance of an ML algorithm, one has to choose metrics relevant to the task being learned. Therefore, the evaluation of the models is often done for unseen data (not the data used for training), also called *test* data.

One of the most straightforward evaluation metrics for a model is the *accuracy*. It measures the percentage of correctly labeled data points in the data set and is calculated as follows:

$$\text{accuracy} = \frac{\text{\# correct predictions}}{\text{\# data points}}$$

For most tasks, measuring the accuracy is equivalent to measuring the *error loss*. In other words, a high accuracy indicates a low loss value. However, in the case of regression tasks, accuracy is not a suitable metric for evaluation because regression ML algorithms predict continuous or numeric values, not labels. The error (loss) function is the metric used to quantify the model's performance for these tasks. There are several ways to represent the loss. We discuss some of the available functions next.

The 0-1 Loss Function

One method for representing the error function is the $0-1$ loss. It is used for classification tasks and counts the number of misclassified data points. The total $0-1$ loss is then calculated as the sum of the individual loss values across all examples in the dataset.

$$L(y, \hat{y}) = \begin{cases} 0, & \text{if } y = \hat{y} \\ 1, & \text{otherwise} \end{cases}$$

Cross Entropy Loss

The *cross-entropy loss* function is a metric used in classification problems. Here, y_i and $\hat{y_i}$ are the true probability and the predicted probability, respectively, that the input data point x corresponds to the label i. The cross-entropy loss penalizes data points that are wrongly classified with high confidence. The formula can be found below. It is computed as the average dissimilarity between predicted probabilities and true labels. This dissimilarity is computed as the negative logarithm of the predicted probability for the correct class.

$$L(y, \hat{y}) = -(y_i * \log(\hat{y_i}) + (1 - y_i) * \log(1 - \hat{y_i}))$$

Mean Squared Error

The *mean squared error* (MSE), also known as the L2 loss, is widely used in regression tasks since it measures the numerical difference between true and predicted values. MSE calculates the average of the squared differences between the predicted and actual values of a dataset.

$$L(y, \hat{y}) = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}$$

Mean Absolute Error

The *mean absolute error* (MAE), also known as the L1 loss, is a loss function that calculates the average of the absolute differences between the predicted and actual values of a dataset. Similarly to mean squared error, it is often used in regression tasks. The difference is that MAE is less sensitive (or more robust) to outliers than MSE because it gives equal weight to all errors, regardless of their magnitude. In contrast, MSE amplifies the effect of larger errors due to squaring.

$$L(y, \hat{y}) = \frac{\sum_{i=1}^{N}|y_i - \hat{y}_i|}{N}$$

Huber

The *Huber error* function is a loss function used in ML for regression problems. It is a hybrid of the mean squared error and mean absolute error loss functions, which provides a balance between their benefits. It behaves like MSE for smaller errors and like MAE for larger errors. By introducing a threshold parameter δ, the Huber loss transitions between the two, as we see in the formula below. The Huber loss function is more robust to outliers than MSE because it transitions to behave like MAE for larger errors, striking a balance between the two and compromising sensitivity to outliers.

$$L(y, \hat{y}) = \begin{cases} 0.5 * (y - \hat{y})^2, & \text{if } |y - \hat{y}| \leq \delta \\ \delta * (|y - \hat{y}| - 0.5 * \delta), & \text{otherwise} \end{cases}$$

Log Cosh

The *logarithmic hyperbolic cosine* loss function is a smooth approximation of the Huber loss function that combines elements of both logarithmic and hyperbolic cosine functions, as we see in the formula below. It is commonly used in regression tasks to balance the robustness of the Huber loss and the smoothness of the mean squared error (MSE). The LogCosh loss is differentiable, continuous, and less sensitive to outliers compared to MSE.

$$L(y, \hat{y}) = \sum log(cosh(\hat{y}_i - y_i))$$

Tukey's Biweight

Tukey's biweight loss function [7] is often used in regression tasks. As shown in Fig. 2.1, it is similar to Log Cosh. However, it is even less sensitive to outliers and conceals them during backpropagation in deep-learning tasks [2]. That is, it down weights the impact of outliers: if the residuals are smaller than -4 or larger than 4, we see that the loss starts decreasing towards $-\inf$ or $\inf$. We define Tukey's loss function using a tuning constant, usually denoted as c, determining the threshold for outlier detection. The loss function assigns smaller weights to the samples with larger residuals beyond the threshold. According to current research [2], a sensible value for the constant c is $c = 4.685$.

$$L(\hat{y}, y) = \begin{cases} \frac{c^2}{6} * (1 - (1 - (\frac{(\hat{y}-y)}{c})^2)^3 & |\hat{y} - y| < c \\ \frac{c^2}{6} & \text{else} \end{cases}$$

Z-score Normalized Loss Functions

The Z-score normalization of loss functions involves transforming the input values of the loss function (actual and predicted labels) to have a mean of zero and a standard deviation of one. This normalization technique helps compare and interpret loss values across different models or datasets by removing the scale and shifting effects. It is useful when working with loss functions sensitive to the scale of the loss values [52]. We give the formulas for the Z-score normalization of MSE, MAE, Huber, LogCosh, and Tukey's Biweight loss function below.

Z-score MSE

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (\frac{y_i - \mu(y_i)}{\sigma(y_i)} - \frac{\hat{y}_i - \mu(\hat{y}_i)}{\sigma(\hat{y}_i)})^2$$

Z-score MAE

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} |\frac{y_i - \mu(y_i)}{\sigma(y_i)} - \frac{\hat{y}_i - \mu(\hat{y}_i)}{\sigma(\hat{y}_i)}|$$

Z-score Huber

$$L(y, \hat{y}) = \begin{cases} 0.5 * (\frac{y_i - \mu(y_i)}{\sigma(y_i)} - \frac{\hat{y}_i - \mu(\hat{y}_i)}{\sigma(\hat{y}_i)})^2, & \text{if } |\frac{y_i - \mu(y_i)}{\sigma(y_i)} - \frac{\hat{y}_i - \mu(\hat{y}_i)}{\sigma(\hat{y}_i)}| \leq \delta \\ \delta * (|\frac{y_i - \mu(y_i)}{\sigma(y_i)} - \frac{\hat{y}_i - \mu(\hat{y}_i)}{\sigma(\hat{y}_i)}| - 0.5 * \delta), & \text{otherwise} \end{cases}$$

Z-score Tukey's Biweight

$$L(\hat{y}, y) = \begin{cases} \frac{c^2}{6} * (1 - (1 - (\frac{(\frac{\hat{y}_i - \mu(\hat{y}_i)}{\sigma(\hat{y}_i)} - \frac{y_i - \mu(y_i)}{\sigma(y_i)})}{c})^2)^3 & |\frac{y_i - \mu(y_i)}{\sigma(y_i)} - \frac{\hat{y}_i - \mu(\hat{y}_i)}{\sigma(\hat{y}_i)}| < c \\ \frac{c^2}{6} & \text{else} \end{cases}$$

Z-score Log Cosh

$$L(y, \hat{y}) = \sum log(cosh(\frac{\hat{y}_i - \mu(\hat{y}_i)}{\sigma(\hat{y}_i)} - \frac{y_i - \mu(y_i)}{\sigma(y_i)}))$$

2.1.3 Learning Process

Depending on the available data, ML algorithms can have different learning processes and are divided into three main categories: *supervised*, *unsupervised*, and *reinforcement learning*. Supervised learning is based on a model with available feature vectors x and their corresponding labels y. On the other hand, in unsupervised learning, labels are unavailable, and the algorithm aims to find a structure in the data by identifying common features among the feature vectors. Finally, reinforcement learning deals

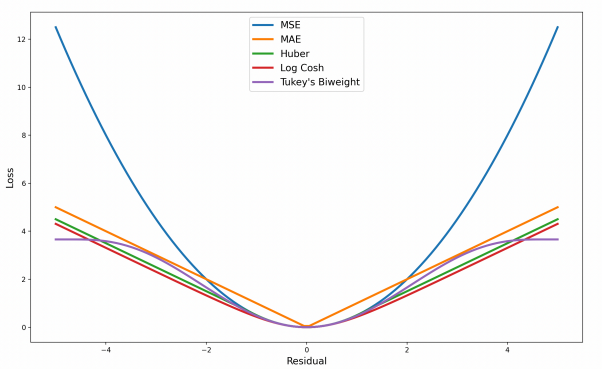


Figure 2.1: Comparison of regression loss functions. On the horizontal axis, we depict the residual: the difference between true and predicted labels. This corresponds to $y - \hat{y}$ in the formulas.

with the idea of agents in a specific environment and aims to maximize the cumulative reward of the agents given some taken actions.

2.1.4 Multi-output Learning

Traditional ML tasks were initially designed to solve simple single-output problems, such as predicting the energy consumption of an engine based on temperature or binary classifying mail as spam or not spam [77]. However, the challenges of learning tasks have increased, and multi-output architectures have emerged as solutions. Instead of predicting only one output value, we use an input data point to predict with a set of labels simultaneously. Same as in classical learning, there are two types of tasks: multi-output classification and multi-output regression [9].

The input instances are represented in multi-output learning as n – features vectors $X = (x_1, ..., x_n)$ and the corresponding labels as $Y = (y_1, ..., y_d)$ where d is the number of the possible class labels. The task is to find a function $f : X \to Y$ for regression tasks and a function $f : X \to 2^Y$ for classification tasks where categorical values denote the labels.

In 2009, Tsoumakas and Katakis [69] introduced *multi-label classification* and provided a comprehensive overview of the different methods and techniques used for multi-label classification tasks. They classified multi-label classification approaches into two main categories. First, there is the *problem transformation methods* that transform the multi-label classification problem into one or more single-label classification or regression problems. The second category is *algorithm adaptation methods*. These methods, on the other hand, focus on extending specific learning algorithms to handle multi-label data directly. Instead of transforming the problem into a single-label format, these methods modify existing algorithms or develop new algorithms tailored to handle multi-label scenarios. In this research, we are interested in the second category of multi-output learning.

In 2014, Zhang *et al.* [81] conducted a review on multi-output learning and classified multi-output learning into three categories based on the *correlations* between labels. The first technique is *First-order strategy*. Here, it is assumed that the labels are not correlated, and the multi-output task is then decomposed into several independent classification/regression problems. The second approach is *Second-order strategy*, where it is assumed that the labels are pairwise correlated, e.g., label co-occurrences [22]. Finally, there is the *Higher-order strategy*: in this case, the labels are highly correlated.

2.1.5 Ensembles in ML

Ensemble learning in ML is a technique that combines multiple individual models (base learners) to make more accurate predictions or classifications than any individual model achieves on its own [51]. The idea behind ensembles is that the combined knowledge and diverse perspectives of multiple models can improve overall performance and reduce the risk of overfitting [58].

There are several ensemble methods, including bagging, boosting, and stacking.

Bagging stands for Bootstrap Aggregating. It involves training multiple base learners independently on different subsets of the training data, created by random sampling with replacement (bootstrapping) [10]. Random sampling with replacement is a method of selecting a sample from a population where each unit can be chosen more than once. This means that after selecting a unit randomly, it is returned to the population and has the same chance of being selected again. Each base learner produces its prediction, and the final prediction is obtained by aggregating the predictions of all base learners. Aggregation can be done through majority voting (for classification) or averaging (for regression). Popular bagging algorithms include Random Forests, which are ensembles of decision trees, and Extra-Trees, which are similar to Random Forests but use random splits.

Boosting is an iterative ensemble technique where base learners are trained sequentially, with each subsequent learner focusing on the samples that the previous learners have struggled with [21]. Boosting aims to improve the overall ensemble's performance by reducing bias. The final prediction is typically obtained by combining the predictions of all base learners using weighted voting or weighted averaging, where the weights are assigned based on the base learners' performance. Notable boosting algorithms include AdaBoost, Gradient Boosting, and XGBoost.

Stacking also known as stacked generalization [72], involves training multiple base learners and then training a meta-learner (also called a blender or aggregator) to make the final prediction using the predictions of the base learners as input. The base learners can be different algorithms or variations of the same algorithm. Stacking allows the meta-learner to learn how to best combine the base learners' predictions. It can be performed in multiple stages, where the outputs of one stage serve as input to the next stage.

Ensembles often achieve higher predictive performance than individual models by combining the strengths of multiple models and mitigating their weaknesses. Moreover, ensembles are more robust to outliers and noise in the data, as the individual errors of the base learners cancel each other out to some extent. Furthermore, ensembles better generalize to unseen data by capturing different aspects of the problem space. By combining multiple models, ensembles can also reduce the risk of overfitting by avoiding excessive reliance on a single model's idiosyncrasies. However, ensembles come with increased computational complexity and memory requirements due to training and storing multiple models. They also require careful tuning of hyperparameters and may be more challenging to interpret compared to individual models.

Overall, ensembles have proven to be powerful techniques in ML, and their success is evident in various real-world applications [46, 80].

2.2 Deep Learning

In many real-world artificial intelligence applications, the difficulty comes from the fact that even a small variance in the data can affect the learning process and, thus, the outcome. When classifying an object in a picture, the light in the room or the viewing angle should not influence the outcome in any way - it is still the same object.

However, it is not trivial to determine which features are important and which are not. That is, extracting abstract features from raw data (data that has not been processed before) needs refined learning mechanisms, such as the human mind. *Deep Learning* can solve these difficulties by building complex systems out of simpler systems. The architecture used in deep learning algorithms is therefore based on neural networks composed of one or multiple layers. Each layer simplifies the original model and corresponds to a simpler underlying concept of the data.

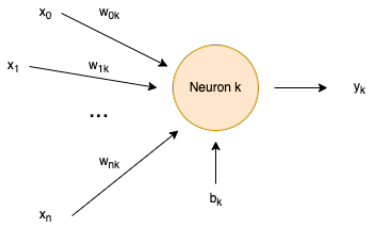


Figure 2.2: Unit of the neural networks: the neuron.

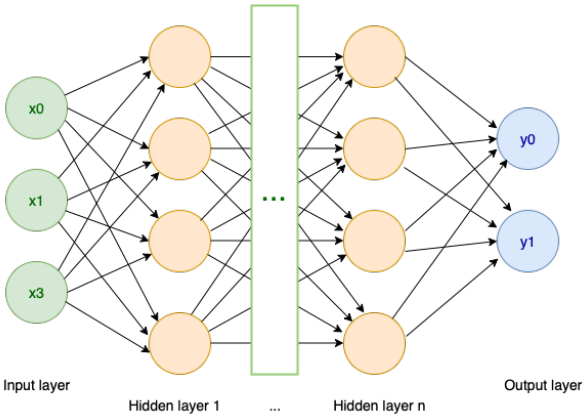


Figure 2.3: Multilayer perceptron architecture: one input layer, several hidden layers, and one output layer.

2.2.1 Neurons

Neural networks are based on layers of units called *neurons*. Initially introduced in 1943 by McCullough and Pitts [45], the neural networks were not organized into layers nor trained, but it was shown that ”any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes“. In other words, it was proven that such a network can compute any function a computer can.

As previously mentioned, neural networks comprise one or several layers of neurons. Each neuron takes an input and produces an output sent as input to one or more neurons in the following layer [16]. The output of the neuron is calculated by summing up the weighted inputs. These weights are given by the connections from the input to the neuron. Finally, one bias term is added to this sum, as we see in Figure 2.2. The formula below calculates the output of neuron k:

$$y_k = \sum_{i=0}^{n} w_{ik} * x_i + b_k$$

2.2.2 Multilayer Perceptrons

Multilayer perceptrons (MLPs) ”are the quintessential deep learning models“ [24]. An MLP comprises one or multiple fully connected layers of neurons. The first layer is called the *input layer*, the last layer is the *output layer* that predicts the value of the input data point, and all layers in between are called *hidden layers*. We present an MLP network in Fig. 2.3.

An MLP is a feed-forward neural network. That is, information flows from the input x through the intermediate computations that happen in the hidden layers and, finally, to the output layer. There is no computation going backward. Each neuron in the hidden layer i applies a function f_i to the input it receives. This function is called the *activation* function. In theory, it is possible to apply different activation functions to neurons in the same layer. However, this is not done in practice. Given an MLP

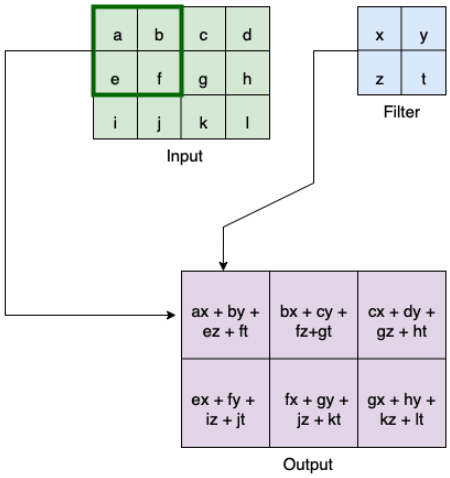


Figure 2.4: The convolution linear operation between 3 x 4 input and 2 x 2 filter.

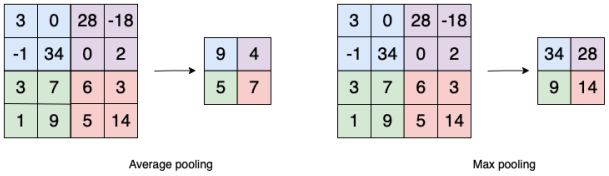


Figure 2.5: The average and max pooling operations.

with n hidden layers, the output is calculated as follows:

$$f(x) = f_n(...(f_1(f_0(x)))) = y$$

Different types of activation functions can be used in an MLP. The most important ones to note for this research are:

- Rectified Linear Unit (ReLU) [28] is one of the most used activation functions in neural networks. This function simply applies the absolute value to the input:

$$\text{ReLU}(x) = |x| \text{ or } f(x) = \max\{0, x\}$$

- Softmax, or *normalized exponentiation* function [6] converts an input vector of K components into a probability distribution of K possible outcomes. This is:

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1,..,K} e^{x_j}}$$

2.2.3 Convolutional Neural Networks

Convolutional Neural Network (CNN) [5] is a type of neural network that uses a mathematical linear operation called *convolution* in at least one of its layers. The convolution operation is performed by applying the dot product between the input and a set of weights; the latter is called *filter* or *kernel*. The filter size is smaller than the input size, allowing the multiplication to happen multiple times at different input points. Fig. 2.4 shows a convolution operation for an input of size 3 x 4 and a filter of size 2 x 2. As before, an activation function is applied to the output, typically ReLU.

Among the convolution layers, a CNN often has *pooling* layers. Pooling layers have the purpose of reducing the data size from one layer to another. The most used types of pooling are *average* and *max* pooling. We present both in Fig. 2.5.

The best advantage of the pooling layer is that they have the property of being invariant to translations. If one translates the input by a small amount, most values of the pooling layer output do not change. We present an example in Fig. 2.6. Three out of four pooling layer outputs did not change after translating the input data. This is a very helpful aspect if the prediction is more influenced by the presence of the feature in the data point, not specifically its position. For example, determining whether a picture contains a flower is not determined by the position of the flower in that picture: whether it is on the left or right, upwards or downwards, it is still the same object.

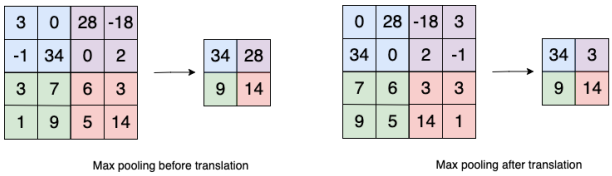


Figure 2.6: The max pooling operation after translation of input data.

2.2.4 Transfer Learning

In deep learning, neural network architectures can quickly get many layers and are thus time-consuming to train from scratch [34]. To solve this challenge, one can use *transfer learning*. In transfer learning, networks already trained on known datasets and experimentally showed good performance are further used for predicting the outputs on other datasets. For example, in the field of image recognition, there are the well-known ResNet-50, ResNet-101, and ResNet-152 models that showed great performance in different competitions [25].

Transfer learning relies on the idea that the feature learning process is similar for multiple datasets. Once there is a trained network for a large enough dataset that shows good generalization, it can be also used for other (smaller) datasets [71].

One form of transfer learning is fine-tuning. Fine-tuning is a concept of transfer learning that requires a bit of learning [68, 63]. It works by *freezing* the first part of a network that takes care of feature extraction and lets the network train further only in the last layers. The final layers from the original network can also be replaced to match with a different output size or type.

2.2.5 Data augmentation

Deep learning models typically require much training data to effectively learn complex patterns and generalize to unseen examples. However, obtaining a massive labeled dataset can be expensive or impractical in many real-world scenarios. Data augmentation (DA) helps overcome this limitation by artificially expanding the training dataset, effectively increasing the diversity and quantity of training samples available for model training.

Another aspect to consider is that deep learning models are prone to overfitting. This means they memorize the training data instead of learning the underlying representations. Data augmentation can be used as a regularization technique by introducing variations in the training data.

In the context of side-channel analysis, Cagli *et al.* [12] employed DA and convolutional neural networks to counter the effects of clock jitter on the attack traces. The authors employ two data augmentation methods referred to as Shifting Deformation and Add-Remove Deformation. The Shifting Deformation technique mimics a random delay countermeasure, while the Add-Remove Deformation technique simulates a clock jitter effect.

In this work, we use two data augmentation techniques. Similarly to Cagli *et al.* [12], we use shifting, as well as Gaussian noise. Algorithm 1 and Algorithm 2 show their implementations, respectively.

Algorithm 1 Shifting DA

Require: trace, label
augmented_trace ← []
augmented_label ← label
n ← len(label)
shift ← $random(-5, 5)$
if shift > 0 **then**
augmented_trace[$0 : n -$ shift] ← trace[shift : n]
augmented_trace[$n -$ shift : n] ← trace[0 : shift]
else
augmented_trace[0 : abs(shift)] ← trace[$n -$ abs(shift) : n]
augmented_trace[abs(shift) : n] ← trace[$0 : n -$ abs(shift)]
end if

Algorithm 2 Gaussian noise DA

Require: trace, label
augmented_trace ← []
augmented_label ← label
n ← len(label)
noise ← $normal(0, 1, n)$
augmented_trace ← trace + noise

2.3 Advanced Encryption Standard

The Advanced Encryption Standard (AES) was created by the U.S. National Institute of Standards and Technology (NIST) for the encryption of electronic data in 2001 [62]. It is one of the most used symmetric key cryptosystems in IoT devices [50]. AES takes as input a fixed 128-bit block of the plaintext and a 128, 192, or 256 bits key. The key length specifies the number of rounds used on the plaintext to obtain the final ciphertext: 10, 12, or 14 rounds respectively. The unit 128-bit block that AES operates on can be seen as an 8 x 8 matrix having 1-byte entries. There are four primitive operations that the cryptosystem uses: `SubBytes`, `ShiftRows`, `MixColumns`, and `AddRoundKey`. The algorithm works as follows:

1. `KeyExpansion` - the round keys are derived from the original key.
2. `AddRoundKey` - each input block byte is `xor-ed` with a byte of the first round key
3. For 9, 11, or 13 rounds, do the following:
   (a) `SubBytes`- each input block byte is substituted according to an S-box
   (b) `ShiftRows` - a cyclic shifting of rows of the input block
   (c) `MixColumns` - linear mixing operation applied on the columns of the input block
   (d) `AddRoundKey` - each input block byte is `xor-ed` with a byte of the next round key
4. Final round:
   (a) `SubBytes`
   (b) `ShiftRows`
   (c) `AddRoundKey` - each input block byte is `xor-ed` with a byte of the last round key

In terms of the security of AES, the best results that have been obtained for key recovery require $2^{126.0}$ operations for AES-128, $2^{189.9}$ for AES-192 and $2^{254.3}$ for AES-256 [64]. The AES algorithm, when used correctly, cannot currently be attacked in a classical manner that would permit someone without access to the key to read data that has been encrypted. However, other types of attacks can easily recover the secret key for AES and are thoroughly discussed in 2.4.

2.4 Side-Channel Attacks

As explained in 2.3, currently, there is no efficient full attack on AES that can recover the secret key. However, side-channel analysis can break the key by analyzing the *leakage* of the implementation of the cryptosystem on an (IoT) device. That is, one breaks AES by exploiting not the vulnerabilities of the cryptosystem itself but the vulnerabilities of the implementation or hardware. The leakage exploited when breaking AES is the energy (power) consumption of the device that is running AES.

In Section 2.3, we briefly explained how `AddRoundKey`, a primitive operation used in AES, applies the `xor` operation on the input block and the secret key. That is, the key determines how many bit flips happen at that point during the execution. Bit flips have a great impact on the power consumption of a device; thus, one can investigate the correlation between this leakage and different key candidates to retrieve the correct secret key. When designing an SCA against AES, one has to model this leakage. The most frequently used leakage models are the *Hamming Weight (HW)*, *Identity (ID)*, *Most Significant Bit (MSB)*, and *Least Significant Bit (LSB)*.

2.4.1 Non-profiled Analysis

In 1999, Kocher *et al.* [33] introduced the first side-channel attacks that use the leakage in the power consumption of AES implementations were Simple Power Analysis (SPA) and Differential Power Analysis (DPA).

SPA graphically inspects the power traces over time and can retrieve sensitive information about an algorithm. Although it cannot retrieve the secret key of AES, it has been used to identify the number of rounds; thus, it retrieves the key size [40].

On the other hand, DPA can break AES [33] by taking advantage of `SubBytes` and `AddRoundKey`. It works by first gathering many power traces and the corresponding plaintexts. The next step is to compute the expected leakage given each possible key candidate byte. In case of key length 128 bits, one has to calculate the *intermediate values*: $I_{j,i} = \text{S-box}[X_{j,i} \oplus K_i]$ for each $K_i \in \{0, 1, 2, ..., 255\}, \forall j \in$ plaintexts, $\forall i \in \{0, 1, 2, ..., 15\}$, where S-box is the substitution box used by `SubBytes`. Consequently, we split these traces into two subsets based on a chosen function f and calculate the average of each subset's intermediate values. This is also called *Difference of Means* because it computes the difference in the two subsets' averages, which statistically estimates the significance of a hypothesis. The hypothesis used for DPA on AES is that the output of the encryption produces small variations in the power traces. That is, for the correct key candidate K_i, the test shows spikes (variations) when examining the mean of the subsets.

2.4.2 Profiled Analysis

Profiled side-channel attacks have proven more powerful than the classic SPA and DPA attacks. Different from non-profiled attacks, the profiled analysis assumes that the attacker has access to a copy of the victim's device, hence access to an unlimited number of power traces. In the first phase of a profiled attack, called the *profiling* phase, the attacker gathers N power traces and corresponding plaintexts, and since they own the device, the secret key k_j is known for each plaintext j.

Moreover, to build the model that we use for the attack, we have to calculate the leakage values for each power trace given the known secret key $L(x_i, k), \forall i \in \{1, 2, ..., N\}$. Using this model, the attacker then gathers M traces from the victim's device and can find the corresponding key k_v.

The first type of profiled attack was the Template Attack [14]. It assumes that per class, the traces follow a multivariate Gaussian distribution. At that time, it was the most powerful side-channel attack against AES since it could break even protected implementations of the cryptographic algorithm.

Nonetheless, there are ML and DL attacks that can easily break AES implementations protected by even more countermeasures. Research has demonstrated that deep learning is the most effective method to perform side-channel attacks [3], although some challenges arise when using this method-

ology. Some of them include hyper-parameter tuning, which can greatly increase the overall complexity of the attacks, or models that work on specific datasets but not on others.

2.4.3 Evaluation Metrics

In the case of SCA, the goal is to find the secret key that breaks the security of the cryptosystem. That is, the output of an ML attack consists of a key guessing vector $v = (k_1, k_2, ..., k_{|K|})$ where the keys are ordered according to the probabilities of being the correct ones: the first one is most probably the correct one. To that extent, to assess the performance of the attack, two important evaluation metrics can be considered: *Guessing Entropy (GE)* and *Success Rate (SR)*.

Guessing Entropy

In 1994, Massey [44] proposed the Guessing Entropy metric in the context of guessing the discrete value of a variable X in several trials, and it measures the average number of trials needed to obtain the correct value X. This is similar to retrieving the secret key of AES in SCA and can also be applied in this setting. That is, it measures the number of key candidates to test for correctness after the attack to gain insight into its complexity, compared to the complexity of a brute-force key guess [61]. If the first guessed key k_1 is the correct one, the attack has the best-guessing entropy, and its value is 1.

Success Rate

The (first-order) success rate of an attack measures the probability that the correct key has the first position in the guessing vector defined above [61]. One can also define an o order SR to measure the probability that the correct key can be found in the first o values of the guessing vector.

2.4.4 Countermeasures

With the development of SCA against AES, there has been an increase in designing countermeasures at both the implementation and hardware levels. Different types of countermeasures are discussed next.

Masking

Masking, which involves XOR-ing all the intermediate values of an implementation with random Boolean values, is a technique used to defeat some types of side-channel attacks [59]. SCA such as DPA become impractical as a result of the unexpected power consumption of the system. For AES, masking is done at the level of the `SubBytes` operation and masks the output of the S-box in the following way:

$$\text{S-masked} = \text{S-box}[x_i \oplus k_i] \oplus r_i$$

$$x_i = i\text{-th plaintext}$$

$$k_i = i\text{-th key}$$

$$r_i = i\text{-th mask}$$

Desynchronization

Another commonly used countermeasure for AES is desynchronization, also known as *random delays*. It adds randomness to the power traces with respect to the time domain [76] by shifting the values

of the power consumption. We see the pseudocode for adding desynchronization to a trace in Algorithm 3. This countermeasure increases the uncertainty of the attacker about the moment in time where computations have happened [6].

Algorithm 3 Adding desynchronization

Require: oldTrace, maxDesync ≥ 0
  newTrace $\leftarrow$ []
  $i \leftarrow 0$
  desync $\leftarrow random(0,$ maxDesync$)$
  while $i +$ desync $< len($oldTrace$)$ **do**
    newTrace$[i] \leftarrow$ oldTrace$[i +$ desync$]$
    $i \leftarrow i + 1$
  end while

2.4.5 Datasets

With the increase and development of the side channel analysis research for AES, several public datasets have become available with the purpose of consistent performance benchmarking. For this work, we consider four datasets.

ASCADf This dataset contains electromagnetic (EM) traces collected from an 8-bit AVR-based microcontroller that implements a first-order Boolean masked AES-128 [4]. In our experiments, we consider two versions of this dataset. The first version consists of the trimmed interval of 700 features that includes the second-order leakages of the third S-box in the first encryption round, which we refer to as ASCADf. We also run experiments with the raw measurements from the NOPOI scenario presented in [53], which contains 10 000 features, and we refer to it as `ASCADf raw` dataset. We attack the third key byte (the counting starts at 0), and our leakage model is:

$$Y(k) = S\text{-box}[\text{plaintext}_2 \oplus k] \oplus m_2$$

ASCADr This dataset was collected from the same measurement setup as ASCADf. However, the difference is that the encryption keys are randomized [4]. For this work, we use the part of this dataset commonly used as the test set in DLSCA, where all plaintexts are encrypted with the same key. We consider two versions of this dataset. The first version consists of the trimmed interval of 1 400 features that include the second-order leakages of the third S-box in the first encryption round, which we simply refer to as ASCADr. We also run experiments with the raw measurements from the NOPOI scenario from [53], which contains 25 000 features, and we refer to it as `ASCADr raw` dataset. We attack the third key byte (the counting starts at 0), and our leakage model is:

$$Y(k) = S\text{-box}[\text{plaintext}_2 \oplus k] \oplus m_2$$

AES_RD This dataset contains traces collected from a smart card with an 8-bit Atmel AVR microcontroller. As a countermeasure, random delays using the Floating Mean method were added to the traces to protect against simple side-channel attacks [15]. We attack the first key byte, and our leakage model is:

$$Y(k) = S\text{-box}[\text{plaintext}_0 \oplus k]$$

AES_HD This dataset targets an unprotected hardware implementation of AES-128 written in VHDL in a round-based architecture. Using a Xilinx Virtex-5 FPGA of a SASEBO GII evaluation board, side-channel traces were collected using a high sensitivity near-field EM probe positioned above a decoupling capacitor on the power line [31]. We attack the eighth key byte, and our leakage model is:

$$Y(k) = S\text{-box}^{-1}[C_7 \oplus k] \oplus C_3$$

The values C_i and C_j are two ciphertext bytes. The relationship between them is given by the ShiftRows operation.

Chapter 3

Related Work

In this chapter, we present and examine prior scientific research in the context of deep learning-based side-channel analysis (SCA) and unsupervised learning. Our discussion begins by reviewing the state-of-the-art models used in profiled deep learning-based SCA. Subsequently, we introduce the research that has been conducted in the area of non-profiled and unsupervised deep learning-based side-channel attacks. Lastly, we present and define the research questions based on the state-of-the-art findings.

3.1 Deep Learning in Side-Channel Analysis

Deep learning models have been widely used in side-channel analysis on AES since they can automatically extract and learn features from electromagnetic or power traces. State-of-the-art work in this field includes using different deep learning architectures, such as multi-layer perceptions, and convolutional neural networks

In 2018, Benadjila *et al.* [56] published the first open dataset as a common baseline for research in this field called ASCAD. The authors proposed a study of deep learning algorithms in the context of side-channel analysis and addressed the choice of hyper-parameters for convolutional neural networks. They experimentally proved that CNNs, together with data augmentation, can break protected AES implementations by various countermeasures. These results were further confirmed by Kim *et al.* [31]. The authors proposed a new CNN instance that can efficiently break AES and experimentally proved how adding noise to the power traces can significantly reduce the number of input traces for a successful attack.

Zaid *et al.* [78] proposed a methodology for building efficient CNN architectures by examining the effect that each hyperparameter has on the efficiency of the attack. The authors use heatmaps and weight visualization to analyze and determine which patterns are most influential during the training process, depending on the implemented countermeasure. Later, Wu *et al.* [75] proposed an automated hyperparameter tuning method (AutoSCA) that is based on Bayesian Optimization. Experimentally, it was proven that AutoSCA can produce efficient networks with better performance than state-of-the-art models.

3.2 Unsupervised Deep Learning in Side-Channel Analysis

Unsupervised learning aims to learn a model without knowing the output values or labels during the training phase. This is the case with all non-profiled attacks, such as the classic DPA. As mentioned in 2.4.1, DPA partitions the power traces into two subsets according to guessed intermediate values for each key candidate. The labeling is done by using a statistical distinguisher that measures the

consistency of each subset. There are several ways to perform DPA considering different statistical metrics: Difference of Means [33], Mutual Information [23, 55, 70], or Variance Test [60].

Most of the research in deep learning-based side-channel analysis assumes a profiled environment; therefore, they are supervised methods [27, 39, 38, 79, 35]. However, in 2019, B. Timon [67] was the first to design an unsupervised deep learning-based attack that works similarly to DPA. However, the distinguisher in Timon's method uses deep learning, namely neural networks, as is described next. This approach is referred to as differential deep learning analysis (DDLA).

Timon's method works by training 256 neural networks, one for each possible candidate key byte, using the power traces as inputs and the intermediate values corresponding to a chosen leakage model for that key candidate as output (labels). The research proved that for the correct key, the neural network shows significantly higher accuracy during training or, equivalently, smaller error loss. The study considers two datasets: ASCADf synchronized fixed key and traces collected from the Chip-Wisperer-Lite Board by the author. In terms of the architecture, a multi-layer perceptron was used for classifying synchronized fixed-key ASCADf and Chip-Whisperer traces and a convolutional neural network for 20-desynchronized Chip-Whisperer traces. This technique, however, uses the LSB and MSB leakage models, and the author states that it does not work when using the Identity leakage model.

In 2020, Ramenzapour *et al.* [57] proposed a different kind of unsupervised deep learning-based side-channel analysis. More specifically, the authors proposed an LSTM autoencoder for the feature extraction of the power traces. The experiments proved that autoencoders, even in the context of a classic non-profiled attack such as DPA, can significantly improve performance, requiring 10 times fewer power traces for a successful attack. Similarly to Timon [67], the work employed the features extracted by the autoencoder to learn a leakage model for each key candidate using an MLP architecture. The features are then clustered based on the estimated leakage model for each key candidate, and the key corresponding to the highest inter-cluster difference is chosen as the correct key. This work, however, does not employ any experiments on the datasets that are used in side-channel analysis research.

Kwon *et al.* [36, 37] proposed an improvement to the existing non-profiled side-channel attacks, more specifically Timon's work [67], by making use of deep learning-based unsupervised autoencoders. This method uses unsupervised deep learning to improve feature extraction by training a neural network architecture with noisy and noise-reduced power traces as input-output pairs. The authors experimentally proved that unsupervised deep learning-based autoencoders increase performance, especially in the case of higher-order non-profiled side-channel attacks where the intermediate values of the traces are not known.

In 2021, Won *et al.* [73] built upon Timon's approach and converted the input traces into a picture-like format. Furthermore, they argued that using a validation set and considering its accuracy when determining the correct key is more appropriate than considering the accuracy of the training data. That is because, in deep learning, overfitting can often happen; thus, it can be the case that learning with incorrect keys can be faster than learning with the correct key. This approach retrieved the correct key even with 100 and 50-desynchronized ASCADf datasets.

In 2022, Hoang *et al.* [26] proposed a novel methodology to perform unsupervised deep learning-based side-channel attacks. Instead of training 256 different neural networks as Timon did in 2019, [67], the authors proposed a multi-output classification network that can predict simultaneously all 256 key hypotheses. This method uses two deep networks: a multi-layer perceptron and a convolutional neural network, respectively. Experimentally, it was shown that this method is 9 up to 30 times faster compared to DDLA.

Later, Do *et al.* [18] experimentally proved that not only the multi-output network can be used for classification but also for regression. The latter is the first architecture in unsupervised deep learning-based side-channel attacks that uses Identity labeling and can easily distinguish the correct key candidate.

The most recent work in multi-output learning in the context of side-channel analysis was published in 2023 [17]. Do *et al.* further worked on the initial idea of multi-output regression and showed that this approach breaks desynchronized traces and achieves results that are 40 times faster than DDLA and demonstrate a success rate improvement of at least 30%.

3.3 Research Questions

Based on the (lack of) current research in the field of unsupervised deep learning-based side-channel analysis, we formulate the research questions as follows:

Research Question 1

How can we review and quantify the attack performance of the available deep learning-based non-profiled side-channel analysis methodologies?

- What are the success rate and average guessing entropy of the available approaches for different datasets: ASCADf, ASCADr, AES_HD, AES_RD, and desynchronized ASCADf?
- Does data augmentation improve the performance of these approaches?
- Do machine learning ensembles improve the performance of these approaches?

Research Question 2

Can we derive a new methodology for deep learning-based non-profiled side-channel analysis that outperforms state-of-the-art work?

- What are the necessary elements for a new technique to build a non-profiled side-channel attack?
- How does it compare against the existing one in terms of different datasets: ASCADf, ASCADr, AES_HD, AES_RD, desynchronized ASCADf?
- Can we define an objective function that can be used in hyperparameter tuning?

Chapter 4

Review of Existing Unsupervised DL SCA

In this chapter, we answer the first research question by evaluating the existing approaches toward unsupervised deep learning side-channel analysis. We conduct a comprehensive study on Timon's work from 2019 [67], as well as on the method proposed by Do *et al.* in 2023 [17]. We investigate how these methods perform in terms of success rate and guessing entropy, given different variations in the number of traces, leakage models, neural network architectures, or the presence of countermeasures.

4.1 Motivation

Deep learning has been demonstrated to be effective in side-channel attacks, as evidenced by various studies [49, 48, 42]. However, there has been relatively little research on deep learning-based unsupervised side-channel analysis. Existing supervised deep learning approaches assume that the attacker has access to a copy of the victim's device, thus knowing the secret key during the profiling phase. Unfortunately, this is not always the case in real-life scenarios.

In 2019, B. Timon proposed a new approach to non-profiled deep learning-based side-channel analysis, which used neural networks as the distinguishers against fixed-key ASCADf for each key candidate [67]. In the remainder of this work, we refer to this Timon's approach as differential deep learning analysis (DDLA).

The first deep learning architecture used in the experiments conducted on synchronized fixed-key ASCADf in [67] is a Multi-Layer Perceptron (MLP). This MLP consists of two hidden layers, with the first containing 20 neurons and the second containing 10 neurons, which employ the Rectified Linear Unit (ReLU) activation function. The output layer of the MLP contains two neurons that are activated using the Softmax function. The leakage models used in this study were the Least Significant Bit (LSB) and Most Significant Bit (MSB). Throughout the remainder of the chapter, this architecture is referred to as DDLA-MLP.

The second deep learning architecture used in the experiments presented in [67] is represented by a Convolutional Neural Network (CNN). This CNN comprises two convolution layers, each containing four filters of varying sizes, with the first layer having 32 filters and the second having 16. To reference this particular model in the following sections of the paper, it is referred to as DDLA-CNN.

In the study conducted by Timon, the CNN model was shown to be effective when applied to 20-desynchronized ChipWhisperer traces. However, it is crucial to note that the technique's performance has not been assessed for higher levels of desynchronization or when used with different datasets. Moreover, the DDLA-MLP method has limitations, as it has not been tested for effectiveness against various datasets, training data size, leakage models, countermeasures, or neural network architectures. To overcome this limitation, we aim to conduct a systematic analysis to evaluate the performance

of Timon's unsupervised deep learning approach, DDLA. Our objective is to quantify the method's effectiveness in terms of success rate and guessing entropy when tested against different neural network architectures, varying the number of traces and epochs, and the influence of data augmentation on its performance.

In 2022, another type of deep learning-based unsupervised side-channel analysis was proposed in a study [18]. In this work, the researchers employed two multi-output neural network architectures to retrieve the correct key from a list of candidate keys. It is worth noting that this work was the first to demonstrate that the Identity leakage model could be used in unsupervised side-channel analysis when using multi-output regression (MOR). The MOR architecture used in this work has two shared dense layers of 500 and 700 neurons, respectively. The final layer consists of 256 neurons, corresponding to each key candidate. The activation function (ReLU), batch size (50), optimizer (Adam), loss function (MSE), as well as learning rate (0.001) are all set to certain values. Similarly, as in the case of DDLA, there were no experiments about how other neural network architectures affect the performance of the attack.

However, recently, the authors showed that MOR could be used with different architectures (MLPs and CNNs) [17], and the success rate is better than DDLA. The hyperparameter space used in this work is larger than their first work: 192 MLP architectures and 36 CNN architectures. They experimentally showed how the number of filters and batch size affect the performance of the attacks. Nonetheless, these hyperparameter spaces are still too small to thoroughly assess the success rate of this technique. Therefore, in this chapter, we aim to answer the following research question:

How can we review and quantify the attack performance of the available deep learning-based non-profiled side-channel analysis methodologies?

- What are the success rate and average guessing entropy of the available approaches for different datasets: ASCADf, ASCADr, AES_HD, AES_RD, and desynchronized ASCADf?
- Does data augmentation improve the performance of these approaches?
- Does the use of machine learning ensembles improve the performance of these approaches?

4.2 Experimental Setup

The experiments were designed and run with *python 3.10* using the deep learning *Tensorflow* library [65]. On top of that, we used *Keras*, a high-level API for TensorFlow [29].

4.2.1 DDLA

In each DDLA experiment, a total of 256 neural networks were trained, with each network corresponding to a single candidate key. For every plaintext and key combination, we computed the intermediate value for the target byte j, $L(\text{S-box}[p_i[j] \oplus k_i[j]])$ where L represents the function that models the leakage. These intermediate values were used as labels for the neural network. Throughout the training process, the network with the highest training accuracy or smallest training MSE loss was considered the most likely candidate for the correct key, as explained in [67]. In our experiments, we used LSB, MSB, and Hamming weight as leakage models. Furthermore, we also considered the accuracy and loss of a randomly chosen validation set. The high-level architecture of the attack can be viewed in Algorithm 4. The original MLP and CNN architectures used in [67] can be found in Figure A.1 and Figure A.3, respectively.

We run 500 models (MLPs and CNNS) randomly chosen from the hyperparameter space defined in Table 4.1 and Table 4.2. We compute the average guessing entropy as we vary the number of traces used in the attack, as well as the success rate as we vary the number of epochs. Furthermore, we used two data augmentation techniques (shifting and Gaussian noise) and plotted the success rate varying the number of original traces in both scenarios.

Algorithm 4 DDLA

Require: plaintexts p_i
  $k \leftarrow 0$
  while $k < 256$ **do**
    $X_k \leftarrow p_i \in$ plaintexts
    $Y_k \leftarrow L(\text{S-box}[p_i \oplus k])$
    train NN with X_k and Y_k
    $k \leftarrow k + 1$
  end while
  return k for NN with best metrics

Table 4.1: Hyperparameter search space for DDLA-MLP experiments.

Hyperparameter	Min	Max	Step
Learning Rate	0.00005	0.005	0.00025
Dense Layers	1	4	1
Mini-batch	50, 100, 200, 300, 400, 500, 1000 or 2000		
Neurons	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200 or 300		
Activation	ReLU, ELU, or SELU		
Optimizer	Adam or RMSprop		
Kernel Initializer	random_uniform, he_uniform, glorot_uniform, random_normal, he_normal, or glorot_normal		

Table 4.2: Hyperparameter search space for DDLA-CNN experiments.

Hyperparameter	Min	Max	Step
Learning Rate	0.00005	0.005	0.00025
Dense Layers	1	2	1
Convolution Layers	1	4	1
Mini-batch	50, 100, 200, 300, 400, 500, 1000 or 2000		
Neurons	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200 or 300		
Filters	4, 8, 12, 16 or 32		
Kernel Size	5, 10, 20, 30 or 40		
Pool Type	Average or Max		
Activation	ReLU, ELU, or SELU		
Optimizer	Adam or RMSprop		
Kernel Initializer	random_uniform, he_uniform, glorot_uniform, random_normal, he_normal, or glorot_normal		

Table 4.3: Hyperparameter search space for MOR-MLP experiments.

Hyperparameter	Min	Max	Step
Learning Rate	0.00005	0.005	0.00025
Dense Layers	1	4	1
Mini-batch	50, 100, 200, 300, 400, 500, 1000 or 2000		
Neurons	200	1000	100
Activation	ReLU, ELU, or SELU		
Optimizer	Adam or RMSprop		
Kernel Initializer	random_uniform, he_uniform, glorot_uniform, random_normal, he_normal, or glorot_normal		

4.2.2 MOR

The MOR approach trains one neural network with 256 outputs, each corresponding to one key candidate. For each key candidate, we computed the intermediate values for each plaintext,

$$L = L_i(\text{S-box}[p[j] \oplus i]), \forall i \in [0, 256],$$

where L_i represents the function that models the leakage. In the case of MOR, both Identity and Hamming weight were used as leakage models. We use the plaintext p as input to a neural network, and the corresponding 256-valued list L is considered the label. The branch that shows the smallest MSE loss during training corresponds to the correct key candidate. An overview of the algorithm can be found in Algorithm 5.

In each MOR experiment, we run 2 500 models randomly chosen from the hyperparameter search spaces defined in Table 4.3 and Table 4.4 and compute the average success rate and guessing entropy varying the number of traces and epochs. We also show how ensembles perform together with this approach, as well as data augmentation. The original MOR-MLP and MOR-CNN architectures used in [17] can be found in Figure A.2 and Figure A.4, respectively. We refer to them in this work as the best MOR-MLP and MOR-CNN models.

Algorithm 5 MOR

Require: plaintexts p_i
Require: traces $x_i \in X$
  $k \leftarrow 0$
  $Y \leftarrow []$
  while $k < 256$ **do**
    $Y_k \leftarrow L(\text{S-box}[p_i \oplus k])$
    $Y \leftarrow Y_k$
    $k \leftarrow k + 1$
  end while train NN with X as input as Y as output
  return k for NN output branch with lowest MSE loss

Table 4.4: Hyperparameter search space for MOR-CNN experiments.

Hyperparameter	Min	Max	Step
Learning Rate	0.00005	0.005	0.00025
Dense Layers	1	2	1
Convolution Layers	1	4	1
Mini-batch	50, 100, 200, 300, 400, 500, 1000 or 2000		
Neurons	200	1000	100
Filters	4, 8, 12, 16 or 32		
Kernel Size	5, 10, 20, 30 or 40		
Pool Type	Average or Max		
Activation	ReLU, ELU, or SELU		
Optimizer	Adam or RMSprop		
Kernel Initializer	random_uniform, he_uniform, glorot_uniform, random_normal, he_normal, or glorot_normal		

4.3 Results

We vary the number of traces and evaluate the performance of both DDLA-MLP and MOR-MLP for all datasets. For DDLA-MLP, we plot the average guessing entropy (GE) computed based on accuracies or MSE loss for both training and validation data. For MOR-MLP, we plot the average success rate (SR) calculated based on the MSE loss for training and validation data. For both techniques, the number of epochs was fixed at 40.

Furthermore, we vary the number of epochs and evaluate the performance of both DDLA-MLP and MOR-MLP for all datasets. For DDLA-MLP, we plot the average success rate computed based on accuracies or MSE loss for both training and validation data. Similarly, for MOR-MLP, we plot the average success rate calculated based on the MSE loss for training and validation data.

We further evaluate the performance of DDLA-MLP and MOR-MLP when used together with data augmentation on all datasets. We compare two DA techniques, shifting and Gaussian noise, in terms of SR. We augment the number of traces by 10 000 if the initial number was larger than this amount, or we double the initial amount in the other case.

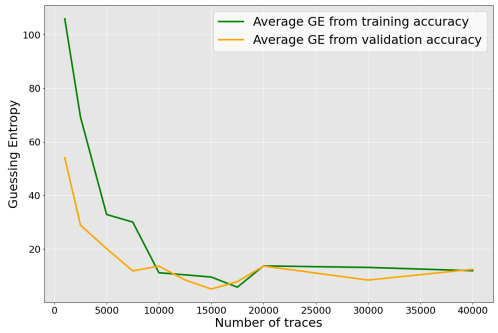
Finally, we evaluate the performance of DL ensembles on all datasets. We consider three ensemble sizes: 20 models, 50 models, and 100 models.

4.3.1 ASCADf

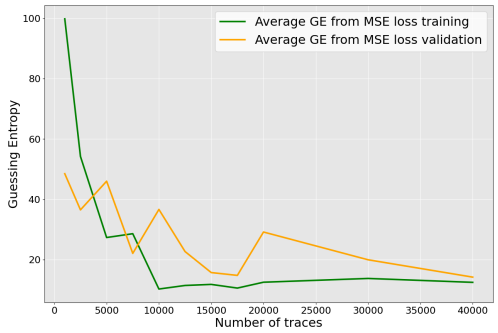
Number of traces

Figure 4.1 shows the average guessing entropy of DDLA-MLP on ASCADf when we vary the number of traces from 1 000 to 40 000. The experiments conducted in Figure 4.1a distinguish the correct key from all the other key candidates by considering the accuracies of 256 the neural networks during both training and validation phases (the neural network that shows the best accuracy corresponds to the correct key). Overall, having a validation set and considering its accuracy performs better than the training set.

Figure 4.1b shows the same experiment, but we distinguish based on the MSE loss instead of accuracy. Here we see that for ASCADf, the training loss shows better convergence to the minimum GE than the validation MSE loss. This is associated with overfitting: the method learned how to perform on the training data very well, resulting in poorer performance on new data.



(a) `ASCADf`, GE varying the number of traces considering the accuracy as the distinguisher.



(b) `ASCADf`, GE varying the number of traces considering the loss as the distinguisher.

Figure 4.1: Guessing entropy (GE) for DDLA-MLP on the ASCADf dataset for the LSB leakage model.

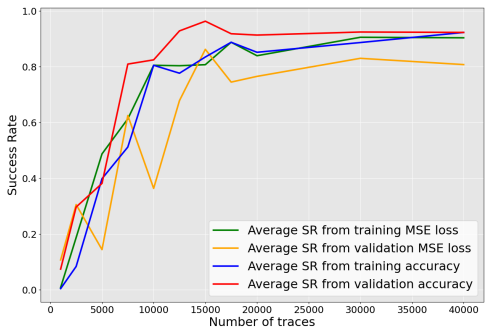


Figure 4.2: Success rate (SR) of the DDLA methodology on the ASCADf dataset varying the number of traces. We consider four different distinguishers: accuracy during training, accuracy during validation, MSE training loss, and MSE validation loss.

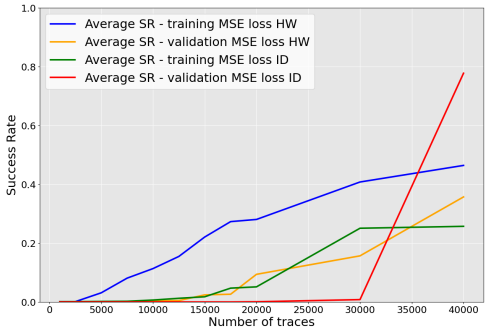


Figure 4.3: Success rate (SR) of the MOR methodology on the ASCADf dataset varying the number of traces. Both Identity and Hamming weight leakage models are considered. The key distinguisher is the MSE loss on training or validation data.

For ASCADf, we obtain similar results whether we use the accuracy or the MSE loss as performance measures when distinguishing the neural network that performs the best: 10 000 are enough to break the secret key.

Figure 4.2 shows the success rate of the DDLA approach on ASCADf varying the number of traces. For this dataset, the validation accuracy brings the best performance when used as the distinguisher.

Figure 4.3 shows the influence of measurements on the performance of the MOR approach. As in the case of DDLA, for ASCADf, the MOR methodology performs better when considering the training MSE loss rather than validation loss. We believe that is the case since both techniques were developed for ASCADf, and other datasets were not thoroughly tested. Furthermore, we see that the success rate of the MOR technique does not exceed 40% on the hyperparameter space from Table 4.3. Compared to DDLA, MOR is more sensitive to the network's parameters.

Number of epochs

Figure 4.4 shows the influence of the number of epochs on the average guessing entropy of DDLA-MLP for ASCADf when the number of input traces is fixed at 40 000. We see that after 10 epochs, the GE is not further improved for either of the distinguishers. Thus, choosing a larger epoch number is not needed for this dataset.

Figure 4.5 shows the guessing entropy of the best MOR model introduced in [17] on ASCADf when

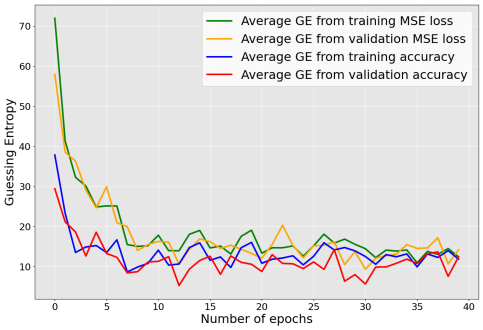


Figure 4.4: Guessing entropy (GE) of the DDLA methodology on the ASCADf dataset varying the number of epochs. We fixed the number of traces at 40 000. We considered four different distinguishers: accuracy during training, accuracy during validation, MSE training loss, and MSE validation loss.

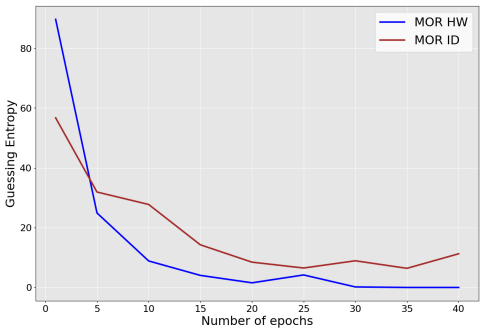
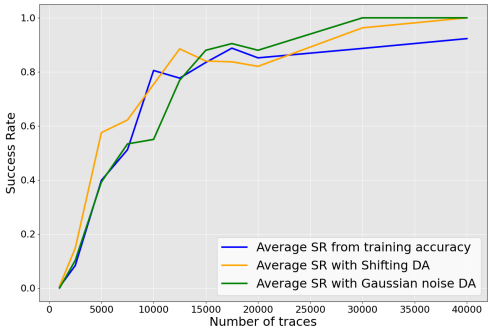
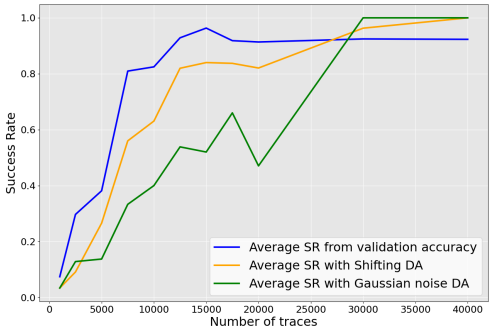


Figure 4.5: Guessing entropy (GE) of the best MOR model on the ASCADf dataset varying the number of epochs. We fixed the number of traces at 20 000. We chose the MSE training loss as the distinguisher.



(a) `ASCADf`, SR varying the number of traces considering the training accuracy as the distinguisher.



(b) `ASCADf`, SR varying the number of traces considering the validation accuracy as the distinguisher.

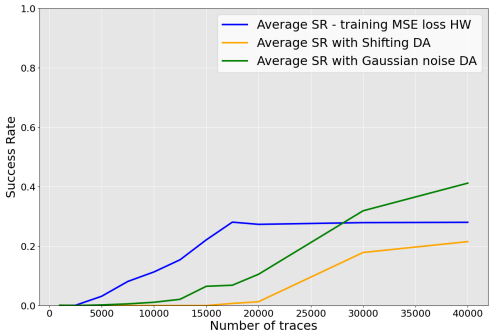
Figure 4.6: Success rate (SR) for DDLA-MLP on the ASCADf dataset for the LSB leakage model. Two different data augmentation techniques were used: shifting and Gaussian noise.

We fixed the number of traces at 20 000. We run the same MLP architecture on randomly chosen data from the ASCADf database and averaged the GE. The Hamming weight leakage model performs better than ID, reaching GE 1 after 30 epochs.

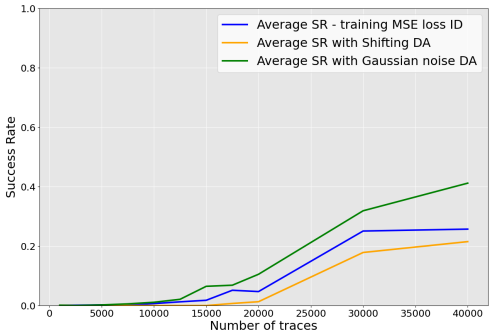
Data augmentation

Figure 4.6 shows the success rate for DDLA-MLP on ASCADf with data augmentation. In Figure 4.6a, we plotted how the average SR of the models calculated based on the training accuracy of the models is altered when using DA. Shifting DA performs better overall when the amount of original traces is smaller than 10 000. However, with more original traces, Gaussian noise DA is preferred. On the other hand, the SR calculated based on the validation accuracy distinguisher is affected when using DA, as we see in Figure 4.6b.

For MOR, the leakage model used highly influences the performance of the attack. Figure 4.7 shows the average SR of MOR-DDLA on ASCADf with and without DA for both leakage models: HW and ID. Figure 4.7a shows that MOR with DA does not improve overall when we use the HW leakage model. However, in Figure 4.7b, we see that Gaussian noise DA improves the success rate of MOR-MLP with ID leakage.



(a) ASCADf, SR varying the number of traces considering the training MSE loss as the distinguisher and HW leakage.



(b) ASCADf, SR varying the number of traces considering the training MSE loss as the distinguisher and ID leakage.

Figure 4.7: Success rate (SR) for MOR-MLP on the ASCADf dataset for both Hamming weight and Identity leakage models. Two different data augmentation techniques were used: shifting and Gaussian noise.

Ensembles

As illustrated in Figure 4.3, when the hyperparameters are selected from a large search space, as defined in Table 4.3, the success rate of MOR is significantly low. Ensembles are thus unsuitable for such a case: if one in 1 000 models succeeds in finding the correct key, averaging their results does not improve the success rate. However, we observed that when we reduce the search space, the SR grows, and ensembles can improve the overall performance of MOR.

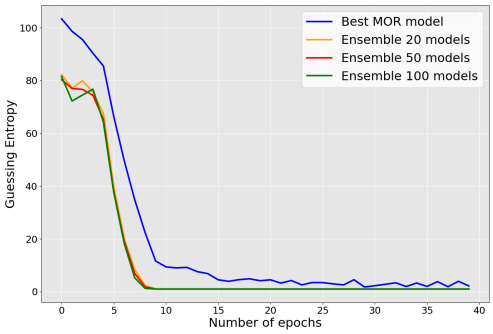
We thus narrowed down the hyperparameter space introduced in Table 4.3 to a smaller space as we see in Table 4.5. We further investigate the performance of ensembles on ASCADf compared to the best MOR model described in [17]. We present the results in Figure 4.8.

Figure 4.8a shows that ensembles in the case of HW leakage model and training MSE loss as the distinguisher reach minimum GE faster than the best MOR model: 10 epochs are enough to break ASCADf. Moreover, Figure 4.8b shows that there is improvement also in the case of the ID leakage model.

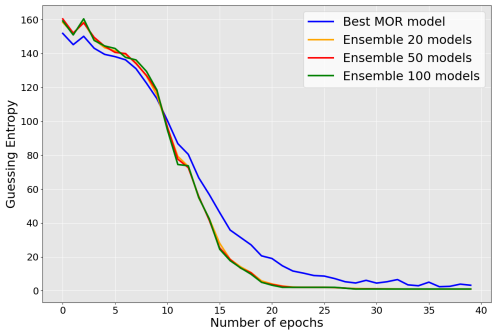
Interestingly, when we use the ID leakage model with the MSE validation loss as the distinguisher, very few epochs are enough to break ASCADf. However, the best MOR model overfits after 20 epochs, as we see in Figure 4.8c. However, ensembles avoid this problem: the GE is maintained to the minimum even after 20 epochs.

Table 4.5: Hyperparameter search space for MOR-MLP used in ensembles.

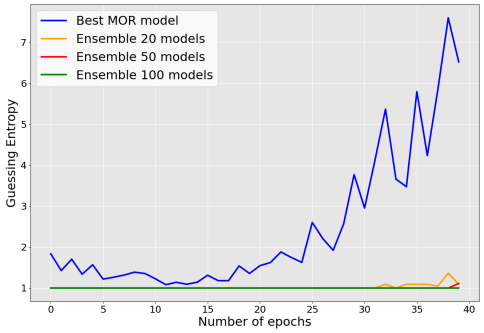
Hyperparameter	Min	Max	Step
Learning Rate	0.00005	0.005	0.00025
Mini-batch	50	200	50
Dense Layers	1	4	1
Neurons	200	1 000	100
Activation	ReLU, ELU, or SELU		



(a) ASCADf, GE varying the number of training epochs considering the training MSE loss as the distinguisher and HW leakage.

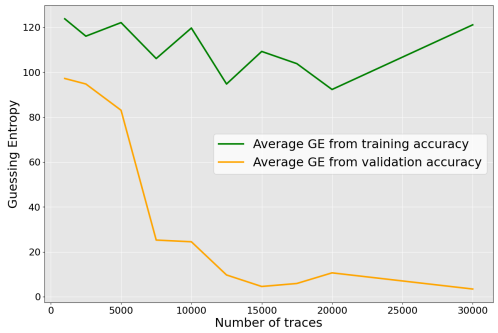


(b) ASCADf, GE varying the number of training epochs considering the training MSE loss as the distinguisher and ID leakage.

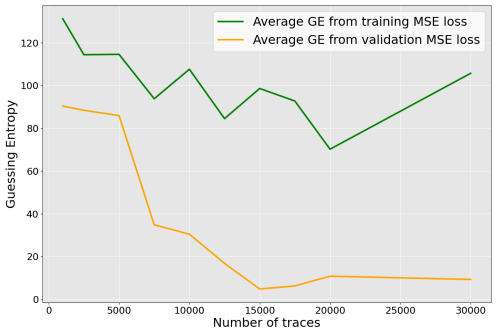


(c) ASCADf, GE varying the number of training epochs considering the validation MSE loss as the distinguisher and ID leakage.

Figure 4.8: Guessing Entropy (GE) for MOR-MLP on the ASCADf dataset for Hamming weight (HW) and Identity (ID) leakage models. We considered both MSE training and validation losses as the distinguishers for ID. For HW, only training loss is considered since there was no improvement for validation MSE loss. The number of traces is 20 000. We employed three different ensemble sizes: 20, 50, or 100 models.



(a) AES_RD, GE varying the number of traces considering the accuracy as the distinguisher.



(b) AES_RD, GE varying the number of traces considering the loss as the distinguisher.

Figure 4.9: Guessing entropy (GE) for DDLA-MLP on the AES_RD dataset for the LSB leakage model.

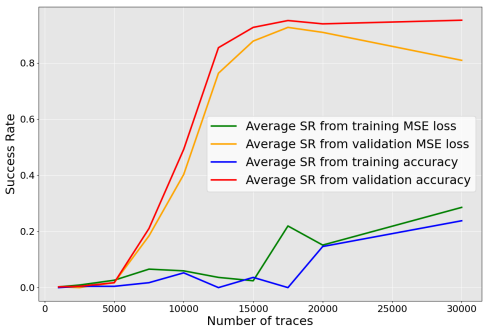


Figure 4.10: Success rate (SR) of the DDLA methodology on the AES_RD dataset varying the number of traces. We considered four different distinguishers: accuracy during training, accuracy during validation, MSE training loss, and MSE validation loss.

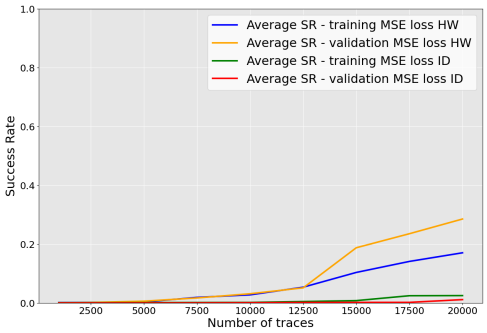


Figure 4.11: Success rate (SR) of the MOR methodology on the AES_RD dataset varying the number of traces. Both Identity and Hamming weight leakage models are considered. The key distinguisher is the MSE loss on training or validation data.

4.3.2 AES_RD

Number of traces

The graph depicted in Figure 4.9a displays the average guessing entropy we obtained by DDLA-MLP when the accuracy is employed as a distinguisher varying the number of input traces. Unlike ASCADf, which can be broken by considering the training accuracy, AES_RD is not vulnerable to such an attack due to overfitting. Therefore, we should only rely on the validation accuracy to evaluate the model's performance. Similarly, Figure 4.9b illustrates that the MSE loss during training cannot serve as the distinguisher for AES_RD, and only the validation MSE loss should be considered. It is worth noting that AES_RD can be compromised with only 15,000 traces.

Figure 4.10 shows the success rate of the DDLA approach varying the number of traces. The validation accuracy performs slightly better than the validation MSE loss as the distinguisher. As explained before, the training accuracy or MSE loss should not be used for AES_RD: the success rate does not reach 40% within 30 000 traces.

Figure 4.11 shows the success rate of the MOR methodology, considering two distinguishers: MSE training and validation loss. We plotted the SR for both leakage models: HW and ID. MOR cannot break AES_RD with ID within 20 000 traces. This is due to overfitting. On the other hand, for HW, this approach shows better performance. Further, the MSE validation loss reaches a higher SR than the MSE training loss after 12 500 input traces.

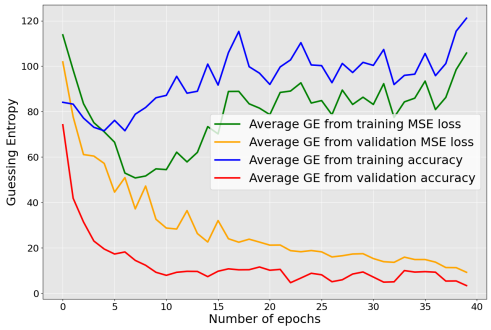


Figure 4.12: Guessing entropy (GE) of the DDLA methodology on the AES_RD dataset varying the number of epochs. We fixed the number of traces at 20 000. We considered four different distinguishers: accuracy during training, accuracy during validation, MSE training loss, and MSE validation loss.

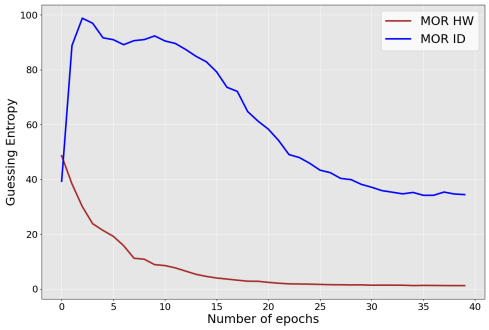


Figure 4.13: Guessing entropy (GE) of the best MOR model on the AES_RD dataset varying the number of epochs. We fixed the number of traces at 20 000. The MSE training loss was considered the distinguisher in the case of ID leakage, and the MSE validation loss in the case of HW leakage.

Number of epochs

Figure 4.12 shows the effect of training epochs against DDLA on AES_RD when the number of input traces is fixed at 20 000. We see that the GE trend does not change after 15 epochs. Thus, DDLA can easily break AES_RD with only 15 epochs required.

We further varied the number of epochs when the number of traces is 20 000 for the MOR approach. Figure 4.13 shows the average GE of the best MOR model introduced in [17] for AES_RD for both leakage models: Identity and Hamming weight. For the Identity leakage model, we choose the MSE training loss as the distinguisher since it performs better. Similarly, for Hamming weight leakage, we choose the validation MSE loss. We see that for Identity, the attack does not reach GE 1 even when the number of epochs is 40, resulting in an unsuccessful attack due to overfitting. However, 25 epochs are enough for HW to result in a successful key recovery.

Data augmentation

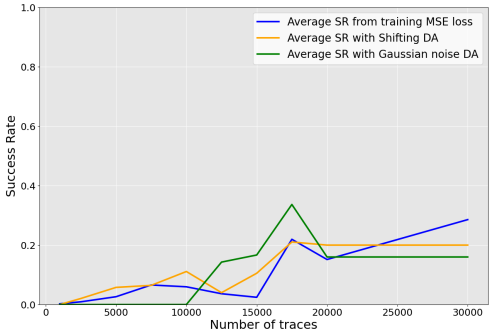
Figure 4.14 shows the success rate of DDLA on AES_RD when data augmentation is used. Figure 4.14a shows both Gaussian noise and shifting are useful against AES_RD, improving the SR when we employ the training MSE loss as the distinguisher. However, we obtain a higher success rate on this dataset when we use the validation MSE loss as the distinguisher. Figure 4.14b shows that DA with Gaussian noise is overall helping improve the SR while shifting drastically reduces it.

In the case of the MOR methodology, we shot the results of data augmentation in Figure 4.15. There is a slight improvement when the number of traces is smaller than 10 000 for both HW and ID leakage models. However, no significant changes in the SR were observed for AES_RD when using MOR together with DA.

Ensembles

As in the case of ASCADf, the MOR approach on AES_RD does not show great performance: the success rate is low when using a large hyperparameter space. We investigate the performance of ensembles with a reduced search space (see Table 4.5).

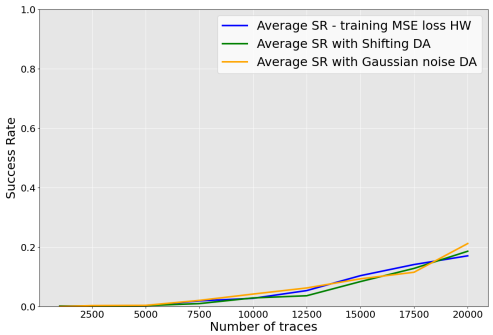
Figure 4.16 shows the average GE of the ensembles compared to the best MOR model. Figure 4.16a shows how ensembles avoid overfitting in the case of the HW leakage model and training MSE loss. As we see in Figure 4.15a, the SR when HW and training MSE loss is significantly low. This is again



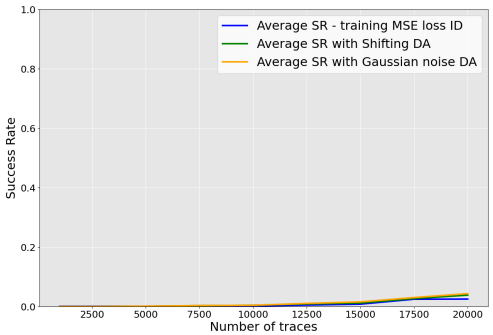
(a) AES_RD, SR varying the number of traces considering the training accuracy as the distinguisher.

(b) AES_RD, SR varying the number of traces considering the validation accuracy as the distinguisher.

Figure 4.14: Success rate (SR) for DDLA-MLP on the AES_RD dataset for the LSB leakage model. Two different data augmentation techniques were used: shifting and Gaussian noise.

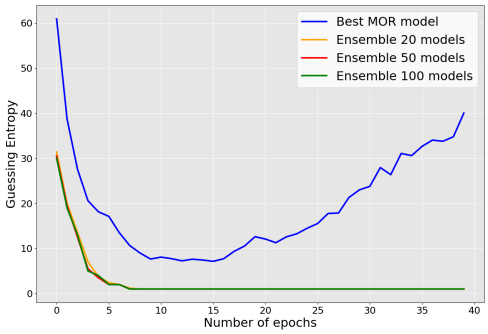


(a) AES_RD, SR varying the number of traces considering the training MSE loss as the distinguisher and HW leakage.

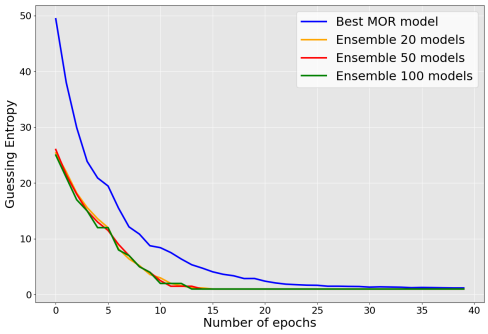


(b) AES_RD, SR varying the number of traces considering the training MSE loss as the distinguisher and ID leakage.

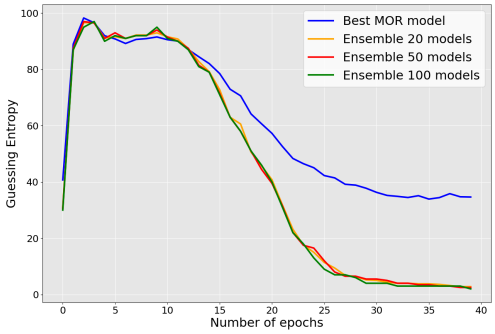
Figure 4.15: Success rate (SR) for MOR-MLP on the AES_RD dataset for both Hamming weight (HW) and Identity (ID) leakage models. Two different data augmentation techniques were used: shifting and Gaussian noise.



(a) AES_RD, GE varying the number of training epochs considering the training MSE loss as the distinguisher and HW leakage.



(b) AES_RD, GE varying the number of training epochs considering the validation MSE loss as the distinguisher and HW leakage.



(c) AES_RD, GE varying the number of training epochs considering the training MSE loss as the distinguisher and ID leakage.

Figure 4.16: Guessing Entropy (GE) for MOR-MLP on the AES_RD dataset for Hamming weight (HW) and Identity (ID) leakage models. We considered both MSE training and validation losses as distinguishers for HW. For ID, only training loss is considered since there was no improvement in validation MSE loss. The number of traces is 20 000. We employed three different ensemble sizes: 20, 50, or 100 models.

confirmed here: after 15 epochs, there is overfitting. However, ensembles avoid this problem, and the correct key can be recovered within 10 epochs.

Moreover, Figure 4.16c shows that ensembles can greatly improve the performance of MOR on AES_RD with the ID leakage model.

4.3.3 AES_HD

Number of traces

Figure 4.17 shows the influence of the number of input traces on the average guessing entropy of DDLA-MLP methodology considering the accuracy and MSE loss as the distinguishers, either on training or validation data. Regardless of the distinguisher, AES_HD cannot be broken with DDLA.

Furthermore, Figure 4.18 shows the success rate of DDLA on AES_HD. We see here again, that the success rate is never larger than 5% for any distinguisher.

In contrast, MOR achieves a slightly better performance on AES_HD. Figure 4.19 shows the influence of the amount of input traces on the average SR of the MOR-MLP approach. When choosing the HW leakage model, the input size does not influence the SR: the attack remains unsuccessful. For ID, more than 15 000 traces are necessary and even then the SR is around 20%.

(a) AES_HD, GE varying the number of traces considering the accuracy as the distinguisher.

(b) AES_HD, GE varying the number of traces considering the loss as the distinguisher.

Figure 4.17: Guessing entropy (GE) for DDLA-MLP on the AES_HD dataset for the LSB leakage model.

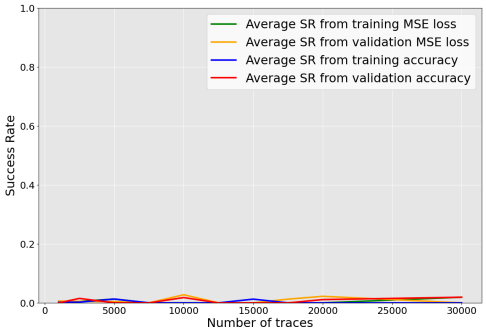


Figure 4.18: Success rate (SR) of the DDLA methodology on the AES_HD dataset varying the number of traces. We considered four different distinguishers: accuracy during training, accuracy during validation, MSE training loss, and MSE validation loss.

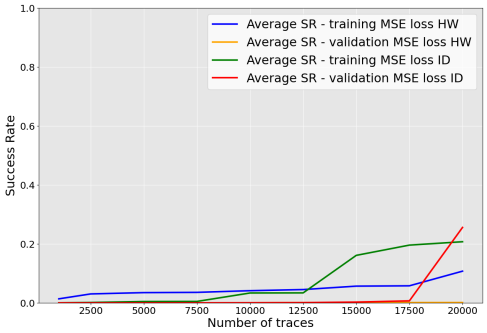


Figure 4.19: Success rate (SR) of the MOR methodology on the AES_HD dataset varying the number of traces. Both Identity and Hamming weight leakage models are considered. The key distinguisher is the MSE loss on training or validation data.

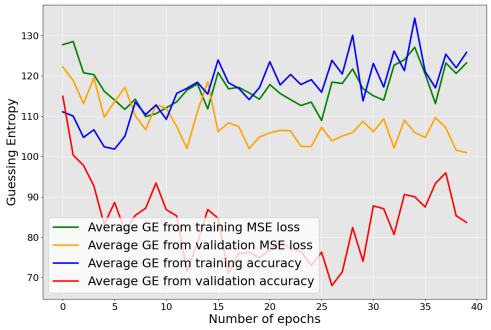


Figure 4.20: The GE (Guessing entropy) of the DDLA methodology on the AES_HD dataset varying the number of epochs. We fixed the number of traces at 30 000. We considered four different distinguishers: accuracy during training, accuracy during validation, MSE training loss, and MSE validation loss.

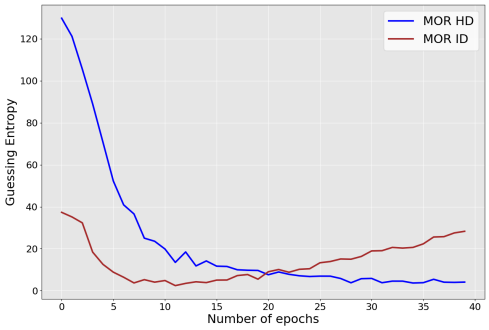


Figure 4.21: Guessing entropy (GE) of the MOR methodology on the AES_HD dataset varying the number of epochs for Identity (ID) and Hamming distance (HD) leakage models. We fixed the number of traces at 20 000. The MSE training loss was considered the distinguisher.

Number of epochs

We varied the number of training epochs and plotted its influence on the GE of the attack. We fixed the number of input traces at 30 000. We show the results in Figure 4.20. Our findings indicate that after 5 epochs, the GE stabilizes and additional epochs do not enhance its performance.

Figure 4.21 shows the influence of the epoch number on the GE of the best MOR model introduced in [17] for 20 000 input traces. We see that for the Hamming distance (HD) leakage model the attack is successful after only 10 epochs. For ID, the MOR model requires more epochs than in the case of HD. However, the model shows overfitting after 20 epochs.

Data augmentation

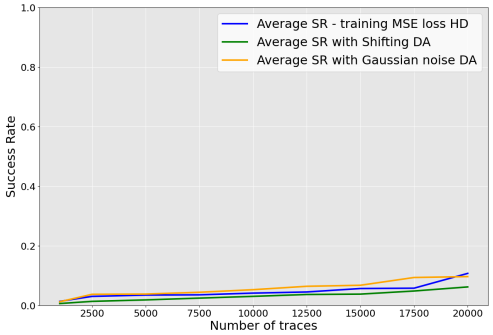
The experiments involving data augmentation and DDLA were not added to our analysis of the AES_HD dataset. The reason for this was that the attack on this dataset was unsuccessful from the outset, regardless of the number of traces used. Therefore, it was expected that data augmentation would not lead to an enhancement in the success rate of the attack.

In the case of MOR, we show the results of using data augmentation in Figure 4.22. For both leakage models, Gaussian noise DA helps by slightly improving the success rate of MOR on AES_HD, while shifting DA reduces the overall SR.

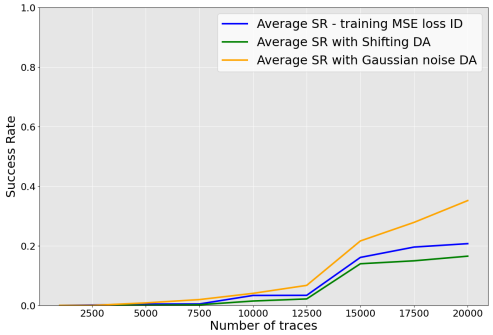
Ensembles

We further investigate the performance of ensembles against the best MOR model on AES_HD. The models used in ensembles are randomly drawn from Table 4.5.

Figure 4.23 shows that ensembles improve the performance (here, GE) of MOR compared to the best model. That is, in the case of the HD leakage model, the GE reaches 0 after 10 epochs. Moreover, in the case of ID, overfitting is avoided, and 5 epochs are enough to break the secret key, as we see in Figure 4.23b.

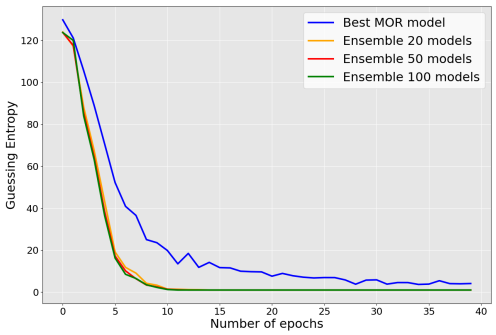


(a) AES_HD, SR varying the number of traces considering the training MSE loss as the distinguisher and HD leakage.

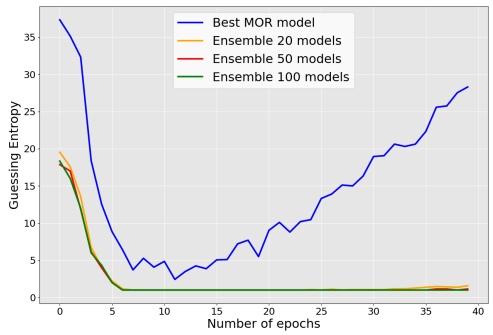


(b) AES_HD, SR varying the number of traces considering the training MSE loss as the distinguisher and ID leakage.

Figure 4.22: Success rate (SR) for MOR-MLP on the AES_HD dataset for both Hamming distance (HD) and Identity (ID) leakage models. Two different data augmentation techniques were used: shifting and Gaussian noise.

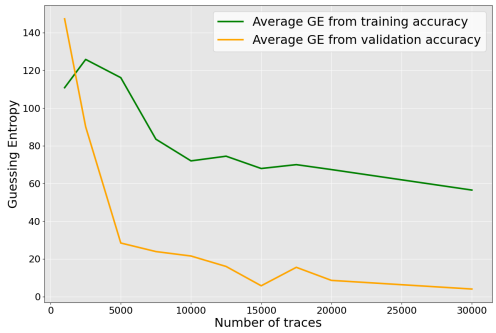


(a) AES_HD, GE varying the number of training epochs considering the training MSE loss as the distinguisher and HD leakage.

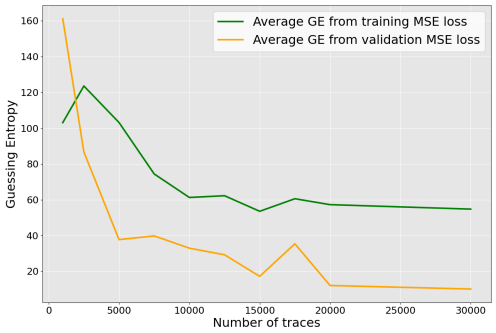


(b) AES_HD, GE varying the number of training epochs considering the training MSE loss as the distinguisher and ID leakage.

Figure 4.23: Guessing Entropy (GE) for MOR-MLP on the AES_HD dataset for Hamming distance (HD) and Identity (ID) leakage models. We use the MSE training loss as the distinguisher. The number of traces is 20 000. We employed three different ensemble sizes: 20, 50, or 100 models.



(a) ASCADr, GE varying the number of traces considering the accuracy as the distinguisher.



(b) ASCADr, GE varying the number of traces considering the loss as the distinguisher.

Figure 4.24: Guessing entropy for DDLA-MLP on the ASCADr dataset for the LSB leakage model.

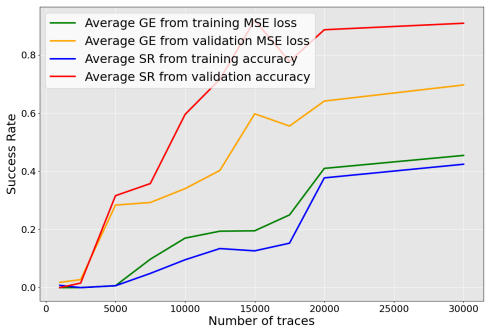


Figure 4.25: Success rate (SR) of the DDLA methodology on the ASCADr dataset varying the number of traces. We considered four different distinguishers: accuracy during training, accuracy during validation, MSE training loss, and MSE validation loss.

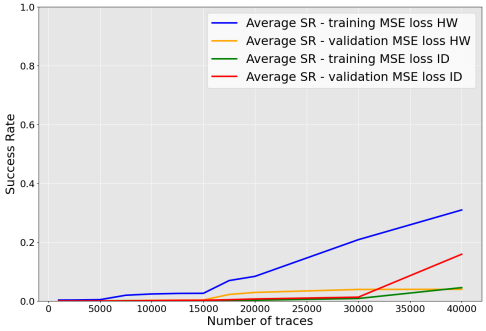


Figure 4.26: Success rate (SR) of the MOR methodology on the ASCADr dataset varying the number of traces. Both Identity and Hamming weight leakage models are considered. The key distinguisher is the MSE loss on training or validation data.

4.3.4 ASCADr

Number of traces

Figure 4.24 shows the guessing entropy of the DDLA-MLP approach on ASCADr. Figure 4.24a considers the accuracy during training or validation phases as the distinguishers, while Figure 4.24b the MSE loss. Different from ASCADf, computing the chosen metric (accuracy or loss) on the validation data results in a better attack than considering the training data. Moreover, having the MSE loss as the distinguisher results in a faster attack: the key is recovered with more than 20 000 traces, while for accuracy, 30 000 traces are needed.

Moreover, Figure 4.25 shows the success rate of the DDLA-MLP technique on ASCADr, varying the number of traces. As in the case of ASCADf, the validation accuracy distinguisher results in the most successful attack.

Figure 4.26 shows the success rate of the MOR-MLP approach on ASCADr, varying the number of traces. As expected, the MOR methodology shows again a high sensitivity to the hyperparameters of the networks and does not reach a success rate as high as in the case of DDLA-MLP when a large search space is considered (Table 4.3).

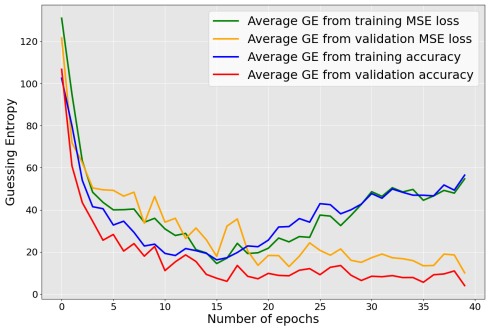


Figure 4.27: Guessing entropy (GE) of the DDLA methodology on the ASCADr dataset varying the number of epochs. We fixed the number of traces at 20,000. We considered four different distinguishers: accuracy during training, accuracy during validation, MSE training loss, and MSE validation loss.

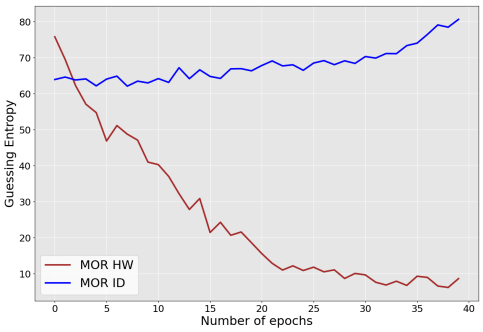


Figure 4.28: Guessing entropy (GE) of the MOR methodology on the ASCADr dataset varying the number of epochs. We fixed the number of traces at 20,000. The MSE training loss was considered the distinguisher.

Number of epochs

Figure 4.27 shows the influence of the number of epochs on the average GE of the DDLA-MLP approach. We see that in the case of both validation, MSE loss and validation accuracy is chosen as the distinguishers; 20 epochs are enough to result in a successful attack. However, considering the same metrics on the training data results in overfitting after 20 epochs: the GE starts increasing.

In Figure 4.28, we plotted the GE of the best MOR model introduced in [17] for the ASCADr dataset. We see that in the case of HW leakage, the model achieves a small GE after 30 epochs. However, overfitting happens for ID, and recovering the secret key is impossible regardless of the number of epochs.

Data augmentation

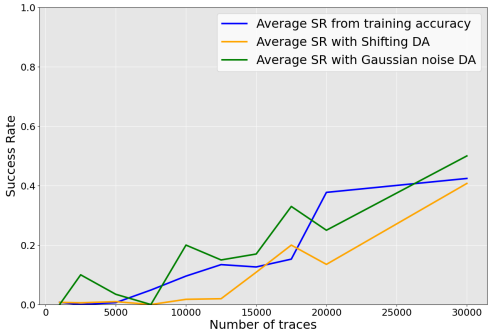
Figure 4.29 shows the average guessing entropy for DDLA-MLP on ASCADr with and without data augmentation. In Figure 4.29a, we see that Gaussian noise DA performs better overall when considering the training accuracy as the distinguisher. This is expected since overfitting can happen when considering a distinguishing metric on training data. Data augmentation indeed helps with this problem. However, when we distinguish the correct key from the key candidates based on validation data, data augmentation drastically reduces the success rate of the attack, as we see in Figure 4.29b.

Figure 4.30 shows the influence of data augmentation on MOR's performance. Figure 4.30a shows that shifting DA helps improve the success rate of MOR when we use the HW leakage model on ASCADr. However, we observe no improvement in the case of the ID leakage model. That is because the success rate of MOR on this dataset is low.

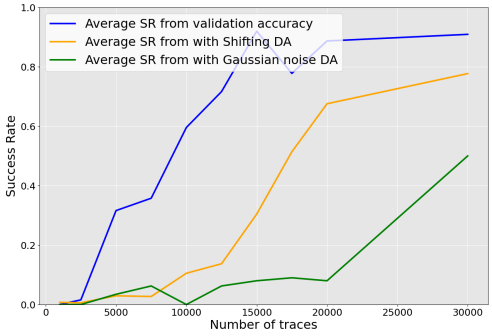
Ensembles

As we see in Figure 4.28, the best MOR model cannot break ASCADr with the ID leakage model. Figure 4.30b further shows the same observation: the SR of MOR on ASCADr is low. Therefore, using MOR ensembles does not show any improvement in the case of the ID leakage model.

However, for HW, ensembles show improvement in both the number of epochs needed to reach GE 1 and overall lower GE. Figure 4.31 shows that ASCADr requires 20 epochs to retrieve the secret key using ensembles. Furthermore, the GE becomes indeed 0 after this number of epochs, while for the best MOR model, the average GE is around 10.

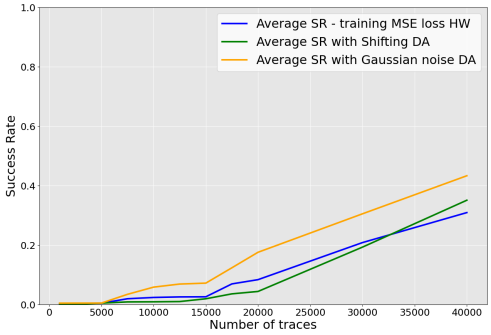


(a) ASCADr, SR varying the number of traces considering the training accuracy as the distinguisher.

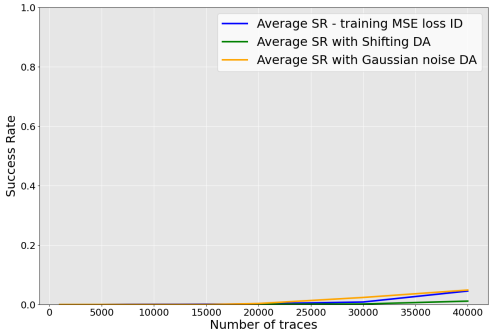


(b) ASCADr, SR varying the number of traces considering the validation accuracy as the distinguisher.

Figure 4.29: Success rate (SR) for DDLA-MLP on the ASCADr dataset for the LSB leakage model. Two different data augmentation techniques were used: shifting and Gaussian noise.



(a) ASCADr, SR varying the number of traces considering the training MSE loss as the distinguisher and HD leakage.



(b) ASCADr, SR varying the number of traces considering the training MSE loss as the distinguisher and ID leakage.

Figure 4.30: Success rate (SR) for MOR-MLP on the ASCADr dataset for both Hamming distance (HW) and Identity (ID) leakage models. Two different data augmentation techniques were used: shifting and Gaussian noise.

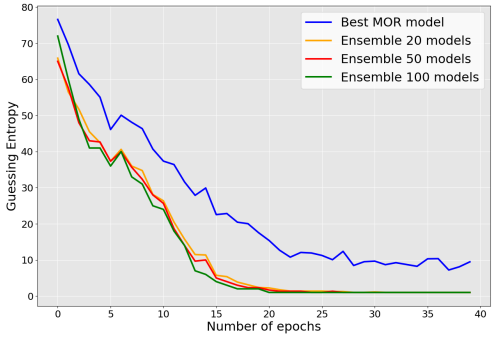


Figure 4.31: ASCADr, Guessing Entropy (GE) varying the number of training epochs considering the training MSE loss as the distinguisher and HW leakage. The number of traces is 20 000. We employed three different ensemble sizes: 20, 50, or 100 models.

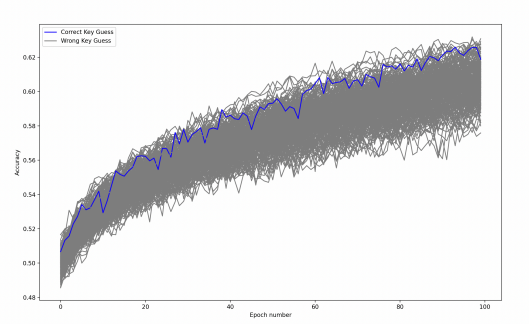


Figure 4.32: DDLA-CNN accuracies for all key candidates on synchronized ASCADf.

4.3.5 Desynchronized ASCADf

As in the case of synchronized ASCADf, we aimed to evaluate DDLA and MOR approaches against desynchronized ASCADf. We randomly choose 50 models from Table 4.2 in the case of DDLA and 250 models from Table 4.4 in the case of MOR for both HW and ID leakage models and varying the number of input traces. However, the success rate for both methodologies is 0, thus we excluded them from this chapter. Instead, we conducted experiments using the best DDLA-CNN [67] and MOR-CNN [17] models.

The CNN architecture used in the original DDLA experiments [67] uses two convolution layers and can break Chip-Whisperer traces with desynchronization 20. One of the first drawbacks of Timon's method is the fact that their attacks did not include experiments against desynchronized ASCADf. We reproduced the CNN model and the results showed that even for *synchronized* ASCADf database, cannot distinguish the correct key candidate, as we see in 4.32.

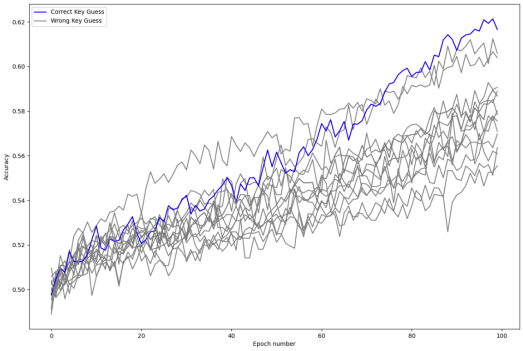
We were able to use this architecture effectively on ASCADf after we added two more dense layers after the convolution layers. More specifically, we added one dense layer of size 20 and one dense layer of size 10, both having ReLU as the activation function. This model can break 15-desynchronized ASCADf. However, Fig. 4.33a shows that for 30-desynchronized traces, the correct key is not clearly distinguished from all other possible key candidates.

We further added one extra convolution layer and noticed that it could better differentiate between the key candidates, as we see in Fig. 4.33b. As expected, increasing the maximum desynchronization level impacts the performance of the CNN model. That is, the number of convolution layers must be increased when working with higher desynchronized traces. Fig. 4.33d shows that the CNN architecture with 5 convolution layers can also break the 50-desynchronized ASCADf dataset.

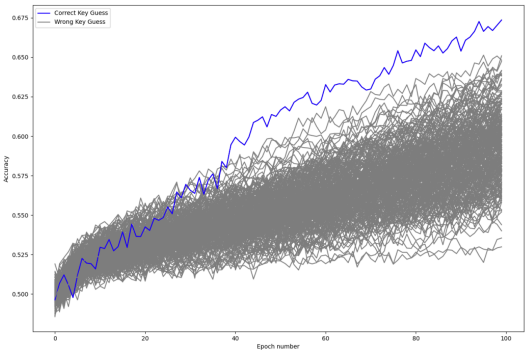
The MOR-CNN best model has two convolution layers (see Figure A.4). As in the case of DDLA, the experiments were conducted on Chip-Whisperer traces with a maximum desynchronization level of 20. We tried to reproduce the results on desynchronized ASCADf, but there were no successful results: the correct key could not be distinguished. However, we propose another CNN architecture that obtained great results with different desynchronization levels. We present this model in Figure A.5.

Figure 4.34 shows the training MSE loss for all key candidates using different amounts of input traces. Our MOR-CNN model can break 50-desynchronized ASCADf even though 10 000 input traces are used, as we see in Figure 4.34a.

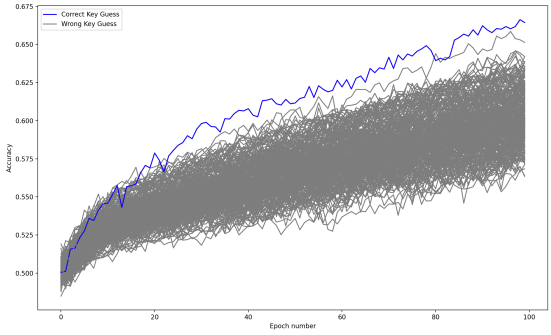
We further varied the maximum desynchronization level and plotted the results in Figure 4.35. The proposed model can easily break 200-desynchronized traces even for 20 training epochs. Surprisingly, the correct key was retrieved faster in the case of 150-desynchronized ASCADf rather than for 100-desynchronized ASCADf. We believe this is due to overfitting for 100-desynchronized ASCADf: the MOR-CNN model is not small; there are 5 convolution layers.



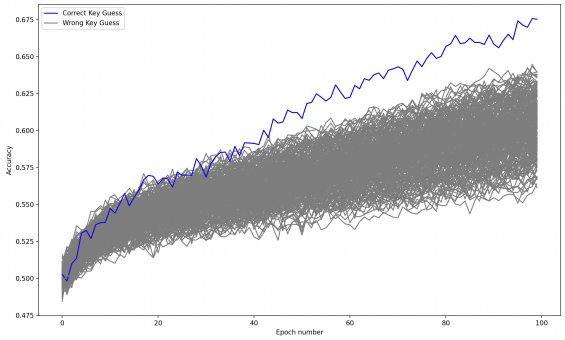
(a) DDLA-CNN accuracies for some keys using 7 500 30-desynchronized traces.



(b) Altered DDLA-CNN accuracies for all keys using 7 500 30-desynchronized traces.

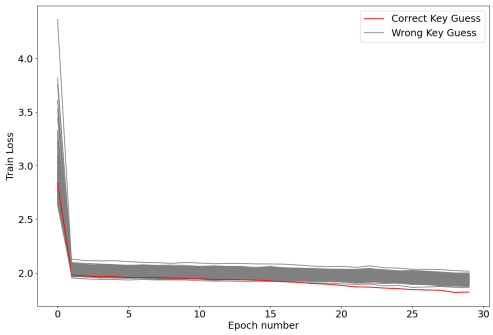


(c) Altered DDLA-CNN accuracies for all keys using 7 500 40-desynchronized traces.

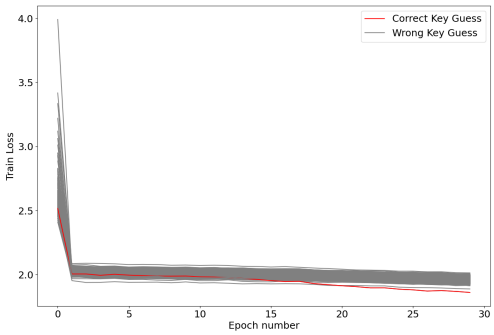


(d) Altered DDLA-CNN accuracies for all keys using 7 500 50-desynchronized traces.

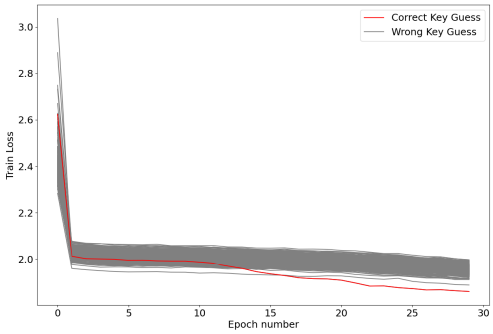
Figure 4.33: DDLA-CNN vs altered DDLA-CNN accuracies varying the maximum desynchronization value.



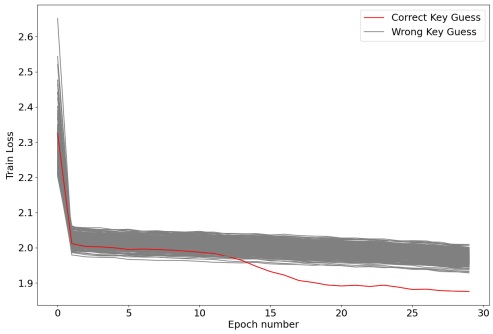
(a) Proposed MOR-CNN MSE loss for all key candidates using 10 000 traces.



(b) Proposed MOR-CNN MSE loss for all key candidates using 15 000 traces.

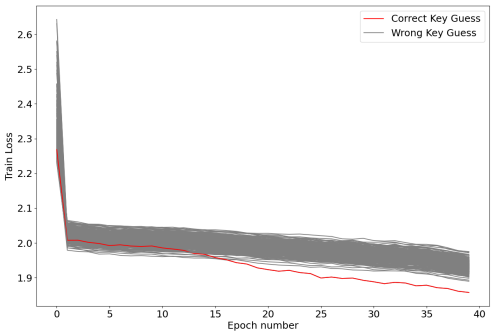


(c) Proposed MOR-CNN MSE loss for all key candidates using 20 000 traces.

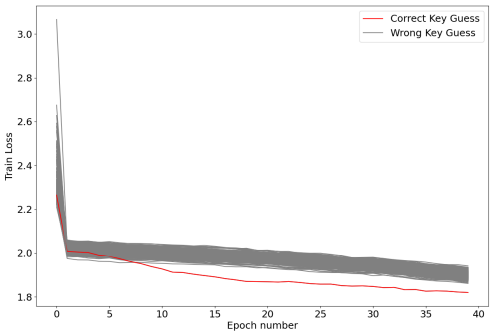


(d) Proposed MOR-CNN MSE loss for all key candidates using 30 000 traces.

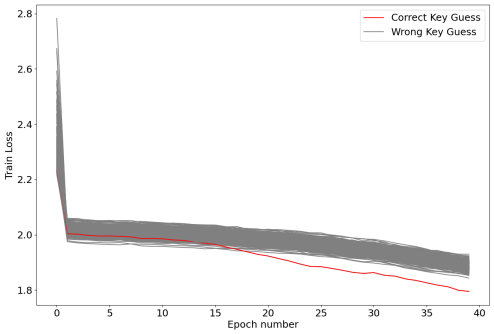
Figure 4.34: The MSE training loss values for the proposed MOR-CNN model varying the number of traces. The dataset used was 50-desynchronized ASCADf.



(a) `100-desynchronized ASCADf`, proposed MOR-CNN MSE loss for all key candidates.



(b) `150-desynchronized ASCADf`, proposed MOR-CNN MSE loss for all key candidates.



(c) `200-desynchronized ASCADf`, proposed MOR-CNN MSE loss for all key candidates.

Figure 4.35: The loss function values for the proposed MOR-CNN model varying the desynchronization level. The number of traces is fixed at 30 000.

4.4 Discussion

In this chapter, we provided experimental results of two deep-learning-based unsupervised side-channel analysis approaches: differential deep learning analysis (DDLA) [67] and multi-output regression (MOR) [18, 17]. For both approaches, we randomly chose models from large hyperparameter spaces; see Table 4.1.

DDLA shows lower sensitivity to the MLP hyperparameters when compared to MOR: the average success rate of the DDLA attacks was close to 100% for ASCADf, AES_RD, and ASCADr. However, DDLA was not able to break AES_HD due to overfitting. On the other hand, the MOR experiments show how much the performance is affected when the models are randomly chosen from such a large space: the success rate is not larger than 30% for any of the datasets.

Furthermore, our experiments show that choosing the accuracy or MSE loss of the validation data improves the performance of DDLA. The original work employed the accuracy/loss during the training phase [67]. This approach results in an unsuccessful attack for an easy dataset to break, such as AES_RD; however, when distinguishing on training data, we saw that the success rate could reach 100%. This is due to overfitting that can easily happen, especially for easy datasets to break where the neural networks can easily fit the labels, even for wrong key candidates.

Moreover, we conducted experiments on desynchronized ASCADf traces for both DDLA and MOR approaches. In the case of DDLA, we could not break ASCADf with the proposed model introduced in [67]. However, we propose another architecture that could break 50-desynchronized ASCADf within 50 epochs for 7 500 traces. Similarly, we could not reproduce the results of MOR with the CNN architecture introduced in [17]. We proposed another model that showed good performance even on 200-desynchronized ASCADf.

The MOR approach showed greater sensitivity to hyperparameters for CNNs than for MLPs. The success rate on desynchronized traces was 0%, and experiments were therefore omitted. We noticed that the networks' size greatly influences the SR for CNNs: too small networks result in unsuccessful attacks, while significantly large networks result in overfitting for all key candidates.

We used two data augmentation techniques: adding shifting and Gaussian noise to the original traces to create new traces. We noticed that in both DDLA and MOR, in most cases, when we use the metric distinguisher on training data, using Gaussian noise data augmentation shows better performance. Overfitting can easily happen for training data, resulting in high training accuracy or low training loss. As expected, data augmentation helps to avoid this overfitting.

Moreover, we noticed that shifting data augmentation reduces, in most cases, the success rate. We hypothesize that shifting, which resembles the desynchronization countermeasure, introduces greater dissimilarity between the augmented shifted traces and the original traces compared to applying Gaussian noise. Consequently, a multi-layer perceptron struggles to effectively generalize from the augmented shifted traces due to this increased dissimilarity.

We further show that ensembles can greatly improve the performance of MOR on all datasets: a smaller number of epochs is needed to reach GE 1 than in the case of the best MOR model [17]. However, ensembles only work in the case of a reduced-size hyperparameter space. A large space, in the case of MOR, results in a small success rate, and ensembles cannot help if most of the models result in unsuccessful attacks.

In the context of DDLA, the use of ensembles is not practical due to the structure of the methodology. Specifically, a single DDLA model consists of 256 neural networks. Consequently, if an ensemble comprising 20 models were to be used, it would require the training of a staggering total of 5 120 neural networks.

To conclude, upon conducting a comparative analysis of the two methodologies, it becomes apparent that the MOR technique exhibits significant potential in the domain of side-channel analysis. Characterized by its employment of unsupervised deep learning principles, MOR uses the Hamming weight and Identity leakage models. Notably, MOR achieves favorable outcomes across various datasets, demonstrating success in at least one leakage model for all examined datasets. In contrast, the DDLA methodology falls short in breaking the AES_HD dataset. Furthermore, MOR uses one neural network

Table 4.6: Summary of the evaluation of DDLA and MOR approaches.

	DDLA	MOR
Success rate (SR)	- for 3/4 of datasets, approx. 100% success rate - not able to break 1 dataset due to overfitting - distinguishing based on validation data shows better performance	- the highest success rate is for HW and is approx 30% - distinguishing based on validation data shows better performance - HW leakage model requires fewer input traces than ID - sensitive to hyperparameters
Guessing entropy (GE)	- for 3/4 datasets, 15,000 input traces are enough to reach the minimum GE - similarly, 15 epochs are enough to reach the minimum GE	- 20,000 input traces are enough to reach minimum GE for most DA and HW - similarly, 20 epochs are enough to reach the minimum GE
Desynchronization	- computationally expensive, SR close to 0% for a large hyperparameter space - random search needed for t he best model	- SR 0% for a large hyperparameter space - random search is needed for the best model
Ensembles	- computationally infeasible	- improving the GE when a small number of input traces is provided - computationally expensive for CNNs
Data augmentation (DA)	- when the distinguisher is based on training data, DA improves the SR	- when the distinguisher is based on training data, DA improves the SR - Shifting DA reduces the SR - Gaussian noise works better

featuring 256 output branches and proves superior efficiency as DDLA incurs substantial overhead by employing 256 separate neural networks. However, it is crucial to note that MOR's performance relies highly on its hyperparameters, making it sensitive to their values. The summary of this evaluation can be found in Table 4.6.

Chapter 5

Multi-output Regression Enhanced

In this chapter, we introduce a novel non-profiled deep learning-based side-channel analysis approach that builds on the state-of-the-art work of Do *et al.* [17]. We start by evaluating different loss functions in the context of multi-output regression, followed by introducing a novel key distinguisher. Moreover, we propose a function that can be used for hyperparameter tuning in a non-profiled context. Finally, we evaluated the novel attack in terms of success rate and guessing entropy for six datasets, as well as for desynchronized traces.

5.1 Motivation

In 2022, Do *et al.* introduced Multi-output regression (MOR) as an alternative non-profiled deep learning-based side-channel attack to Timon's computationally expensive DDLA [67]. It showed great potential by being the first non-profiled deep learning-based SCA that used the Identity leakage model. However, in the same work, the authors also introduced multi-output classification (MOC). It mostly relies on Timon's idea of using the LSB leakage model, but instead of using 256 networks, MOC uses only one. Although MOC has its merits, we decided to focus on MOR due to its novelty and the potential it demonstrated in utilizing the Identity leakage model. However, we acknowledge that there is value in conducting research in the context of MOC as well. Despite its simplifications, MOC presents an interesting research direction and may have unique advantages in certain scenarios. Therefore, we believe that further work should be conducted to explore the effectiveness and potential of MOC in deep learning-based side-channel attacks. By narrowing our focus to MOR while acknowledging the need for research in MOC, we aimed to provide a comprehensive understanding of non-profiled deep learning-based SCAs and contribute to advancing this field.

In Chapter 4, we evaluated the performance of the MOR methodology [17] and identified its weaknesses. More specifically, we experimentally showed the sensitivity to hyperparameters that it exhibits: averaging over a large search space significantly reduces the success rate of the attack. On the other hand, MOR in side-channel analysis shows greater potential than DDLA by using one multi-output network rather than 256 to distinguish between the key candidates.

In this chapter, we aim to enhance the MOR approach by investigating the impact of the distinguisher on its success rate. More specifically, we look into different loss functions and their effects on performance, as well as develop a metric for hyperparameter tuning. While previous studies have explored evaluating optimal loss functions in supervised profiled SCA [30], no prior research has investigated this aspect for unsupervised deep learning-based SCA. Therefore, we aim to fill this research gap and explore the significance of different loss functions in unsupervised deep learning-based SCA using MOR.

Moreover, neither non-profiled deep learning methodologies, DDLA nor MOR provide any discussion towards finding the best model from a hyperparameter search space. Therefore, we aim to find a suitable objective function for hyperparameter tuning in the context of unsupervised non-profiled SCA.

Table 5.1: Success rate (SR) of MOR on ASCADf dataset using the Hamming weight (HW) leakage model and different loss functions. The number of traces is fixed at 20 000. We considered the loss of validation or training data as the distinguishers.

	MSE	z-MSE	MAE	z-MAE	Huber	z-Huber	LogCosh	z-LogCosh	Tukey's	z-Tukey's
validation	15.68%	94.95%	8.14%	3.75%	16.29%	18.46%	14.5%	12.5%	16.3%	17.3%
training	27.31%	92.5%	9.21%	6.25%	18.35%	8.54%	25.8%	23.2%	32.5%	26.7%

Table 5.2: Success rate (SR) of MOR on ASCADr dataset using the Hamming weight (HW) leakage model and different loss functions. The number of traces is fixed at 40 000. We considered the loss of validation or training data as the distinguishers.

	MSE	z-MSE	MAE	z-MAE	Huber	z-Huber	LogCosh	z-LogCosh	Tukey's	z-Tukey's
validation	3.9%	91.2%	1.5%	1.0%	2.2%	8.2%	1.7%	3.0%	2.4%	17.9%
training	30.9%	77.2%	15.9%	9.0%	20.8%	12.2%	33.4%	22.6%	29.4%	30.9%

This chapter answers the following research question:

Can we derive a new methodology for deep learning-based non-profiled side-channel analysis that outperforms state-of-the-art work?

- What are the necessary elements for a new technique to build a non-profiled side-channel attack?
- How does it compare against the existing one in terms of different datasets: ASCADf, ASCADr, AES_HD, AES_RD, desynchronized ASCADf?
- Can we define an objective function that can be used in hyperparameter tuning?

5.2 Finding the best loss function

The current MOR approach uses only the mean squared error (MSE) loss function in their experiments [17]. In the case of non-profiled SCA with the MOR approach, the learning process differs from typical multi-output regression models. Instead of minimizing the empirical error for all outputs, the MOR model specifically aims to minimize the empirical error for a single output, as only one output is associated with the correct key candidate. Therefore, it becomes crucial to identify an efficient loss function that is both sensitive to outliers and capable of reducing the error even further in the presence of outliers. This sensitivity to outliers is desirable from an adversary's perspective since outliers represent the attacker's best estimation of the correct key candidate.

Outliers typically refer to variables or data points that significantly deviate from the mean of the data set. In the context of the MOR model, an effective loss function should assign similar loss values to the incorrect keys while designating the correct key as an outlier with a lower loss value than the rest.

In this section, we aim to investigate the success rate of the MOR-based SCA by considering different loss functions. We compare 10 different loss functions in terms of the success rate they show on ASCADf, for both Hamming weight and Identity leakage models: MSE, MAE, Huber, LogCosh, Tukey's, Z-score MSE, Z-score MAE, Z-score Huber, Z-score LogCosh, and Z-score Tukey's. All these loss functions are introduced in 2.1.2. We randomly drew 1 000 models from the hyperparameter space defined in Table 4.3 and computed the success rate based on these models.

Table 5.1 shows the success rate of the MOR approach on ASCADf considering different loss functions and the Hamming weight leakage model. We use the training and validation loss to distinguish between the correct and the other key candidates. We see that the Z-score MSE, denoted by Z-MSE,

Table 5.3: Success rate (SR) of MOR on ASCADf dataset using the Identity (ID) leakage model and different loss functions. The number of traces is fixed at 40 000. We considered the loss of validation or training data as the distinguishers.

	MSE	z-MSE	MAE	z-MAE	Huber	z-Huber	LogCosh	z-LogCosh	Tukey's	z-Tukey's
validation	0%	74.1%	0.6%	0.3%	0%	0.8%	0%	0.6%	0%	5.6%
training	25.7%	77.6%	35.8%	14%	36.1%	10.4%	33.7%	11.6%	0%	10.2%

Table 5.4: Success rate (SR) of MOR on ASCADr dataset using the Identity (ID) leakage model and different loss functions. The number of traces is fixed at 40 000. We considered the loss of validation or training data as the distinguishers.

	MSE	z-MSE	MAE	z-MAE	Huber	z-Huber	LogCosh	z-LogCosh	Tukey's	z-Tukey's
validation	1.2%	22.2%	1.3%	0%	0%	0.4%	1.1%	0%	0%	1.7%
training	4.5%	9.4%	2.6%	3.6%	0%	2.2%	2.2%	2.8%	0%	7.0%

achieves excellent performance: almost 95% success when the Z-MSE validation loss is taken as the distinguisher.

Similarly, we conducted the same experiment on the ASCADr dataset. We present the results in Table 5.2. As before, Z-MSE achieves the best success rate. We see here that using the validation Z-MSE loss as the distinguisher performs better than the same loss on the training data.

Table 5.3 presents the success rate of the MOR approach on ASCADf considering different loss functions and the Identity leakage model. As before, the Z-MSE loss function achieves the best results: the highest success rate of 77.4% when the validation loss is the distinguisher and 77.6% when the training loss is the distinguisher. Because of this difference in SR between HW and ID leakage models, we further increased the number of traces to 50 000 and evaluated the outcome for ID. We obtained 86% SR for Z-MSE on validation and 85% for training data.

We conducted a similar experiment on ASCADr, with the Identity leakage model; see Table 5.4. We obtain the highest success rate for the Z-MSE loss function, as in the case of ASCADf. Differently, however, is the fact that the success rate is much lower, and we obtain better results when we use the validation Z-MSE to retrieve the correct key. We believe this is the case, more specifically, that for ASCADf, the training loss results in a better key rank since it overfits that dataset.

In conclusion, the Z-MSE score provides the best results when the MOR methodology is applied in the context of unsupervised side-channel analysis. In the remainder of this chapter, we refer to the MOR approach that uses the Z-MSE loss function as multi-output regression enhanced (MORE).

5.3 Objective function for hyperparameter tuning

The MOR models we employ in our study are deep neural networks; thus, there is a need to tune their hyperparameters. Hyperparameter tuning already presents a complex challenge in profiled SCA, where an objective function can be defined based on metrics extracted from a labeled validation set. In profiled SCA, the primary objective is to identify a deep neural network that exhibits higher generalization, thereby enabling a more efficient attack. This entails computing SCA metrics like guessing entropy or success rate and selecting a model that requires fewer validation traces to successfully recover the correct key candidate.

However, in a non-profiled setting, the absence of validation traces where we know the encryption key poses a significant challenge. Consequently, obtaining a model that yields optimal SCA metrics becomes unfeasible. We explore two metrics as distinguishers, that retrieve the correct key and rank the key candidates. The first approach involves sorting the loss function and selecting the key candidate associated with the minimum loss value, as it was originally done for MOR [17]. We see the success rates for this approach for ASCADf and ASCADr in the tables above.

The second approach to retrieve the correct key calculates the Pearson correlation between predicted and correct labels for each key byte candidate and selects the key that corresponds to the highest correlation value. We introduce the Pearson correlation formula in Eq 5.1. We calculate the success rate when the optimizer uses different loss functions in our neural network when the distinguisher is the Pearson correlation on validation or training data rather than the loss function values.

$$C(y,\hat{y}) = \frac{\sum_{i=1}^{n}(\hat{y}_i - \mu(\hat{y}))(y_i - \mu(y))}{\sqrt{\sum_{i=1}^{n}(\hat{y}_i - \mu(\hat{y}))^2}\sqrt{\sum_{i=1}^{n}(y_i - \mu(y))^2}} \tag{5.1}$$

Table 5.5: Success rate (SR) of MOR on ASCADf dataset using the Hamming weight (HW) leakage model and different loss functions. The number of traces is fixed at 20 000. We considered the Pearson correlation on validation or training data as the distinguishers.

	MSE	z-MSE	MAE	z-MAE	Huber	z-Huber	LogCosh	z-LogCosh	Tukey's	z-Tukey's
validation	45.9%	94.9%	37.7%	6.7%	48.5%	29.%	46.1%	31.6%	50.1%	43.%
training	51%	92.5%	43.3%	10.1%	55.9%	35.7%	53.%	37.%	56.1%	49.4%

Table 5.6: Success rate (SR) of MOR on ASCADf dataset using the Identity (ID) leakage model and different loss functions. The number of traces is fixed at 40 000. We considered the Pearson correlation on validation or training data as the distinguishers.

	MSE	z-MSE	MAE	z-MAE	Huber	z-Huber	LogCosh	z-LogCosh	Tukey's	z-Tukey's
validation	25.7%	73.9%	29.2%	22.8%	29.0%	13.7%	34.3%	21.0%	0%	25.1%
training	37.5%	77.6%	43.4%	26.1%	44.5%	16.1%	46.2%	20.4%	0%	26.1%

Table 5.5 shows the success rate of the MOR approach on ASCADf for different loss functions and HW leakage. The Pearson correlation distinguishers between the keys on either training or validation data. We see here that the SR is significantly higher for most loss functions, compared to the results we obtained when the loss values are employed as the distinguishers: see Table 5.1.

Moreover, we conducted the same experiment on ASCADf with the Identity leakage model. The success rates we obtain when we use the loss function values as the distinguishers are not large, as we show in Table 5.3. More specifically, the success is close to 0% in some cases. However, when the Pearson correlation is distinguisher, we see in Table 5.6 that the SR significantly increases.

Table 5.7 shows the success rate of MOR on ASCADr using different loss functions and Pearson correlation as key distinguisher on both validation and training data. Compared to Table 5.2, we see a significant improvement in the success rate for most loss functions. For example, when Huber is the loss function, the SR increases from 16% to 39% for validation data.

Similarly, when we use the ID leakage model in the case of ASCADr, we also observe an increase in the success rate. Table 5.8. When the loss function values are used as the key distinguisher, the success rate for Huber loss is 0%. However, in the case of the Pearson correlation distinguisher, the SR increases to 10% on validation data.

These results show that the Pearson correlation distinguisher provides better results in terms of the success rate than the loss function distinguisher for most loss functions. We choose to use the Z-MSE loss value as the distinguisher in the remainder of this work since the success rate stays the same between these two distinguishers. However, whenever we use another loss function in MOR, we recommend distinguishing the keys based on the correlation values. Thus, one can use this to define an objective function for hyperparameter tuning: the best model corresponds to the highest correlation for all keys. To avoid overfitting, we employ this approach on validation data rather than training data. In the remainder of the paper, the best MORE model is selected according to the highest Pearson correlation from all key candidates.

In summary, the main elements of the MORE approach that enhance performance compared to the MOR methodology can be identified as follows: employing Pearson correlation or the Z-score normalized MSE loss values as the key distinguishing metrics, utilizing Z-score normalized MSE as the loss function, and using a validation set rather than the training data on which one distinguishes the keys.

Table 5.7: Success rate (SR) of MOR on ASCADr dataset using the Hamming weight (HW) leakage model and different loss functions. The number of traces is fixed at 40 000. We considered the Pearson correlation on validation or training data as the distinguishers.

	MSE	z-MSE	MAE	z-MAE	Huber	z-Huber	LogCosh	z-LogCosh	Tukey's	z-Tukey's
validation	36.6%	90.2%	28.4%	8.7%	39.0%	18.4%	46.0%	22.4%	33.5%	36.4%
training	31.5%	77.2%	27.5%	8.7%	33.4%	21.6%	41.4%	28.3%	29.8%	41.5%

Table 5.8: Success rate (SR) of MOR on ASCADr dataset using the Identity (ID) leakage model and different loss functions. The number of traces is fixed at 40 000. We considered the Pearson correlation on validation or training data as the distinguishers.

	MSE	z-MSE	MAE	z-MAE	Huber	z-Huber	LogCosh	z-LogCosh	Tukey's	z-Tukey's
validation	10.8%	20.5%	8.8%	3.9%	10%	3.5%	10.%	4%	0%	7.6%
training	4.3%	9.4%	3.5%	4.8%	2%	5.4%	2.3%	2.8%	0%	9.2%

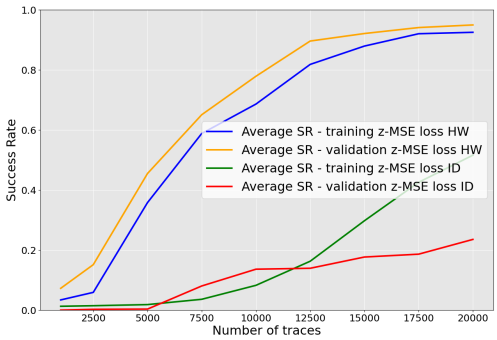


Figure 5.1: Success rate (SR) of the MORE methodology on the ASCADf dataset varying the number of traces. Both Identity and Hamming weight leakage models are considered. The key distinguisher is the z-MSE loss on training or validation data.

5.4 Experimental Setup

We tested the enhanced MOR approach (MORE) on 6 different datasets: ASCADf, AES_RD, AES_HD, ASCADr, ASCADf raw, and ASCADr raw. For every experiment in the remainder of this chapter regarding MLPs, we randomly drew 2 500 models from the hyperparameter space defined in Table 4.3. In the case of desynchronized ASCADf, we randomly chose 500 models from the hyperparameter space introduced in Table 4.4.

For the experiments involving data augmentation (DA), we evaluated two DA techniques: shifting and Gaussian noise. We either doubled the number of original traces if this amount is less than 10 000, or we augmented it with 10 000 otherwise.

For the experiments involving ensembles, we chose three sizes: made up of 20, 50, or 100 models. Each ensemble experiment was run 50 times. The ensembles are compared to the best MORE model. This model is chosen based on the maximum objective function value (Pearson's correlation), see Eq 5.1.

5.5 Results

5.5.1 ASCADf

Number of traces

We varied the number of input traces and calculated the success rate of MORE on ASCADf. We show the results in Figure 5.1. We see that 20 000 traces are enough in the case of HW to break the secret key. However, in the case of ID leakage, it is more difficult for MORE to differentiate between the key candidates: overfitting happens. Figure 5.2 shows that for Identity leakage, a larger number of traces is needed to reach a higher success rate due to the overfitting of the models with this leakage.

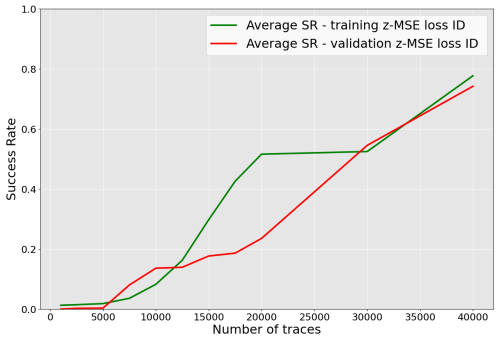
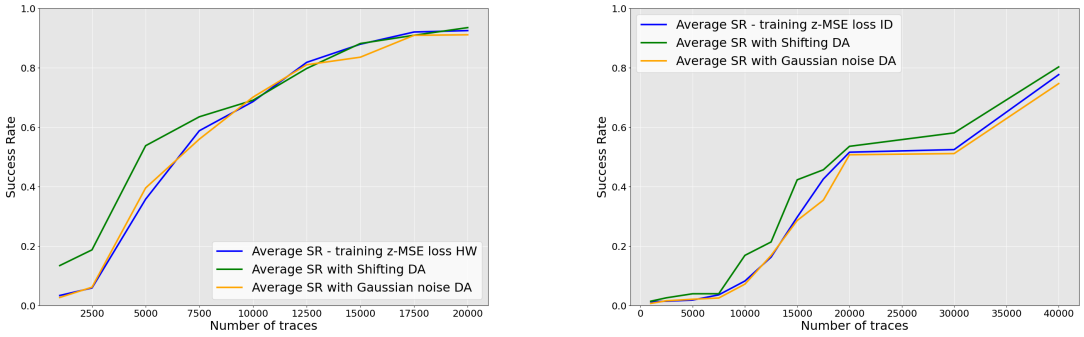


Figure 5.2: Success rate (SR) of the MORE methodology on the ASCADf dataset varying the number of traces for the ID leakage model. The key distinguisher is the z-MSE loss on training or validation data.



(a) `ASCADf`, SR with and without DA for HW.

(b) `ASCADf`, SR with and without DA for ID.

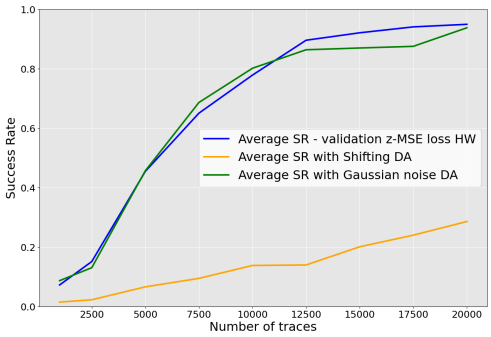
Figure 5.3: Success rate (SR) for MORE-MLP on the ASCADf dataset for the Identity (ID) and Hamming weight (HW) leakage models. We considered the training z-MSE loss as the distinguisher. Two different data augmentation techniques were used: shifting and Gaussian noise

Data augmentation

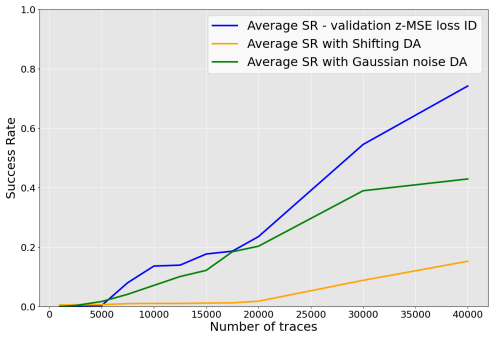
Figure 5.3 illustrates the average guessing entropy of the MORE approach, both before and after the application of data augmentation, when the training z-MSE loss is employed as the distinguishing factor. The figure clearly demonstrates that Gaussian noise data augmentation enhances the overall success rate, particularly when the number of traces is less than 10 000. In the case of HW leakage, there is a slight improvement observed with shifting data augmentation. This implies that data augmentation effectively mitigates overfitting issues when we use the training z-MSE loss function as the distinguishing factor.

Conversely, Figure 5.4 presents the same experiment but with the z-MSE validation loss as the distinguishing factor. It is evident from the figure that the success rate has not improved by Gaussian noise data augmentation, but it also does not impact the performance of MORE. However, when shifting data augmentation is employed, there is a significant reduction in the success rate for both leakage models.

Although there is no substantial improvement in the success rate of MORE with the application of data augmentation, it is worth noting that the average guessing entropy decreases when DA is employed. Figure 5.5 illustrates that Gaussian noise data augmentation yields better results compared to shifting, leading to a slight reduction in the overall guessing entropy across 40 training epochs.



(a) `ASCADf`, SR with and without DA for HW.



(b) `ASCADf`, SR with and without DA for ID.

Figure 5.4: Success rate (SR) for MORE-MLP on the ASCADf dataset for the Identity (ID) and Hamming weight (HW) leakage models. We considered the validation z-MSE loss as the distinguisher. Two different data augmentation techniques were used: shifting and Gaussian noise

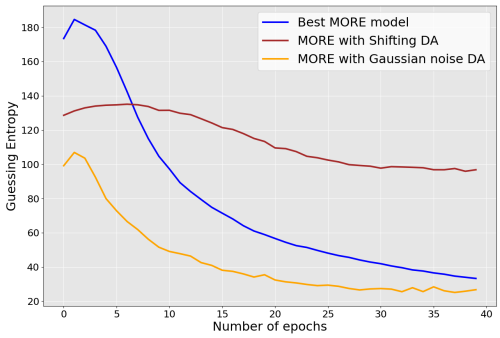
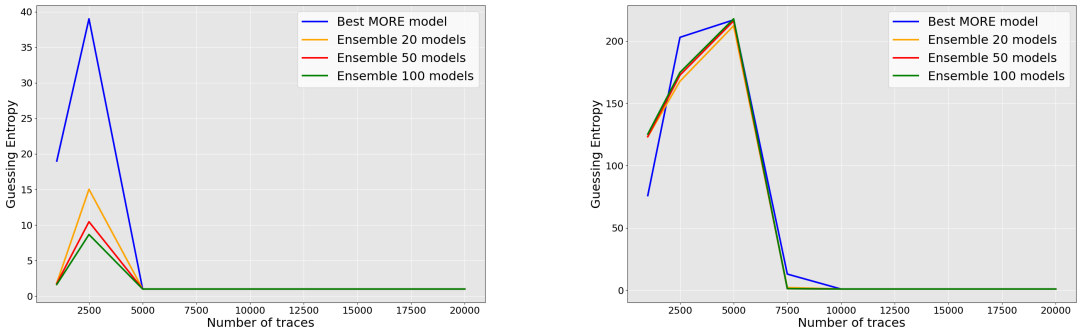


Figure 5.5: Guessing entropy (GE) of the MORE methodology on the ASCADf dataset varying the number of epochs for the HW leakage model with data augmentation (DA). The number of original traces is fixed at 5 000. The key distinguisher is the Pearson correlation on validation data.



(a) `ASCADf`, GE for HW.

(b) `ASCADf`, GE for ID.

Figure 5.6: Guessing entropy for MORE-MLP on the ASCADf dataset for the Identity (ID) and Hamming weight (HW) leakage models. The number of traces is fixed at 20 000. Three different ensemble sizes were used: 20, 50, or 100 models.

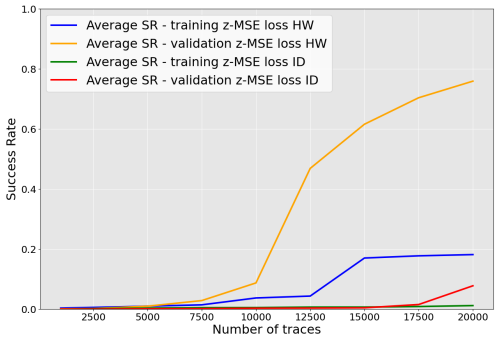


Figure 5.7: Success rate (SR) of the MORE methodology on the AES_RD dataset varying the number of traces. Both Identity and Hamming weight leakage models are considered. The key distinguisher is the z-MSE loss on training or validation data.

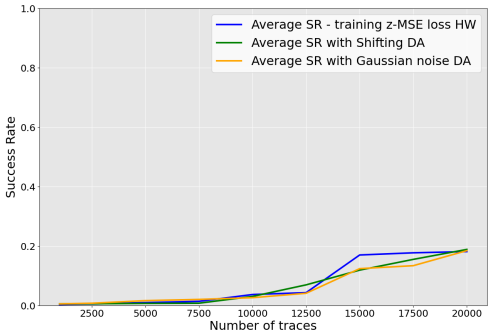
Ensembles

Figure 5.6 presents the average guessing entropy of the MORE approach applied to the ASCADf dataset for ensembles and the best MORE model. Specifically, Figure 5.6a focuses on the Hamming weight leakage model. It is evident from the figure that when ensembles are used, the guessing entropy decreases, particularly when we reduce the number of input traces. This reduction in guessing entropy is more pronounced when compared to the performance of the best model. In the case of the Identity leakage model (Figure 5.6b), it requires a slightly lower number of traces to achieve a guessing entropy of 1: specifically, 7,500 traces are needed, whereas the best model requires 10 000 traces.

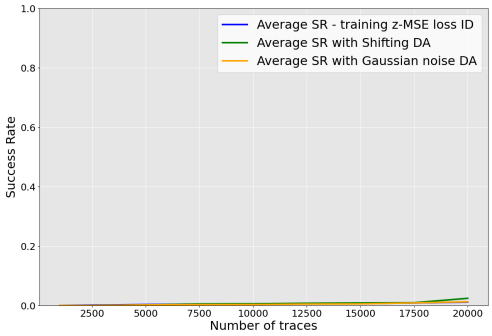
5.5.2 AES_RD

Number of traces

We further examine the impact of the number of traces on the success rate of the MORE approach applied to AES_RD. Figure 5.7 shows the results we obtained from this investigation. Notably, the validation Z-MSE loss demonstrates superior performance for the HW leakage model, achieving an impressive success rate of almost 80% when 20 000 input traces are used. Conversely, the success rate for the ID leakage model remains relatively low, similar to the observations made with MOR. This can be attributed to the inherent vulnerability of AES_RD to overfitting, as this dataset is comparatively easier to break.

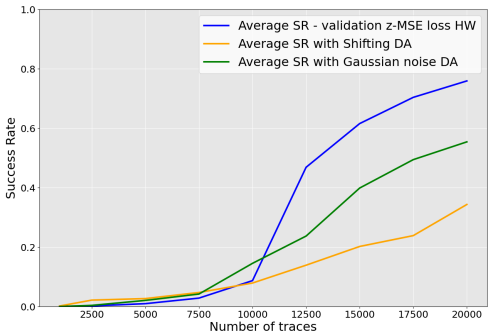


(a) AES_RD, SR with and without DA for HW.

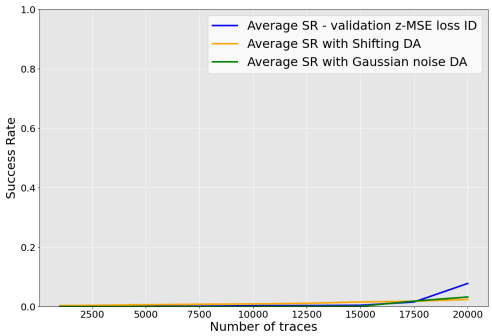


(b) AES_RD, SR with and without DA for ID.

Figure 5.8: Success rate (SR) for MORE-MLP on the AES_RD dataset for the Identity (ID) and Hamming weight (HW) leakage models. We considered the training z-MSE loss as the distinguisher. Two different data augmentation techniques were used: shifting and Gaussian noise.



(a) AES_RD, SR with and without DA for HW.



(b) AES_RD, SR with and without DA for ID.

Figure 5.9: Success rate (SR) for MORE-MLP on the AES_RD dataset for the Identity (ID) and Hamming weight (HW) leakage models. We considered the validation z-MSE loss as the distinguisher. Two different data augmentation techniques were used: shifting and Gaussian noise.

Data augmentation

As in the case of ASCADf, Gaussian noise data augmentation shows a slight improvement in the success rate of MORE on AES_RD when we use the training Z-MSE as the distinguisher and the leakage model is HW, see Figure 5.8a. However, as we see in Figure 5.8b, DA does not improve the low success rate of MORE for the ID leakage model.

We see again in Figure 5.9 that for the validation z-MSE distinguisher, data augmentation results in a significant drop in the success rate. However, as we previously saw in the case of ASCADf, DA helps in reducing the average GE of MORE. Figure 5.10 shows that both data augmentation techniques significantly reduce the average GE of MORE.

Ensembles

Figure 5.11 shows the average guessing entropy of the best MORE model and ensembles. For the Hamming weight leakage model, the GE reaches a minimum faster than the best model: fewer traces are needed. Furthermore, the GE is not as high as for the best model when we use a smaller number of traces.

We previously saw that MORE cannot break AES_RD with the Identity leakage due to overfitting. As expected, ensembles do not help either. Figure 5.11b shows that the GE is overall smaller for ensem-

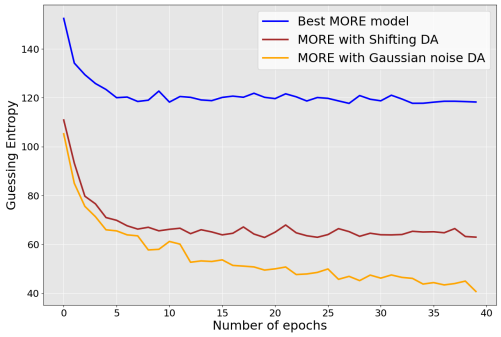
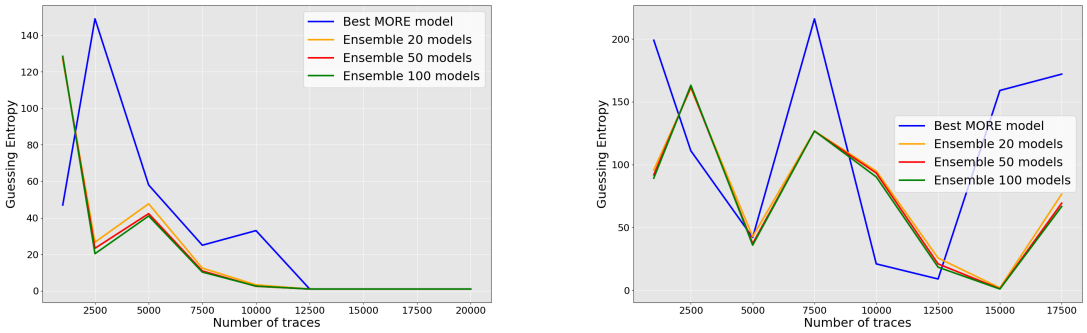


Figure 5.10: Guessing entropy (GE) of the MORE methodology on the AES_RD dataset varying the number of epochs for the HW leakage model with data augmentation (DA). The number of original traces is fixed at 5 000. The key distinguisher is the Pearson correlation on validation data.



(a) AES_RD, GE for HW.

(b) AES_RD, GE for ID.

Figure 5.11: Guessing entropy (GE) for MORE-MLP on the AES_RD dataset for the Identity (ID) and Hamming weight (HW) leakage models. The number of traces is fixed at 20 000. Three different ensemble sizes were used: 20, 50, or 100 models.

bles than for the best model. However, there is a high variance, and GE 1 is not guaranteed.

5.5.3 AES_HD

Number of traces

We evaluated the influence of the number of traces on the success rate of MORE on the AES_HD dataset. In Figure 5.12, it can be observed that when we use a smaller number of traces, the performance of the distinguisher on the training data is superior to that on the validation data. This observation aligns with expectations since, with a limited number of traces, there is a higher likelihood of overfitting occurring.

Data augmentation

In our study, we employed two data augmentation techniques and examined their impact on the success rate of the MORE model for AES_HD.

Figure 5.13 illustrates that when we use the training Z-MSE loss as the distinguisher, data augmentation (DA) does not appear to have a significant influence on the success rate. However, when considering validation data, there is a notable improvement observed for both Gaussian noise and shifting augmentation techniques.

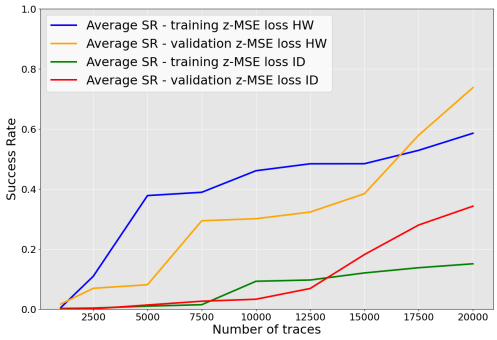
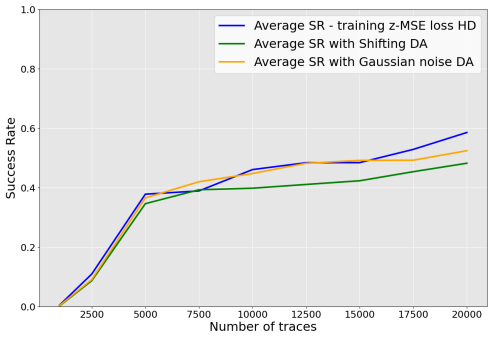
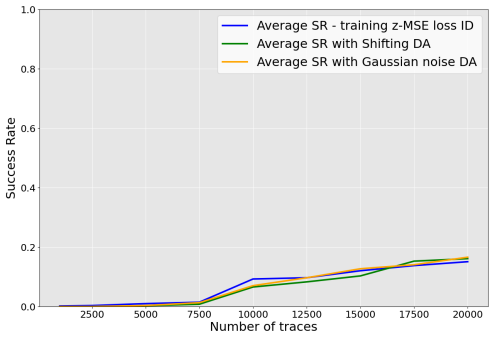


Figure 5.12: Success rate (SR) of the MORE methodology on the AES_HD dataset varying the number of traces. Both Identity (ID) and Hamming distance (HD) leakage models are considered. The key distinguisher is the z-MSE loss on training or validation data.



(a) AES_HD, SR with and without DA for HD.



(b) AES_HD, SR with and without DA for ID.

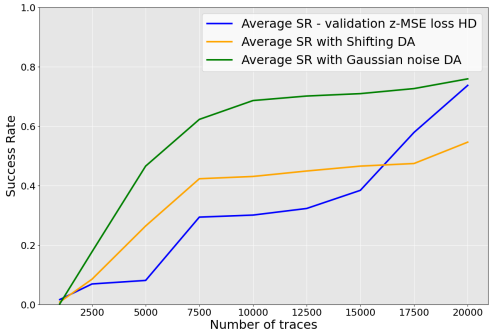
Figure 5.13: Success rate (SR) for MORE-MLP on the AES_HD dataset for the Identity (ID) and Hamming distance (HD) leakage models. We considered the training z-MSE loss as the distinguisher. Two different data augmentation techniques were used: shifting and Gaussian noise.

Furthermore, Figure 5.14 shows that for both leakage models, the success rate more than doubles when Gaussian noise data augmentation is applied, compared to when it is not used.

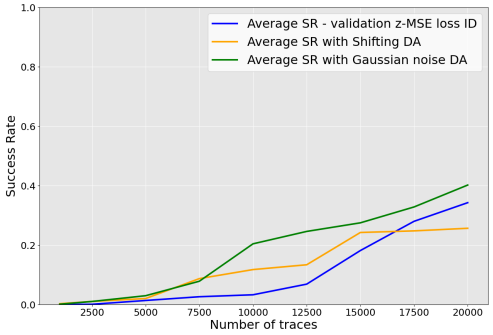
Additionally, the guessing entropy (GE) decreased by approximately a half when shifting data augmentation is combined with the MORE model, as depicted in Figure 5.15. It is worth noting that Gaussian noise data augmentation consistently yields the best results, with the GE remaining below 10 between 5 and 10 training epochs.

Ensembles

We plotted the guessing entropy and compared the best MORE model against ensembles for the AES_HD dataset. Figure 5.16a plots the results for the Hamming distance leakage model. By using ensembles, MORE can reach minimum GE with only 2 500 traces. Even for 1 000 traces, the GE is low: around 15. Similarly, Figure 5.16b shows that ensembles outperform the best model for Identity leakage: 7 500 traces are enough to break this dataset, while the best model requires 12 500.



(a) AES_HD, SR with and without DA for HD.



(b) AES_HD, SR with and without DA for ID.

Figure 5.14: Success rate (SR) for MORE-MLP on the AES_HD dataset for the Identity (ID) and Hamming distance (HD) leakage models. We considered the validation z-MSE loss as the distinguisher. Two different data augmentation techniques were used: shifting and Gaussian noise.

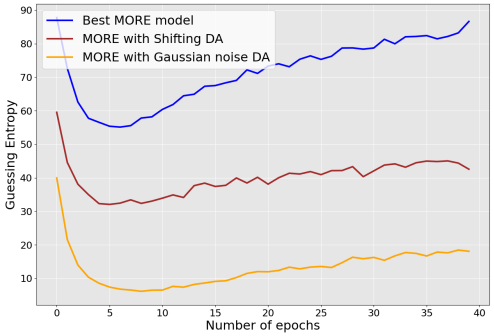
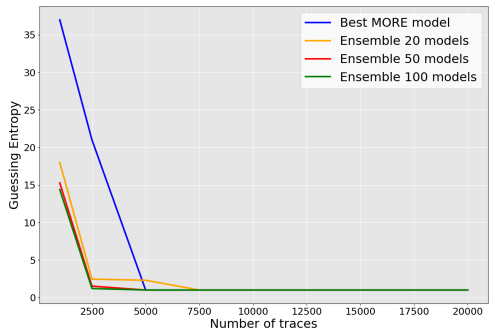
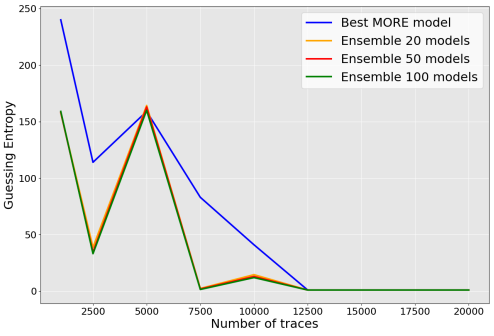


Figure 5.15: Guessing entropy (GE) of the MORE methodology on the AES_HD dataset varying the number of epochs for the HW leakage model with data augmentation (DA). The number of original traces is fixed at 5 000. The key distinguisher is the Pearson correlation on validation data.



(a) AES_HD, GE for HD.



(b) AES_HD, GE for ID.

Figure 5.16: Guessing entropy for MORE-MLP on the AES_HD dataset for the Identity (ID) and Hamming distance (HD) leakage models. The number of traces is fixed at 20 000. Three different ensemble sizes were used: 20, 50, or 100 models.

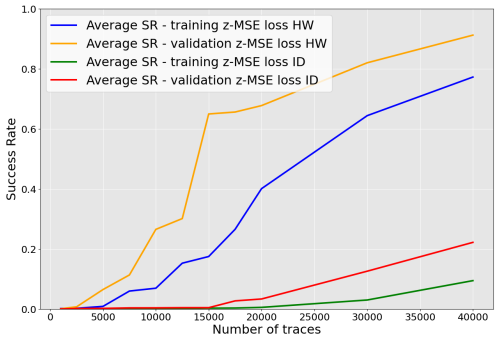


Figure 5.17: Success rate (SR) of the MORE methodology on the ASCADr dataset varying the number of traces. Both Identity and Hamming weight leakage models are considered. The key distinguisher is the z-MSE loss on training or validation data.

5.5.4 ASCADr

Number of traces

We varied the number of traces and plotted the success rate of MORE on ASCADr. Figure 5.17 shows that for the Hamming weight leakage model, we obtain almost 100% success rate for 40 000 traces, and validation Z-MSE loss distinguisher. However, for ID, 40 000 traces are not enough: the SR stays below 30%.

Data augmentation

We further evaluated two data augmentation techniques and plotted the influence on the success rate of MORE for ASCADr.

Figure 5.18 shows that shifting DA slightly improves the GE when we use training z-MSE loss as the distinguisher. Surprisingly, the Gaussian noise does not perform as well as in the case of the other datasets: the success rate is dropped. However, this can be explained by the fact that overfitting decreases after augmenting the dataset.

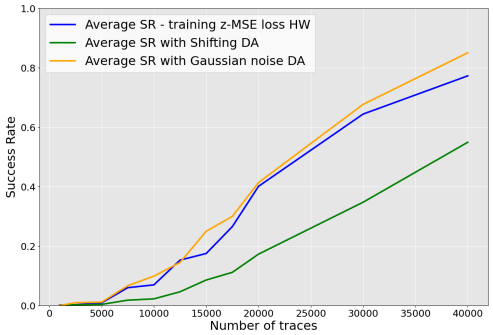
Moreover, the GE is dropped when we employ Gaussian noise DA together with MORE, and the validation Z-MSE is the distinguisher, as shown in Figure 5.19. Shifting DA produces an unsuccessful attack: the SR drops below 40% for 40 000 input traces when we use HW leakage. A similar result can be observed for Identity leakage: DA affects the SR.

These results were further confirmed when we plotted the average GE before and after DA for ASCADr. Figure 5.20 shows that the GE goes down with either data augmentation technique. Shifting DA performs, however, worse.

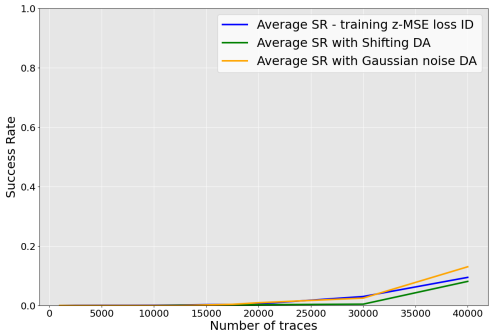
Ensembles

We plotted the guessing entropy and compared the best MORE model against ensembles for the ASCADr dataset. We did not include the results for the Identity leakage since we observed no improvement due to the low success rate we obtain when we use this leakage model.

Figure 5.21 plots the results for the Hamming weight leakage model. Ensembles outperform the best MORE model by reaching a slightly lower GE. However, for both the best model and ensembles, 10 000 are needed to break the secret key.

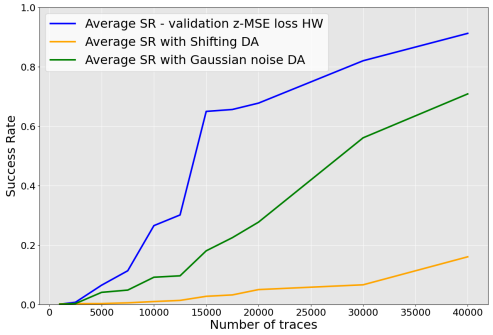


(a) ASCADr, SR with and without DA for HW.

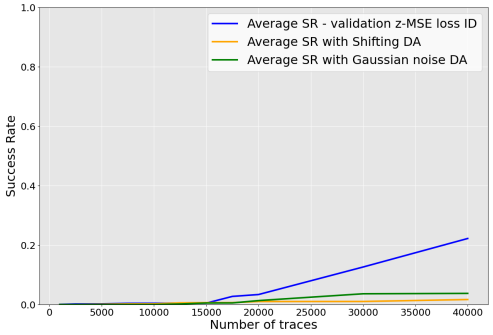


(b) ASCADr, SR with and without DA for ID.

Figure 5.18: Success rate (SR) for MORE-MLP on the ASCADr dataset for the Identity (ID) and Hamming weight (HW) leakage models. We considered the training z-MSE loss as the distinguisher. Two different data augmentation techniques were used: shifting and Gaussian noise.



(a) ASCADr, SR with and without DA for HW.



(b) ASCADr, SR with and without DA for ID.

Figure 5.19: Success rate (SR) for MORE-MLP on the ASCADr dataset for the Identity (ID) and Hamming weight (HW) leakage models. We considered the validation z-MSE loss as the distinguisher. Two different data augmentation techniques were used: shifting and Gaussian noise.

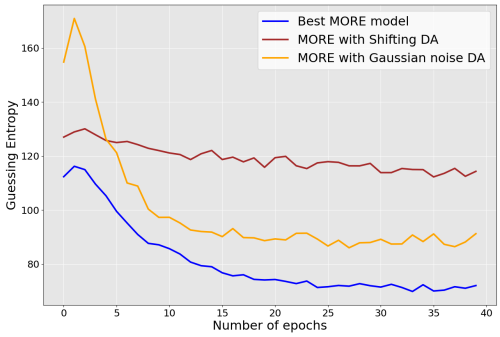


Figure 5.20: Guessing entropy (GE) of the MORE methodology on the ASCADr dataset varying the number of epochs for the HW leakage model with data augmentation (DA). The number of original traces is fixed at 5 000. The key distinguisher is the Pearson correlation on validation data.

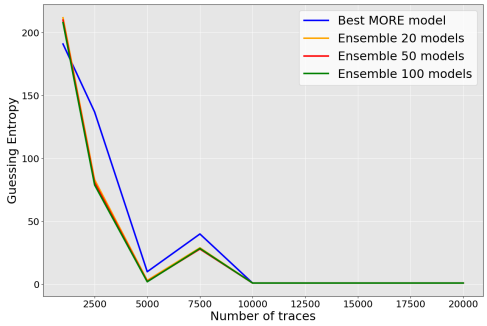


Figure 5.21: Guessing entropy for MORE-MLP on the ASCADr dataset for the Hamming weight (HW) leakage model. The number of traces is fixed at 20 000. Three different ensemble sizes were used: 20, 50, or 100 models.

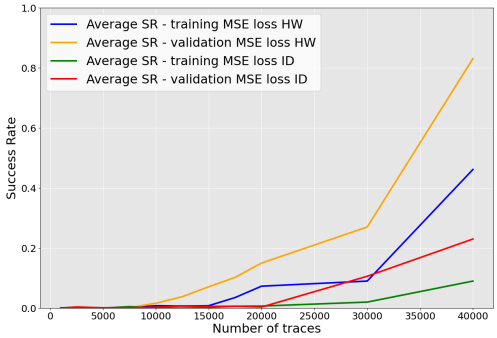


Figure 5.22: Success rate (SR) of the MORE methodology on the ASCADf raw traces varying the number of traces. Both Identity and Hamming weight leakage models are considered. The key distinguisher is the z-MSE loss on training or validation data.

5.5.5 ASCADf raw

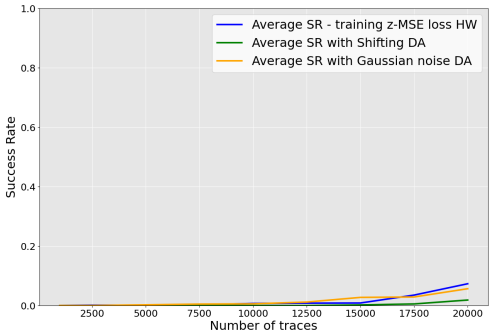
Number of traces

Figure 5.22 shows the influence of the number of input traces on the success rate of the attack. We observe that for both leakage models, when the validation Z-MSE is the distinguisher, we achieve better results than the training loss distinguisher. Compared to ASCADf, the raw dataset require more input traces to achieve a higher success rate: while MORE with HW worked with almost 100% SR on ASCADf, for ASCADf raw, it requires 40 000.

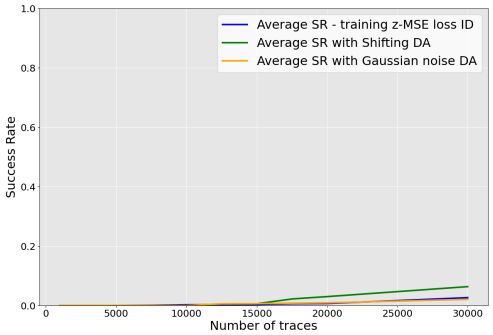
Data augmentation

We further evaluated two data augmentation techniques and plotted the influence on the success rate of MORE for the ASCADf raw dataset. Figure 5.24 demonstrates that in the case of the Hamming weight leakage model, shifting data augmentation slightly improves the success rate (SR) when we employ the training Z-MSE loss as the distinguisher. On the other hand, for the Identity leakage model, Gaussian noise data augmentation performs better, as shown in Figure 5.23b. However, it is important to note that we observe no significant improvements when data augmentation is employed.

Similar to the findings in ASCADf, data augmentation does not provide significant benefits when we use the validation loss as the distinguisher. Figure 5.24 reveals that for the Hamming weight leakage model, the success rate (SR) decreases closer to 0% when we employ data augmentation in combination with the MORE model.

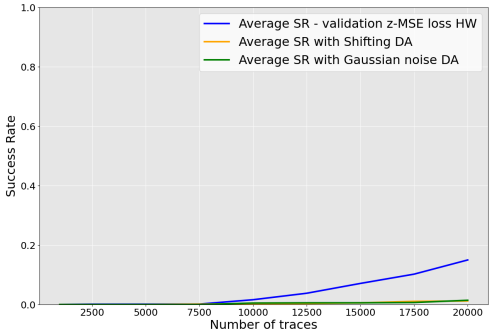


(a) `ASCADf raw`, SR with and without DA for HW.

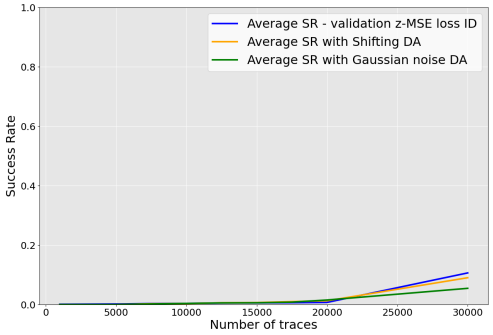


(b) `ASCADf raw`, SR with and without DA for ID.

Figure 5.23: Succes rate (SR) for MORE-MLP on the ASCADf raw dataset for the Identity (ID) and Hamming weight (HW) leakage models. We considered the training z-MSE loss as the distinguisher. Two different data augmentation techniques were used: shifting and Gaussian noise.

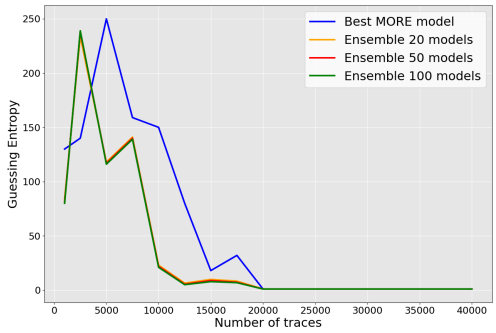


(a) `ASCADf raw`, SR with and without DA for HW.

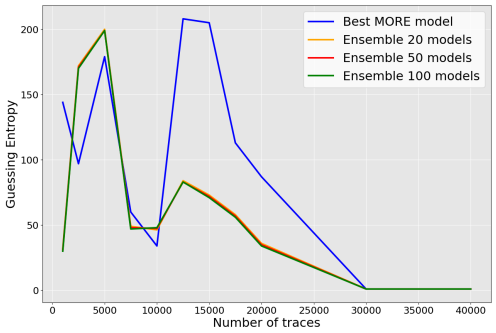


(b) `ASCADf raw`, SR with and without DA for ID.

Figure 5.24: Success rate (SR) for MORE-MLP on the ASCADf raw dataset for the Identity (ID) and Hamming weight (HW) leakage models. We considered the validation z-MSE loss as the distinguisher. Two different data augmentation techniques were used: shifting and Gaussian noise.



(a) `ASCADf raw`, GE for HW.



(b) `ASCADf raw`, GE for ID.

Figure 5.25: Guessing entropy for MORE-MLP on the ASCADf raw dataset for the Identity (ID) and Hamming weight (HW) leakage models. Three different ensemble sizes were used: 20, 50, or 100 models.

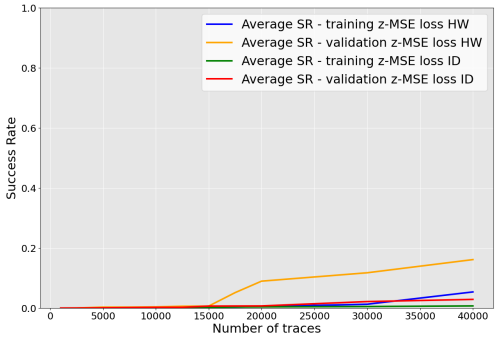


Figure 5.26: Success rate (SR) of the MORE methodology on the ASCADr raw traces varying the number of traces. Both Identity and Hamming weight leakage models are considered. The key distinguisher is the z-MSE loss on training or validation data.

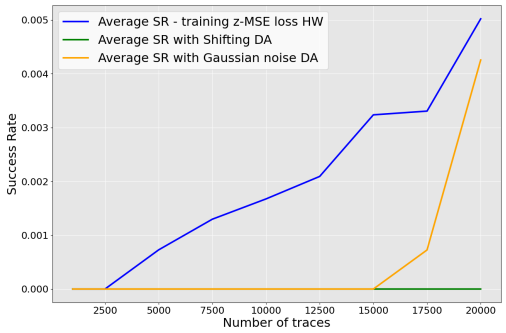
Ensembles

We plotted the guessing entropy and compared the best MORE model against ensembles for the ASCADf raw dataset. The results for the Hamming weight and Identity leakage models are depicted in Figure 5.25. We observed that the ensembles surpassed the performance of the best MORE model by achieving a lower overall guessing entropy. However, it is worth noting that the minimum guessing entropy for both leakage models was achieved as quickly as in the case of the best model, specifically at 20 000 traces for the Hamming weight model and 30 000 traces for the Identity leakage model.

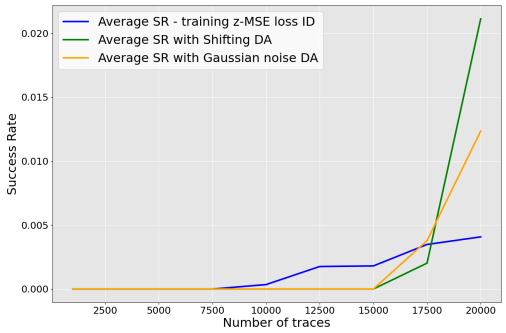
5.5.6 ASCADr raw

Number of traces

In Figure 5.26, we present the success rate of the MORE model on the ASCADr raw dataset, specifically for the Identity and Hamming weight leakage models. The Z-MSE loss is computed for the training and validation sets to evaluate the model's performance. As anticipated, the Hamming weight leakage model shows the highest success rate when the distinguisher is applied to the validation data. On the other hand, the success rate for the Identity leakage model is notably low, even with a dataset consisting of 40 000 traces.

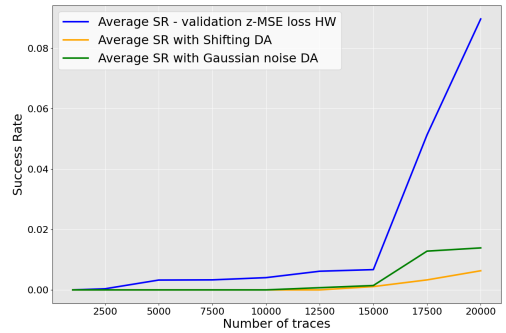


(a) `ASCADr raw`, SR with and without DA for HW.

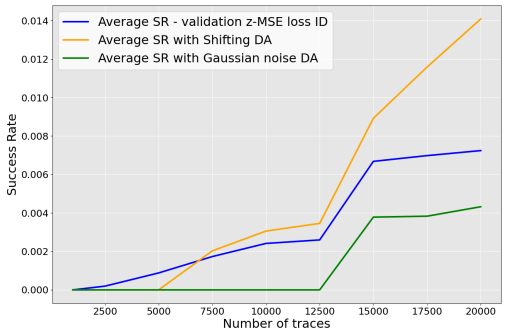


(b) `ASCADr raw`, SR with and without DA for ID.

Figure 5.27: Succes rate (SR) for MORE-MLP on the ASCADr raw dataset for the Identity (ID) and Hamming weight (HW) leakage models. We considered the training z-MSE loss as the distinguisher. Two different data augmentation techniques were used: shifting and Gaussian noise.



(a) `ASCADr raw`, SR with and without DA for HW.



(b) `ASCADr raw`, SR with and without DA for ID.

Figure 5.28: Succes rate (SR) for MORE-MLP on the ASCADr raw dataset for the Identity (ID) and Hamming weight (HW) leakage models. We considered the validation z-MSE loss as the distinguisher. Two different data augmentation techniques were used: shifting and Gaussian noise.

Data augmentation

We conducted a further assessment of two data augmentation methods and examined how they affected the success rate of the MORE technique when applied to the ASCADr raw dataset. The results, illustrated in Figure 5.28, indicate that when using the training Z-MSE loss as the distinguishing factor, the application of shifting data augmentation leads to an enhancement in the success rate (SR) for the Identity leakage model. However, for the Hamming weight leakage model, no such improvement is observed, as demonstrated in Figure 5.27a. It is worth noting that using data augmentation does not result in significant improvements in the SR for this dataset. Similarly, when using the validation loss as the distinguishing factor, data augmentation does not offer significant advantages. Figure 5.28 shows that combining data augmentation with the MORE model leads to a decrease in the success rate (SR) for the Hamming weight leakage model, with SR dropping to around 0.2%. For the Identity leakage model, a slight increase in SR is observed when employing shifting data augmentation.

Ensemble experiments were excluded from this section since the success rate of MORE on ASCADr raw is very low. As previously explained, ensembles that rely on bootstrap bagging cannot improve the performance of the attack if the success rate is small: averaging over mostly wrong results is still wrong.

Table 5.9: Success rate (SR) of MORE on 50-desync ASCADf dataset using the Hamming weight (HW) leakage model and different distinguishers. We considered 20 000, 30 000 or 40 000 input traces.

	20,000	30,000	40,000
Z-MSE training loss	9.2%	11.5%	16.6%
Z-MSE validation loss	6.8%	9.9%	17.5%
Pearson corr training	9.9%	11.5%	16.6%
Pearson corr validation	7.0%	9.0%	16.6%

Table 5.10: Success rate (SR) of MORE on 50-desync ASCADf dataset using the Identity (ID) leakage model and different distinguishers. We considered 20 000, 30 000 or 40 000 input traces.

	20,000	30,000	40,000
Z-MSE training loss	0%	0.7%	1.7%
Z-MSE validation loss	0%	1.0%	2.5%
Pearson corr training	0%	0.3%	1.7%
Pearson corr validation	0.01%	1.4%	2.1%

5.5.7 Desynchronized ASCADf

As mentioned in 4.3.5, the MOR approach results in an unsuccessful attack given a large hyperparameter search space: the success rate is 0% even for 30 000 traces. We conducted the same experiment on desynchronized ASCADf but for the MORE methodology. That is, we randomly chose 500 models from the space introduced in Table 4.4 and varied the number of input traces for both Identity and Hamming weight leakage models.

Table 5.9 shows the success rate of MORE on 50-desynchronized ASCADf when the HW leakage is considered. Unlike in the case of MOR, MORE reaches a higher success rate on the same dataset. We considered 4 different distinguishers: the Z-MSE loss and Pearson correlation on training and validation data. We observe that when the number of traces is smaller than 40 000, using the distinguisher metric on training data results in a slightly higher success rate than when applying it to validation data. This can be due to the size of the validation set and overfitting. CNNs are highly expressive models with a large number of parameters, allowing them to learn complex patterns and structures in the training data. With a small validation set, there is a higher chance that the model may memorize or overly fit the patterns specific to the validation data rather than learning generalizable patterns.

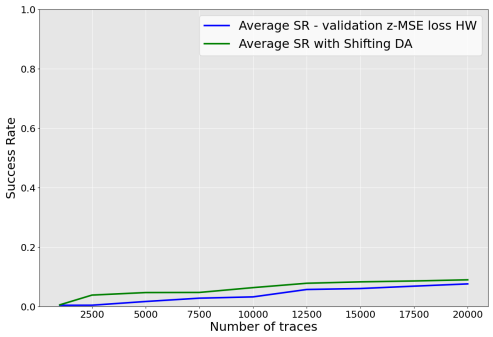
Table 5.10 shows the success rate of MORE on 50-desynchronized ASCADf when the ID leakage is considered. As before, we see that this leakage model requires more input traces than HW in the context of unsupervised deep-learning side-channel analysis. Moreover, here we see better performance when the distinguisher is applied to the validation data rather than the training data.

We have previously seen that shifting data augmentation reduces the success rate of MORE on the ASCADf dataset when the Z-MSE validation loss is considered the distinguisher, see Figure 5.4. We believe that shifting DA results in a higher dissimilarity between the original and augmented traces than in the case of Gaussian noise for the synchronized ASCADf.

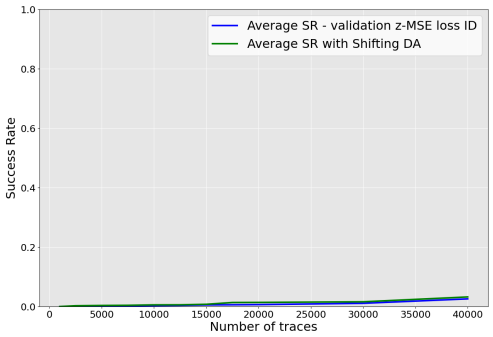
We further conducted data augmentation experiments and observed the influence of shifting DA on 50-desync ASCADf. Figure 5.29 shows the SR before and after data augmentation on desynchronized ASCADf for both Identity and Hamming weight leakage models. We see an improvement in both cases in the SR of the MORE attack, which validates our hypothesis: shifting is similar to desynchronization.

We have seen that in the case of MOR, our proposed MOR-CNN model (Figure A.5) can break ASCADf considering different levels of desynchronization; see Figure 4.35. We used the same model but in the context of MORE: considering the Z-MSE loss as the distinguisher. Figure 5.30 compares the two approaches on the same model. As we see in Figure 5.32b, only 2 epochs are enough to distinguish correctly between the key candidates for MORE, compared to 15 epochs in the case of MOR.

Moreover, from the hyperparameter space defined in Table 4.4, we chose the best model according to the smallest Z-MSE loss of the validation dataset averaged over all key candidates. We chose this objective function rather than the maximum Pearson correlation summed over all key candidates

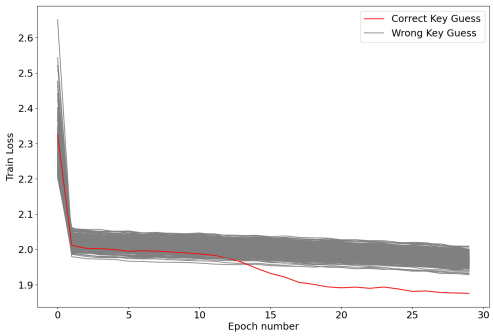


(a) `50-desync ASCADf`, SR with and without DA for HW.

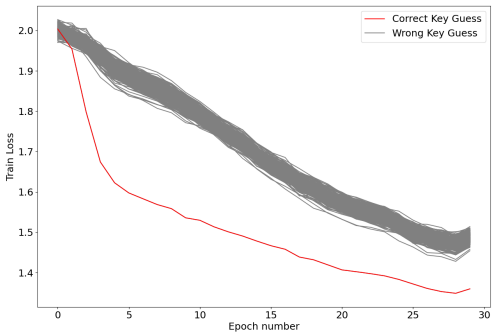


(b) `50-desync ASCADf`, SR with and without DA for ID.

Figure 5.29: Success rate (SR) for MORE-CNN on the 50-desync ASCADf dataset for the Identity (ID) and Hamming weight (HW) leakage models. We employed the shifting data augmentation technique.



(a) `50-desynchronized ASCADf`, proposed MOR-CNN MSE loss for all key candidates using 30 000 traces.



(b) `50-desynchronized ASCADf`, proposed MOR-CNN on Z-MSE loss for all key candidates using 30 000 traces.

Figure 5.30: The proposed MOR-CNN model on both MOR and MORE methodologies. The Hamming weight (HW) leakage model is considered in both cases.

since we experimentally noticed that it results in an overall smaller guessing entropy. We evaluate its performance by varying the maximum desynchronization level on ASCADf. Figure 5.31 shows that the best model can distinguish the correct key within 15 epochs even when the maximum level of desynchronization is 250. Furthermore, we see that in the case of 50-desync ASCADf, the key is correctly distinguished even after the first epoch. As expected, increasing the desynchronization level results in increasing the number of training epochs.

We further varied the number of traces and evaluated the performance of the best MORE-CNN model against 50-desynchronized ASCADf traces. Compared to the proposed MOR-CNN model that needs at least 10 000 input traces to break the secret key (Figure 4.34), the MORE approach can correctly distinguish between the keys even when 2 500 traces are used as input. Figure 5.33 shows these results.

5.5.8 Fine-tuning

Fine-tuning is a technique used to improve the performance of a deep learning model on a specific task or dataset. It involves adjusting the parameters of a pre-trained model to better fit the data for the new task. We previously noticed that MORE CNNs models are easily susceptible to overfitting. Overfitting can result in poor generalization (the approach would not work on different datasets) or an unsuccessful attack. The latter can happen if the CNN is too large (i.e., many layers and neurons) and can quickly learn even for incorrect key candidates. Figure 5.33c is an example of the latter. The model could learn at the same rate for all key candidates, becoming harder to retrieve the correct one. On the other hand, Figure 5.32a shows MOR and the model that barely learns for incorrect key candidates: the loss slightly decreases across 40 epochs, while for the correct key, we see a more inclined slope.

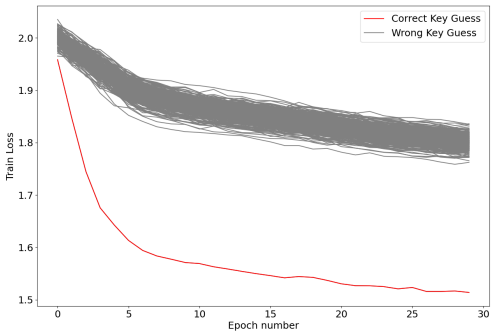
To that extent, we employed fine-tuning for our CNN MORE models and investigated the effect on the success rate of the attacks. We hypothesize that fine-tuning prevents overfitting. Our experiments trained the models for 10 epochs, and then we froze the first layers of our models (in all experiments, we chose the first 100 layers) that take care of the feature selection of the input data points. By freezing the first layers, you prevent them from being updated during training, which can help prevent overfitting by reducing the number of trainable parameters in the model. This technique is often used in transfer learning, where a pre-trained model is fine-tuned on a new task.

We run the fine-tuning experiments for ASCADf with 50-desynchronization for 30 000 input data points. The SR did not improve when we distinguished based on training data loss values: it remained at 11%. However, the SR for Z-MSE validation increased from 9.9% to 11.9%. Better performance on training data than validation data can indicate overfitting. Thus, we believe that fine-tuning can help mitigate the overfitting issue for CNN models.

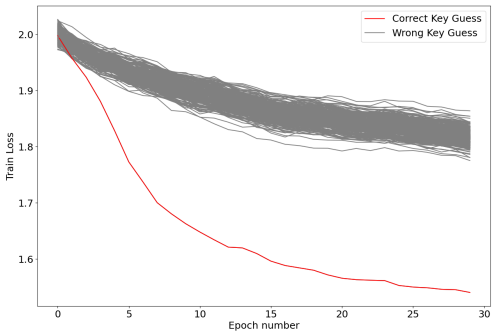
5.6 Discussion

In this chapter, we proposed a new loss function that can be used in multi-output regression: the normalized Z-score MSE (Z-MSE). We further showed that distinguishing based on validation rather than training data results in a higher success rate. Moreover, we propose an objective function that can be used for hyperparameter tuning based on the Pearson correlation. For each key candidate, we calculate the correlation between the actual and predicted labels and return the key that shows the largest correlation among all traces. This approach is called MORE (multi-output regression enhanced).

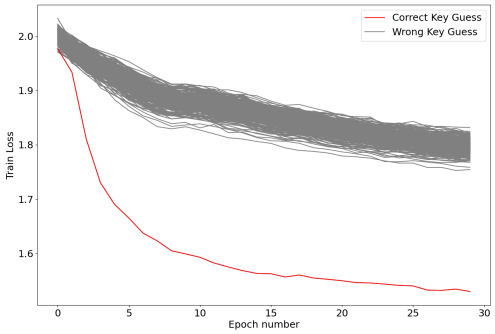
Our experiments showed that using the Z-MSE loss values on validation data as distinguisher results in above 90% success rate (SR) for ASCADf and the Hamming weight leakage model for only 20 000. Furthermore, this approach demonstrates a larger SR than the MSE loss function (as done in state-of-the-art work [17]) when the SR is 27% for the same settings. This is the case since the Z-MSE is less sensitive to outliers than MSE. In the context of unsupervised side-channel analysis, we aim to have the same (or very similar) loss value for all key candidates but not for the correct one. This is because if the correct key is, for example, 3, the key guess 4 is as wrong as the key guess 240. From the adversary's perspective, outliers refer to the attacker's most probable key guess. Consequently,



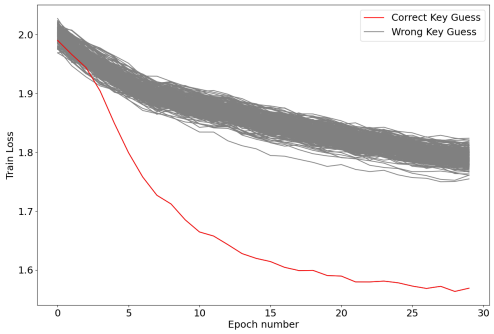
(a) `50-desynchronized ASCADf`, best MORE-CNN Z-MSE loss for all key candidates.



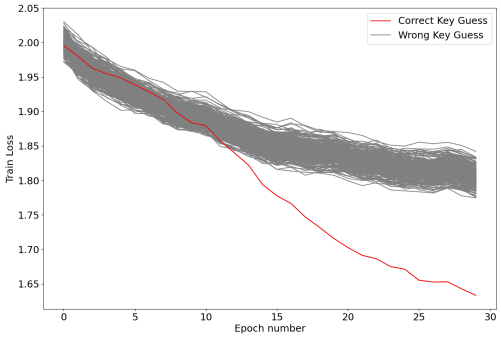
(b) `100-desynchronized ASCADf`, best MORE-CNN Z-MSE loss for all key candidates.



(c) `150-desynchronized ASCADf`, best MORE-CNN Z-MSE loss for all key candidates.

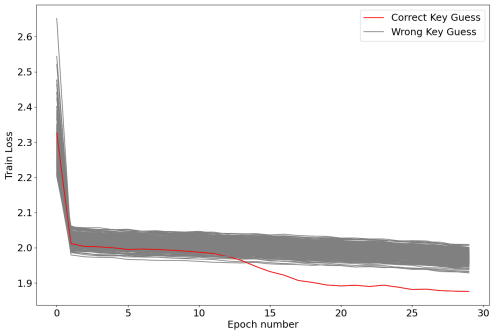


(d) `200-desynchronized ASCADf`, best MORE-CNN Z-MSE loss for all key candidates.

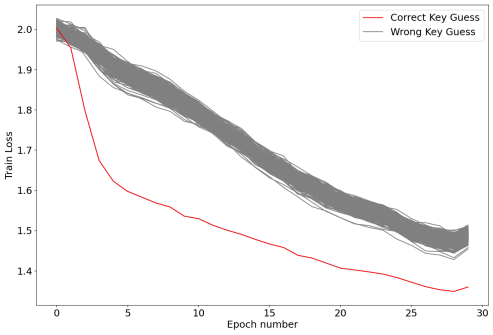


(e) `250-desynchronized ASCADf`, best MORE-CNN Z-MSE loss for all key candidates.

Figure 5.31: The loss function values for the best MORE-CNN model varying the desynchronization level. The number of traces is fixed at 30 000.

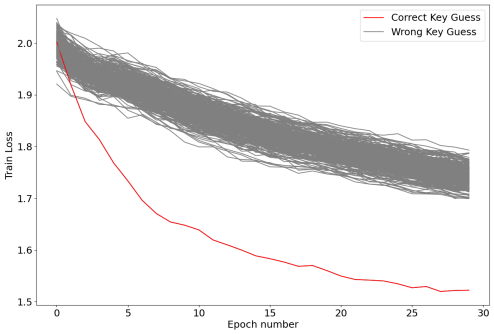


(a) `50-desynchronized ASCADf`, proposed MOR-CNN MSE loss for all key candidates using 30 000 traces.

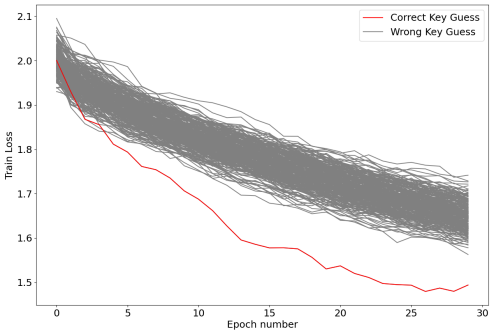


(b) `50-desynchronized ASCADf`, proposed MOR-CNN on Z-MSE loss for all key candidates using 30 000 traces.

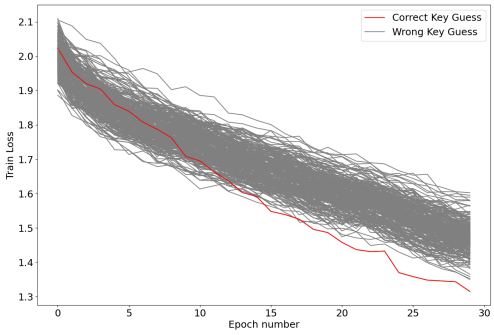
Figure 5.32: The proposed MOR-CNN model on both MOR and MORE methodologies.



(a) `50-desynchronized ASCADf`, best MORE-CNN Z-MSE loss for all key candidates using 10 000 traces.



(b) `50-desynchronized ASCADf`, best MORE-CNN on Z-MSE loss for all key candidates using 5 000 traces.



(c) `50-desynchronized ASCADf`, best MORE-CNN on Z-MSE loss for all key candidates using 2 500 traces.

Figure 5.33: The best MORE-CNN model on 50-desynchronized ASCADf varying the number of traces.

we aim to design a loss function that assigns an exceptionally low loss value (outlier) to the correct key compared to all other possible keys. These outliers can significantly influence the MSE. To ensure equal penalization of all incorrect key guesses, we favor using normalized MSE.

We also obtained a high success rate in the case of the ASCADr dataset, but here, we need 40 000 input traces to reach SR above 90% for the HW leakage model. We believe we need more traces than for ASCADf because ASCADr traces have 1 400 points of interest, while ASCADf traces have 700. This difference can be explained by the fact that there is a need for more input data points if they have many features. This is because the number of parameters in the model increases with the number of features in the input vector. As a result, the model may overfit the training data and perform poorly on new data.

Our experiments involved 10 different loss functions. We compared MSE, MAE, Huber, LogCosh, Tukey's Biweight, and their normalized Z-score counterparts. Even though Z-MSE is considerably the best in terms of the attack's success rate, it is worth investigating the other functions as well. MSE penalizes more significant errors due to the squaring operation. MAE does not involve squaring the errors, so it treats all errors equally and is less sensitive to outliers. Huber loss behaves like MSE for small errors and like MAE for significant errors by introducing a threshold parameter δ to differentiate between the two behaviors. If the absolute error is below δ, it behaves like MSE (quadratic loss), and if it exceeds δ, it behaves like MAE (linear loss). LogCosh behaves similarly to the Huber loss, as it is like the mean squared error (MSE) for small errors and the mean absolute error (MAE) for larger errors. However, unlike the Huber loss, the transition from the quadratic to linear behavior is continuous rather than abrupt. Tukey's biweight loss assigns lower weights to larger residuals (outliers) than smaller residuals. It has a quadratic behavior for residuals within a specific range $|y_{true} - y_{pred}| \leq c$, where it down weights outliers significantly. Beyond this range, the loss function transitions to a linear behavior, assigning a weight equal to half the magnitude of the residual. We noticed that MAE performs the worst overall. The difference in performance between the MAE and alternative loss functions such as Huber, LogCosh, and Tukey's Biweight in your multi-output regression task can be attributed to several factors. One key consideration is how the difference between predicted and actual labels affects the loss values. While MAE gives equal weight to all errors, MSE squares the difference, making the gap between the correct and other predictions more prominent when the difference exceeds 1. However, the loss value decreases after squaring for values that are close together (difference smaller than 1). This hypothesis aligns with your observations that MSE, Huber, LogCosh, and Tukey exhibit similar behavior due to their increased sensitivity to outliers. We also noticed a good performance when using Tukey or its normalized Z-score counterpart.

However, we are still interested in why the normalized Z-score MSE works considerably better than the other normalized loss function. To that extent, we plotted the loss values for all key candidates during the last epoch of a randomly chosen model. We show the results in Figure 5.34. For all loss functions, the correct key is correctly distinguished (the correct key is 224 in this case). However, the largest gap is observed for Z-MSE, where the loss for the correct key is around 1.6, while for the other key candidates 1.8. For the other loss functions, the gap is considerably smaller. By scaling these values, as done in Figure 5.35, we observe that the last statement is true: Z-MSE creates the largest gap between the correct key and the others. We believe this is why Z-MSE works the best among all loss functions.

On the other hand, one can further wonder why MSE does not perform considerably better than the other loss functions: Huber, LogCosh, and Tukey's Biweight. We hypothesize this is because the advantages of MSE that we observe for its normalized Z-score counterpart are no longer important when the data is not scaled (or normalized): the squared differences between the true and predicted labels follow a distribution with high variance, becoming challenging to distinguish correctly between the keys. We show this in Figure 5.36. We see how for Z-MSE, the correct key has the smallest loss value (corresponds to value 224 on the horizontal axis), while for MSE, the loss values vary, and the correct key is not distinguished correctly.

We also saw that for ASCADr raw traces, the MORE approach performs considerably worse even when input traces are as many as 40 000: we believe two reasons can explain this. Firstly, we hypothesize that the models employed in our experiments are too small to learn for these datasets. In general, if the input data points have a lot of features, the relationships between those features are relatively simple

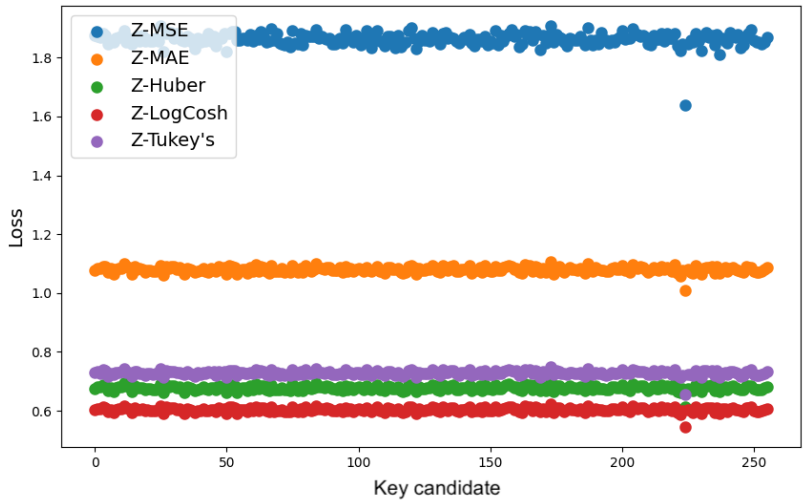


Figure 5.34: The normalized Z-score loss functions and their values for each key candidate.

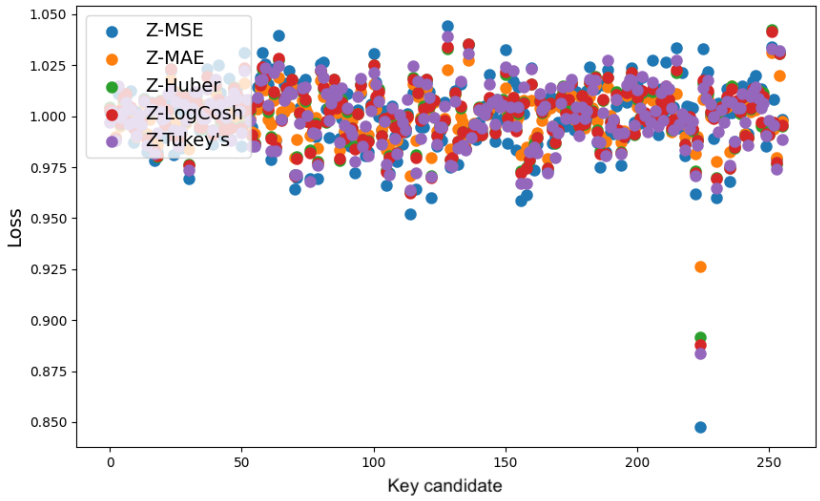


Figure 5.35: The normalized Z-score loss functions and their values for each key candidate. The loss values are scaled (divided by their mean).

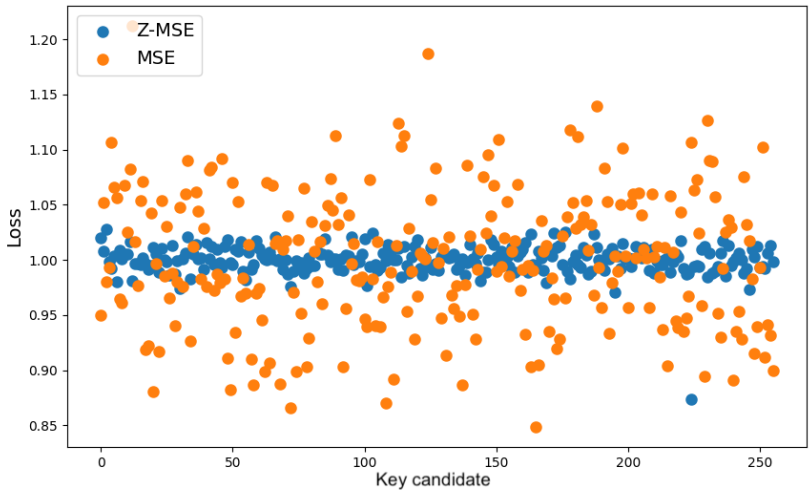


Figure 5.36: The MSE and normalized Z-score MSE values for each key candidate. The loss values are scaled (divided by their mean).

and can be captured by a smaller model. Therefore, a larger model may not be necessary. A smaller model can provide sufficient capacity to learn the relevant patterns in such cases. On the other hand, if the input data has a high-dimensional feature space with intricate interactions and dependencies, a larger deep-learning model may be more suitable. A larger model can offer increased capacity to capture the complexities in the data, potentially leading to improved performance. We think the second scenario holds for ASCADr raw traces. The second hypothesis that can explain the low SR of MORE on the raw dataset is that the models are large enough (enough neurons or layers) to learn the features from this dataset, but 40 000 traces are not enough (same case as when comparing ASCADf and ASCADr). We think the second hypothesis is more likely in this context since we can select the best model from the space for ASCADr raw, and the correct key is retrieved with minimum GE within 30 epochs.

Our experiments showed a significant difference in the SR for the same amount of traces (20 000) between ID and HW leakage models: while we obtained around 90% SR for HW, the SR for ID was only approximately 20%. However, when the number of input traces is more significant (50 000), the SR increases to approximately 90% for ID. We believe this is because of the HW leakage model's class imbalance. As discussed in [54], HW shows less misclassification since it has only nine classes compared to 256 for Identity. It is thus more robust to noise.

As in Chapter 4, we evaluated the performance of MORE with two data augmentation techniques: Gaussian noise and shifting. As before, we saw that DA improves the SR for most datasets for at least one leakage model. Overall, Gaussian noise shows better performance than shifting. As Chapter 4 mentions, shifting resembles desynchronization, and MLPs are not powerful enough to break desynchronized traces. However, in the case of raw traces, shifting performs slightly better. We hypothesize that this is because the raw traces have many features: 10 000 for ASCADf raw and 25 000 for ASCADr raw; thus, shifting with a small value of 5 does not change the traces significantly. We also evaluated DA on the best MORE model chosen according to the highest objective function value. We plotted the average guessing entropy of the best model with and without DA. We noticed that at least one DA technique results in a lower GE (usually Gaussian noise, even though shifting DA also showed improvement) for all datasets. It is worth mentioning that DA does not improve some datasets or leakage models.

On the contrary, the SR decreases significantly. We believe this is because data augmentation reduces overfitting, which can quickly happen in the MORE context. As we showed for MOR, the best results were obtained for the ASCADf dataset since this approach was developed to suit it [17]. Similarly, although we try to minimize overfitting by using a validation set, MORE is still susceptible to overfitting.

Moreover, we further used MORE and ensembles. We noticed that when the SR is high enough (more than 10%), we can use ensembles to reduce the number of traces needed to reach the minimum guessing entropy or reduce the overall GE when the number of traces is not high enough to break the secret key.

Lastly, we evaluated the performance of MORE on CNNs on the desynchronized ASCADf dataset. Compared to MOR, when we obtained an SR of 0%, MORE can reach 17% SR on 50-desynchronized ASCADf when enough traces are provided: 40 000 for HW leakage. For ID, with 40 000 traces, we obtained an SR of only 2.5%. However, the best model chosen according to the highest Pearson correlation among all validation traces shows excellent performance: the correct key is retrieved within the first epoch for HW and 30 000 input traces. Furthermore, even for a significant desynchronization level of 250, we retrieve the correct secret key with 1 GE after 13 epochs. Compared to MOR, MORE on desynchronized ASCADf requires fewer traces: even with 2 500 we can retrieve the correct secret key within 30 epochs, while MOR requires at least 10 000 input traces.

Chapter 6

Conclusion

Profiled side-channel attacks (SCA) are crucial for assessing cryptographic algorithm security, but they require an almost identical reference device to the target, limiting their practicality. Non-profiled attacks offer a realistic alternative when the attacker lacks access to a profiling device or assumptions about the target. These complex attacks require innovative techniques to exploit side-channel information without prior knowledge. In this work, we focused on non-profiled deep learning-based SCA. This chapter summarizes our contributions, outlines the main findings systematically concerning the research questions introduced in Chapter 3, and proposes several future work directions.

6.1 Contributions

- This works provides a thorough evaluation of the existing unsupervised deep-learning-based side-channel analysis approaches: differential deep learning analysis [67], and multi-output regression [17]. We show that using a validation set on which the key distinguishing is done rather than the training set improves the overall success rate for all datasets. We further showed that employing ensembles for multi-output regression and data augmentation can also raise the success rate and lower the overall guessing entropy.

- We evaluate numerous loss functions in the context of multi-output regression side-channel analysis: MSE, MAE, Huber, LogCosh, Tukey's Biweight, and their normalized Z-score counterparts. We propose a novel approach that improves the state-of-the-art unsupervised deep-learning-based side-channel attacks that uses the normalized Z-score MSE (Z-MSE) loss function.

- We further propose two methods by which the key distinguishing can be done: either by considering ranking the keys based on the smallest Z-MSE loss or based on the highest Pearson correlation between the actual and predicted labels.

- We propose that the objective function for hyperparameter tuning can be based on either the smallest Z-MSE loss averaged for all key candidates or the highest Pearson correlation averaged over all key candidates.

- We show that our novel approach can easily break traces protected by countermeasures. We varied the maximum level of desynchronization for ASCADf traces and showed that even for 250-desynchronized traces, the attack is successful within 15 epochs.

- We can further improve this novel approach by employing ensembles: we are able to lower the number of traces needed to break the key for almost all datasets. Furthermore, data augmentation reduces the average guessing entropy and sometimes results in a higher success rate.

6.2 Research Questions

RQ1 How can we review and quantify the attack performance of the available deep learning-based non-profiled side-channel analysis methodologies?

Chapter 4 compares two non-profiled deep learning-based side-channel analysis approaches: differential deep learning analysis (DDLA) [67] and unsupervised multi-output regression (MOR) [17] on four datasets: ASCADf, ASCADr, AES_HD, and AES_RD. The first drawback of the two methodologies is that the hyperparameter space for the neural network was either non-existent (DDLA) or minimal (MOR). We used ample hyperparameter space and randomly chose models from there. We showed that DDLA is less sensitive to hyperparameters than MOR: it can reach almost 100% success rate (SR) for all datasets for the Hamming weight (HW) leakage model except for AES_HD, where the approach was unsuccessful. On the other hand, in the case of MOR, the success rate is not larger than 30% for any of the datasets.

Furthermore, unlike the original DDLA, where the authors only considered training data, our experiments used validation data to distinguish between the key candidates. We showed that it could significantly improve the performance of the attack, especially for the AES_RD dataset, where overfitting easily happens during training. The SR did not exceed 40% for this dataset when distinguishing is based on training data. However, if we use a validation set, the SR can go as high as 100%. We observed a similar result for MOR: for AES_HD and HW leakage, 0% SR is obtained when using training data for distinguishing, while 10% is observed when using validation data.

Finally, we performed experiments for both methodologies, data augmentation, and ensembles. The study used two data augmentation techniques (shifting and Gaussian noise) on the input traces. Gaussian noise showed better performance in most cases, reducing overfitting. However, shifting data reduced the success rate due to increased dissimilarity between the original and augmented traces. Furthermore, deep learning ensembles improved MOR performance with a small hyperparameter space. Still, not practical for DDLA due to its structure: a single DDLA model consists of 256 neural networks, making it computationally demanding to train an ensemble with 20 models, requiring a total of 5,120 neural networks.

RQ2 Can we derive a new methodology for deep learning-based non-profiled side-channel analysis with good results?

Chapter 5 presents MORE (multi-output regression enhanced), a novel approach to unsupervised deep learning-based side-channel analysis that uses the normalized Z-score MSE (Z-MSE) as a loss function, Pearson correlation as a key distinguisher and proposes a new method that can be used for hyperparameter tuning in a non-profiled context. We evaluated MORE against six datasets: ASCADf, ASCADr, AES_HD, AES_HD, ASCADf raw, and ASCADr raw.

We started answering this question by researching different loss functions and their impact on the success rate of the attacks. We evaluated MSE, MAE, Huber, LogCosh, Tukey's Biweight, and their normalized Z-score counterparts. We saw that MAE performs the worst because it penalizes all errors similarly, while MSE, Huber, LogCosh, and Tukey's have quadratic behavior penalizing more significant errors. Moreover, we reasoned that Z-MSE performs best than the other normalized functions because it creates the most significant gap between the correct key candidate and the others. Finally, we hypothesize that there is no significant difference between MSE and the others (as in the case of their normalized counterparts) because the advantages of MSE (penalizing more significant errors significantly more than more minor errors) are no longer practical when the data is not scaled and shows high variance.

By utilizing Z-MSE loss values on validation data for distinguishing, MORE achieves a remarkable success rate of above 90% for the ASCADf and Hamming weight (HW) leakage model with just 20 000 traces where the models were randomly drawn from an ample hyperparameter space. The superiority of Z-MSE over the traditional MSE loss function was evident, as the state-of-the-art approach only yielded a 27% success rate under similar conditions. This discrepancy can be explained by Z-MSE's sensitivity to outliers: which is more robust than MSE, which in the context of unsupervised side-channel

analysis, is desired.

We further show that ranking the key candidate from the most likely to the least likely can also be done based on the Pearson correlation between accurate labels and predicted labels, improving the attack's success rate for all loss functions. For example, the SR for the state-of-the-art is 27% when we key rank based on the MSE loss and 51% when we key rank based on the Pearson correlation and still use the MSE loss function. The best result is still obtained for Z-MSE. Because of these results, we propose to perform hyperparameter tuning based on the highest overall Pearson correlation between actual and predicted labels averaged over all key candidates or by the lowest overall Z-MSE loss averaged over all key candidates.

Additionally, we showed that MORE works in the case of desynchronized traces. We randomly selected CNN models from an ample hyperparameter space. Compared to the state-of-the-art work, MOR, where the success rate was 0% for both HW and ID leakage models, MORE found some models for which the attack was successful. We further showed that selecting the best model based on the lowest overall Z-MSE loss for all key candidates can easily break even 250-desynchronized ASCADf traces: 12 epochs are enough to distinguish correctly between the keys.

Furthermore, by attacking different datasets, we experimentally showed the demand for more input data points that arise from the complexity introduced by an increased number of points of interest in multi-output regression. The experiments revealed that the MORE approach performed considerably worse with ASCADf and ASCADr raw traces, even with 40,000 input traces.

Moreover, by using and comparing two different leakage models, Hamming weight and Identity (ID), we showed that ID requires more input traces that can be explained by the class imbalance of HW, which makes it more robust to noise and gives an overall lower misclassification rate.

Finally, we showed that the MORE attack can be improved by employing ensembles and data augmentation. For data augmentation, Gaussian noise is overall better overall datasets; however, both techniques help reduce the average Guessing entropy.

6.3 Limitations and Future Work

In this section, we mention the limitations of the proposed multi-output regression enhanced (MORE) approach for side-channel analysis and highlight potential avenues for future research.

Comparison with PLDL Approach

Our research compares the MORE approach against the state-of-the-art MOR [17]. However, Wu *et al.* recently introduced the non-profiled plaintext labeling deep learning (PLDL) SCA approach that shows great potential, outperforming MOR [74]. The PLDL approach labels the traces with their corresponding plaintexts rather than with the leakage model. It determines the correct key based on the highest correlation between the predicted labels and plaintext distribution given a key candidate. One direction for future work is to compare MORE with PLDL. We can gain insights into their relative strengths and weaknesses by conducting a comparative analysis between these approaches. Furthermore, exploring the potential integration of MORE and PLDL techniques could lead to developing a more powerful side-channel analysis technique that combines the benefits of both approaches.

Exploring Multi-Output Classification Techniques

While the current work focuses on multi-output regression, exploring multi-output classification (MOC) techniques could also be valuable. MOC and MOR were introduced together by Do *et al.* in 2022 [18]. MOC relies upon Timon's approach of employing the LSB/MSB leakage models, although it differs by using a single network instead of 256. While MOC has advantages, we emphasized exploring MOR due to its demonstrated potential of using the Hamming weight and Identity leakage model in a non-profiled

method. Nevertheless, we recognize the importance of conducting research regarding MOC. Despite its simplified approach, investigating whether similar results can be achieved using MOC techniques can provide valuable insights and help determine the situations in which MOC may outperform MOR. This evaluation could involve experimentation with a diverse hyperparameter space, different datasets, and measuring the success rate of MOC-based side-channel analysis. Additionally, ensemble methods or data augmentation techniques could be explored to enhance the performance of MOC-based approaches.

Exploration of Alternative Loss Functions

Our experiments employed ten loss functions, including mean squared error (MSE), mean absolute error (MAE), Huber, LogCosh, Tukey's Biweight, and their normalized Z-score counterparts. Another area of future research is the exploration of alternative loss functions for the MORE approach. While MORE uses the normalized Z-score mean squared error (Z-MSE) as the loss function, it is worth investigating whether alternative loss functions offer improved performance. Exploring different loss functions could help identify the most effective one for optimizing the performance of the MORE approach.

Investigating Ensemble Techniques

Furthermore, investigating different ensemble techniques could contribute to enhancing the MORE approach. The current research focuses on the bootstrap aggregating (bagging) approach, where the models' results are averaged. However, alternative ensemble techniques, such as majority voting or stacking, could be explored. Majority voting involves taking the option that receives the highest number of agreements on the models' predictions. Stacking involves employing another deep-learning model to aggregate the results of the individual models. By exploring different ensemble techniques, we can improve the robustness and accuracy of the MORE approach.

In conclusion, while the MORE approach presents a novel and effective technique for side-channel analysis, future research has several avenues to further enhance its performance and capabilities. By comparing with alternative approaches and exploring different techniques, loss functions, and ensemble methods, we can advance the field of side-channel analysis and develop even more powerful methodologies for analyzing and mitigating side-channel vulnerabilities.

Bibliography

[1] Claudine Badue et al. “Self-driving cars: A survey”. In: *Expert Systems with Applications* 165 (2021), p. 113816.

[2] Vasileios Belagiannis et al. “Robust Optimization for Deep Regression”. In: (May 2015).

[3] Ryad Benadjila et al. “Deep learning for side-channel analysis and introduction to ASCAD database”. In: *Journal of Cryptographic Engineering* 10.2 (2020), pp. 163–188.

[4] Ryad Benadjila et al. “Deep learning for side-channel analysis and introduction to ASCAD database”. In: *J. Cryptographic Engineering* 10.2 (2020), pp. 163–188. doi: 10.1007/s13389-019-00220-8. url: https://doi.org/10.1007/s13389-019-00220-8.

[5] Y. Bengio and Yann Lecun. “Convolutional Networks for Images, Speech, and Time-Series”. In: (Nov. 1997).

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. isbn: 0387310738.

[7] Michael Black and Anand Rangarajan. “On the Unification Line Processes, Outlier Rejection, and Robust Statistics with Applications in Early Vision”. In: *International Journal of Computer Vision* 19 (July 1996), pp. 57–91. doi: 10.1007/BF00131148.

[8] Xavier Bonnetain, María Naya-Plasencia, and André Schrottenloher. “Quantum Security Analysis of AES”. In: *IACR Transactions on Symmetric Cryptology* 2019, Issue 2 (2019), pp. 55–93. doi: 10.13154/tosc.v2019.i2.55-93. url: https://tosc.iacr.org/index.php/ToSC/article/view/8314.

[9] Hanen Borchani et al. “A survey on multi□output regression”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5 (2015).

[10] Leo Breiman. “Bagging Predictors”. In: *Machine Learning* 24.2 (1996), pp. 123–140.

[11] Eric Brier, Christophe Clavier, and Francis Olivier. “Correlation Power Analysis with a Leakage Model”. In: vol. 3156. Aug. 2004, pp. 16–29. isbn: 978-3-540-22666-6. doi: 10.1007/978-3-540-28632-5_2.

[12] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. “Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures”. In: Aug. 2017, pp. 45–68. isbn: 978-3-319-66786-7. doi: 10.1007/978-3-319-66787-4_3.

[13] Nuno Carneiro, Gonçalo Figueira, and Miguel Costa. “A data mining based system for credit-card fraud detection in e-tail”. In: *Decision Support Systems* 95 (2017), pp. 91–101.

[14] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. “Template Attacks”. In: *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*. Vol. 2523. Lecture Notes in Computer Science. Springer, 2002, pp. 13–28. doi: 10.1007/3-540-36400-5_3.

[15] Jean-Sébastien Coron and Ilya Kizhvatov. “An Efficient Method for Random Delay Generation in Embedded Software”. In: *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*. 2009, pp. 156–170. doi: 10.1007/978-3-642-04138-9_12. url: https://doi.org/10.1007/978-3-642-04138-9_12.

[16] CHRISTIAN W. DAWSON and ROBERT WILBY. “An artificial neural network approach to rainfall-runoff modelling”. In: *Hydrological Sciences Journal* 43.1 (1998), pp. 47–66. doi: 10.1080/02626669809492102. eprint: https://doi.org/10.1080/02626669809492102. url: https://doi.org/10.1080/02626669809492102.

[17] Ngoc-Tuan Do, Van-Phuc Hoang, and Van Sang Doan. “A novel non-profiled side channel attack based on multi-output regression neural network”. In: *Journal of Cryptographic Engineering* (2023), pp. 1–13.

[18] Ngoc-Tuan Do et al. “MO-DLSCA: Deep Learning Based Non-profiled Side Channel Analysis Using Multi-output Neural Networks”. In: *2022 International Conference on Advanced Technologies for Communications (ATC)*. 2022, pp. 245–250. doi: 10.1109/ATC55345.2022.9943024.

[19] Morris Dworkin et al. *Advanced Encryption Standard (AES)*. en. 2001-11-26 2001. doi: https://doi.org/10.6028/NIST.FIPS.197.

[20] Meherwar Fatima, Maruf Pasha, et al. “Survey of machine learning algorithms for disease diagnostic”. In: *Journal of Intelligent Learning Systems and Applications* 9.01 (2017), p. 1.

[21] Yoav Freund, Robert E Schapire, et al. “Experiments with a new boosting algorithm”. In: *icml*. Vol. 96. Citeseer. 1996, pp. 148–156.

[22] Nadia Ghamrawi and Andrew McCallum. “Collective multi-label classification”. In: *International Conference on Information and Knowledge Management*. 2005.

[23] Benedikt Gierlichs et al. “Mutual Information Analysis”. In: *Proceeding Sof the 10th International Workshop on Cryptographic Hardware and Embedded Systems*. CHES ’08. Washington, DC, USA: Springer-Verlag, 2008, pp. 426–442. isbn: 9783540850526. doi: 10.1007/978-3-540-85053-3_27. url: https://doi.org/10.1007/978-3-540-85053-3_27.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[25] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *arXiv preprint arXiv:1512.03385* (2015).

[26] Van-Phuc Hoang, Ngoc-Tuan Do, et al. “Efficient Non-profiled Side Channel Attack Using Multi-output Classification Neural Network”. In: *IEEE Embedded Systems Letters* (2022).

[27] Gabriel Hospodar et al. “Machine learning in side-channel analysis: A first study”. In: *J. Cryptographic Engineering* 1 (Dec. 2011), pp. 293–302. doi: 10.1007/s13389-011-0023-x.

[28] Kevin Jarrett et al. “What is the best multi-stage architecture for object recognition?” In: *2009 IEEE 12th International Conference on Computer Vision*. 2009, pp. 2146–2153. doi: 10.1109/ICCV.2009.5459469.

[29] *Keras API*. url: https://keras.io/.

[30] Maikel Kerkhof et al. “Focus is key to success: a focal loss function for deep learning-based side-channel analysis”. In: *Constructive Side-Channel Analysis and Secure Design: 13th International Workshop, COSADE 2022, Leuven, Belgium, April 11-12, 2022, Proceedings*. Springer. 2022, pp. 29–48.

[31] Jaehun Kim et al. “Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2019), pp. 148–179.

[32] Paul C. Kocher. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. In: *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO ’96. Berlin, Heidelberg: Springer-Verlag, 1996, pp. 104–113. isbn: 3540615121.

[33] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. “Differential Power Analysis”. In: *Advances in Cryptology - CRYPTO ’99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 388–397. doi: 10.1007/3-540-48405-1_25.

[34] Sajja Tulasi Krishna and Hemantha Kumar Kalluri. “Deep learning and transfer learning approaches for image classification”. In: *International Journal of Recent Technology and Engineering (IJRTE)* 7.5S4 (2019), pp. 427–432.

[35] Takaya Kubota et al. “Deep learning side-channel attack against hardware implementations of AES”. In: *Microprocessors and Microsystems* 87 (2021), p. 103383.

[36] Donggeun Kwon, HeeSeok Kim, and Seokhie Hong. “Improving non-profiled side-channel attacks using autoencoder based preprocessing”. In: *Cryptology ePrint Archive* (2020).

[37] Donggeun Kwon, Heeseok Kim, and Seokhie Hong. “Non-profiled deep learning-based side-channel preprocessing with autoencoders”. In: *IEEE Access* 9 (2021), pp. 57692–57703.

[38] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. “Power analysis attack: An approach based on machine learning”. In: *Int. J. of Applied Cryptography* 3 (Jan. 2014), ied Cryptography. doi: 10.1504/IJACT.2014.062722.

[39] Liran Lerman et al. “A Machine Learning Approach Against a Masked AES”. In: Nov. 2013, pp. 61–75. isbn: 978-3-319-08301-8. doi: 10.1007/978-3-319-08302-5_5.

[40] Owen Lo, William Buchanan, and Douglas Carson. “Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA)”. In: *Journal of Cyber Security Technology* 1 (Sept. 2016), pp. 1–20. doi: 10.1080/23742917.2016.1231523.

[41] Xiangjun Lu, Chi Zhang, and Dawu Gu. “Attention - Based Non-Profiled Side-Channel Attack”. In: *2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. 2021, pp. 1–6. doi: 10.1109/AsianHOST53231.2021.9699481.

[42] Xiangjun Lu et al. “Pay Attention to Raw Traces: A Deep Learning Architecture for End-to-End Profiling Attacks”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021.3 (July 2021), pp. 235–274. doi: 10.46586/tches.v2021.i3.235-274. url: https://tches.iacr.org/index.php/TCHES/article/view/8974.

[43] Lukas Malina et al. “On perspective of security and privacy-preserving solutions in the internet of things”. In: *Computer Networks* 102 (2016), pp. 83–95. issn: 1389-1286. doi: https://doi.org/10.1016/j.comnet.2016.03.011. url: https://www.sciencedirect.com/science/article/pii/S1389128616300779.

[44] James L. Massey. “Guessing and entropy”. In: *Proceedings of 1994 IEEE International Symposium on Information Theory* (1994), pp. 204–.

[45] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[46] Eitan Menahem et al. “Improving malware detection by applying multi-inducer ensemble”. In: *Computational Statistics Data Analysis* 53.4 (2009), pp. 1483–1494. issn: 0167-9473. doi: https://doi.org/10.1016/j.csda.2008.10.015. url: https://www.sciencedirect.com/science/article/pii/S0167947308004763.

[47] Tom M Mitchell. *Machine learning*. Vol. 1. 9. McGraw-hill New York, 1997.

[48] Thorben Moos, Felix Wegener, and Amir Moradi. “DL-LA: Deep Learning Leakage Assessment: A modern roadmap for SCA evaluations”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021.3 (July 2021), pp. 552–598. doi: 10.46586/tches.v2021.i3.552-598. url: https://tches.iacr.org/index.php/TCHES/article/view/8986.

[49] Naila Mukhtar et al. “Improved Hybrid Approach for Side-Channel Analysis Using Efficient Convolutional Neural Network and Dimensionality Reduction”. In: *IEEE Access* 8 (2020), pp. 184298–184311. doi: 10.1109/ACCESS.2020.3029206.

[50] Pedro Sanchez Munoz et al. “Analyzing the Resource Utilization of AES Encryption on IoT Devices”. In: *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* (2018), pp. 1200–1207.

[51] D. Opitz and R. Maclin. “Popular Ensemble Methods: An Empirical Study”. In: *Journal of Artificial Intelligence Research* 11 (Aug. 1999), pp. 169–198. doi: 10.1613/jair.614. url: https://doi.org/10.1613/jair.614.

[52] S. Gopal Krishna Patro and Kishore Kumar Sahu. *Normalization: A Preprocessing Stage*. 2015. arXiv: 1503.06462 [cs.OH].

[53] Guilherme Perin, Lichao Wu, and Stjepan Picek. “Exploring Feature Selection Scenarios for Deep Learning-based Side-channel Analysis”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2022.4 (Aug. 2022), pp. 828–861. doi: 10.46586/tches.v2022.i4.828-861. url: https://tches.iacr.org/index.php/TCHES/article/view/9842.

[54] Stjepan Picek et al. “The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-channel Evaluations”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (Nov. 2018), pp. 209–237. doi: 10.46586/tches.v2019.i1.209-237.

[55] Emmanuel Prouff and Matthieu Rivain. “Theoretical and Practical Aspects of Mutual Information Based Side Channel Analysis”. In: *Applied Cryptography and Network Security*. Ed. by Michel Abdalla et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 499–518. isbn: 978-3-642-01957-9.

[56] Emmanuel Prouff et al. *Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database*. Cryptology ePrint Archive, Paper 2018/053. https://eprint.iacr.org/2018/053. 2018. doi: 10.1007/s13389-019-00220-8. url: https://eprint.iacr.org/2018/053.

[57] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. “SCAUL: Power side-channel analysis with unsupervised learning”. In: *IEEE Transactions on Computers* 69.11 (2020), pp. 1626–1638.

[58] Lior Rokach. “Ensemble-based classifiers”. In: *Artif. Intell. Rev.* 33 (Feb. 2010), pp. 1–39. doi: 10.1007/s10462-009-9124-7.

[59] F.-X. Standaert, E. Peeters, and J.-J. Quisquater. “On the masking countermeasure and higher-order power analysis attacks”. In: *International Conference on Information Technology: Coding and Computing (ITCC’05) - Volume II*. Vol. 1. 2005, 562–567 Vol. 1. doi: 10.1109/ITCC.2005.213.

[60] François-Xavier Standaert, Benedikt Gierlichs, and Ingrid Verbauwhede. “Partition vs. Comparison Side-Channel Distinguishers: An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices”. In: *Information Security and Cryptology – ICISC 2008*. Ed. by Pil Joong Lee and Jung Hee Cheon. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 253–267. isbn: 978-3-642-00730-9.

[61] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. “A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks”. In: *EUROCRYPT ’09: Proceedings of the 28th Annual International Conference on Advances in Cryptology*. Berlin, Heidelberg, 2009, pp. 443–461.

[62] National Institute of Standards and Technology. “Advanced Encryption Standard”. In: *NIST FIPS PUB 197* (2001).

[63] Nima Tajbakhsh et al. “Convolutional neural networks for medical image analysis: Full training or fine tuning?” In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1299–1312.

[64] Biaoshuai Tao and Hongjun Wu. “Improving Biclique Cryptanalysis on AES”. In: vol. 9144. June 2015. isbn: 978-3-319-19961-0. doi: 10.1007/978-3-319-19962-7_3.

[65] *Tensorflow*. url: https://www.tensorflow.org/.

[66] Sergios Theodoridis and Konstantinos Koutroumbas. “Chapter 2 - Classifiers Based on Bayes Decision Theory”. In: *Pattern Recognition (Fourth Edition)*. Ed. by Sergios Theodoridis and Konstantinos Koutroumbas. Fourth Edition. Boston: Academic Press, 2009, pp. 13–89. isbn: 978-1-59749-272-0. doi: https://doi.org/10.1016/B978-1-59749-272-0.50004-9. url: https://www.sciencedirect.com/science/article/pii/B9781597492720500049.

[67] Benjamin Timon. “Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis”. In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019.2 (2019), pp. 107–131. doi: 10.13154/tches.v2019.i2.107-131. url: https://doi.org/10.13154/tches.v2019.i2.107-131.

[68] Edna Chebet Too et al. “A comparative study of fine-tuning deep learning models for plant disease identification”. In: *Computers and Electronics in Agriculture* 161 (2019), pp. 272–279.

[69] Grigorios Tsoumakas and Ioannis Katakis. “Multi-Label Classification: An Overview”. In: *International Journal of Data Warehousing and Mining* 3 (Sept. 2009), pp. 1–13. doi: 10.4018/jdwm.2007070101.

[70] Nicolas Veyrat-Charvillon and François-Xavier Standaert. “Mutual Information Analysis: How, When and Why?” In: *Cryptographic Hardware and Embedded Systems - CHES 2009*. Ed. by Christophe Clavier and Kris Gaj. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 429–443. isbn: 978-3-642-04138-9.

[71] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. “A survey of transfer learning”. In: *Journal of Big data* 3.1 (2016), pp. 1–40.

[72] David H. Wolpert. “Stacked generalization”. In: *Neural Networks* 5.2 (1992), pp. 241–259. issn: 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80023-1. url: https://www.sciencedirect.com/science/article/pii/S0893608005800231.

[73] Yoo-Seung Won et al. “Non-Profiled Side-Channel Attack Based on Deep Learning Using Picture Trace”. In: *IEEE Access* 9 (2021), pp. 22480–22492. doi: 10.1109/ACCESS.2021.3055833.

[74] Lichao Wu, Guilherme Perin, and Stjepan Picek. *Hiding in Plain Sight: Non-profiling Deep Learning-based Side-channel Analysis with Plaintext/Ciphertext*. Cryptology ePrint Archive, Paper 2023/209. https://eprint.iacr.org/2023/209. 2023. url: https://eprint.iacr.org/2023/209.

[75] Lichao Wu, Guilherme Perin, and Stjepan Picek. *I Choose You: Automated Hyperparameter Tuning for Deep Learning-based Side-channel Analysis*. Cryptology ePrint Archive, Paper 2020/1293. https://eprint.iacr.org/2020/1293. 2020. url: https://eprint.iacr.org/2020/1293.

[76] Lichao Wu and Stjepan Picek. “Remove Some Noise: On Pre-processing of Side-channel Measurements with Autoencoders”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020.4 (Aug. 2020), pp. 389–415. doi: 10.13154/tches.v2020.i4.389-415. url: https://tches.iacr.org/index.php/TCHES/article/view/8688.

[77] Donna Xu et al. *A Survey on Multi-output Learning*. 2019. doi: 10.48550/ARXIV.1901.00248. url: https://arxiv.org/abs/1901.00248.

[78] Gabriel Zaid et al. “Methodology for efficient CNN architectures in profiling attacks”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2020), pp. 1–36.

[79] Gabriel Zaid et al. “Ranking loss: Maximizing the success rate in deep learning side-channel analysis”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021), pp. 25–55.

[80] Boyun Zhang et al. “Malicious Codes Detection Based on Ensemble Learning”. In: vol. 4610. July 2007, pp. 468–477. isbn: 978-3-540-73546-5. doi: 10.1007/978-3-540-73547-2_48.

[81] Min-Ling Zhang and Zhi-Hua Zhou. “A review on multi-label learning algorithms”. In: *IEEE transactions on knowledge and data engineering* 26.8 (2013), pp. 1819–1837.

Appendix A

Model architecture comparison

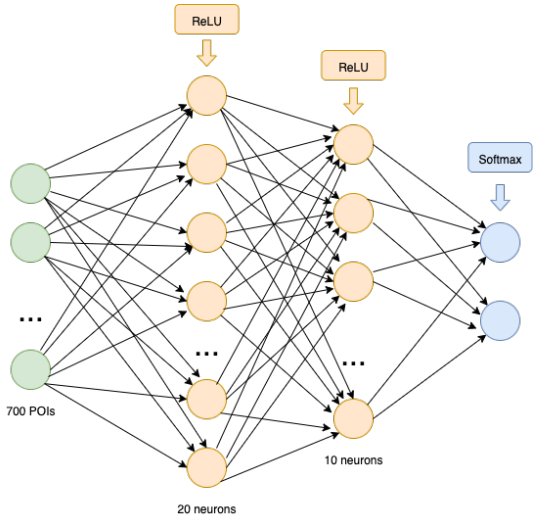


Figure A.1: The DLLA-MLP architecture introduced in [67].

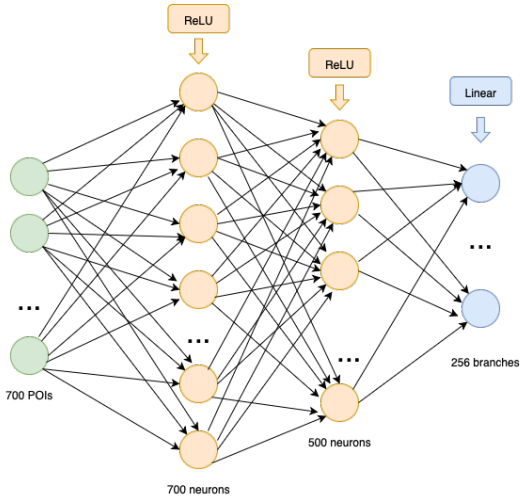


Figure A.2: The MOR-MLP architecture introduced in [17].

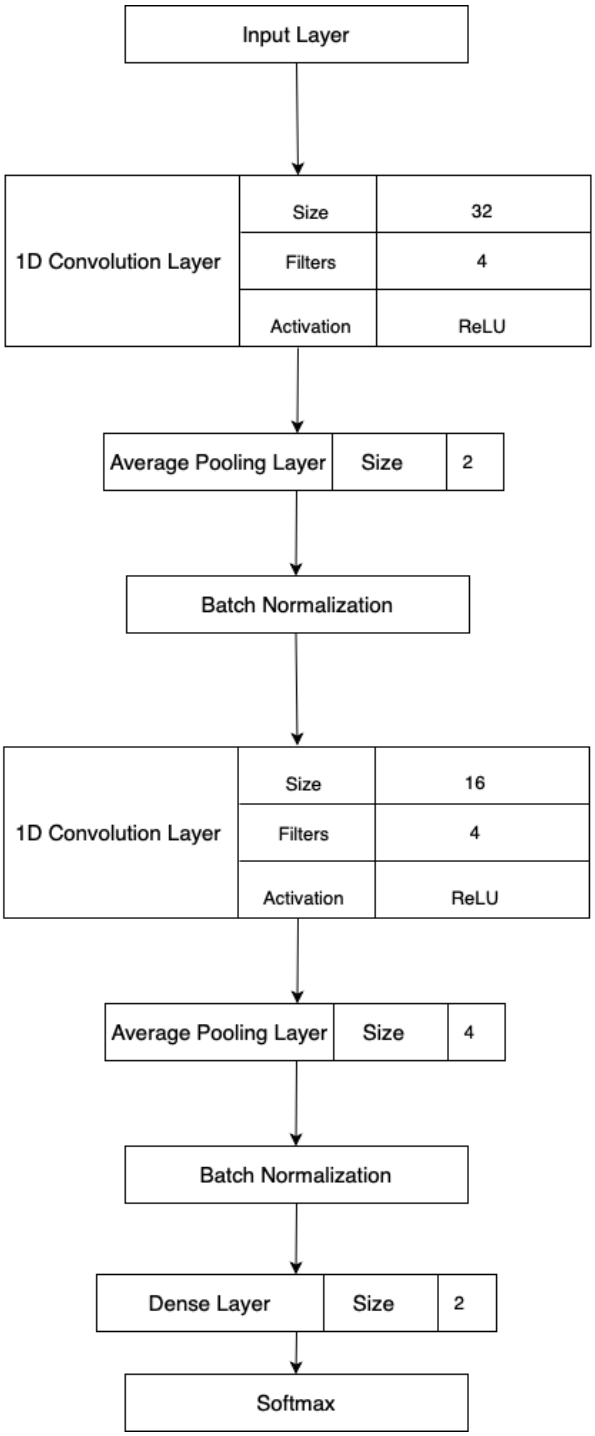


Figure A.3: The DDLA-CNN architecture introduced in [67].

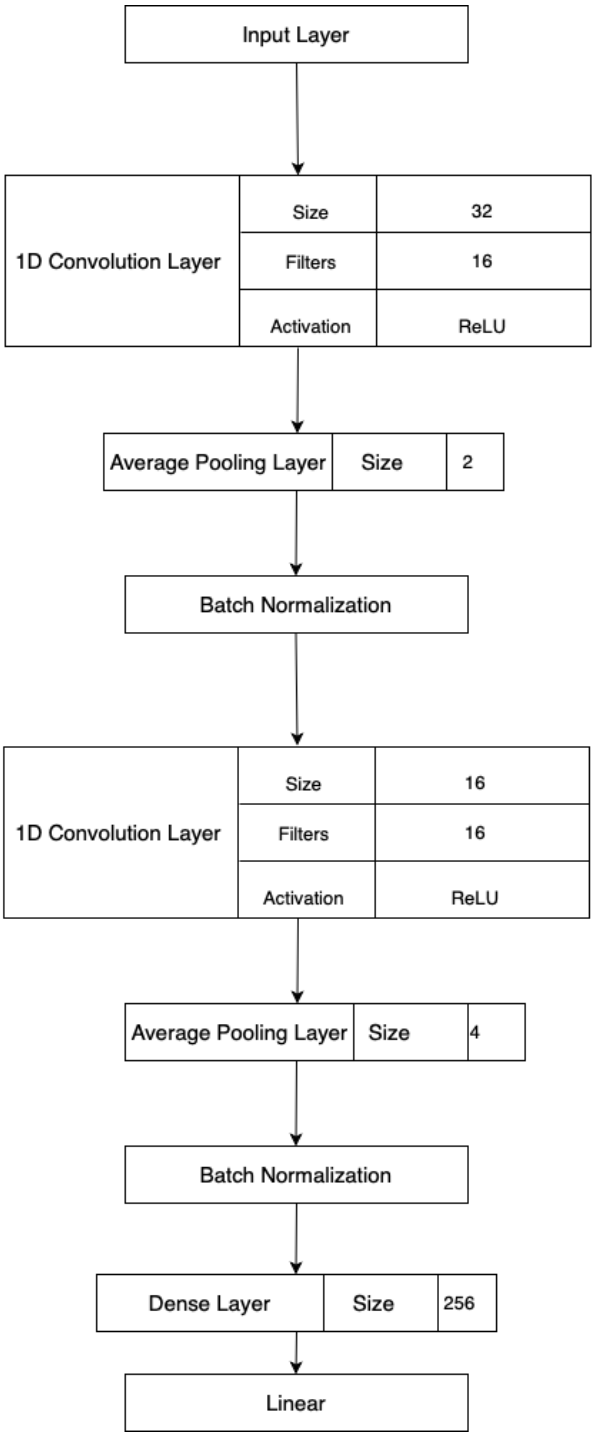


Figure A.4: The MOR-CNN architecture introduced in [17].

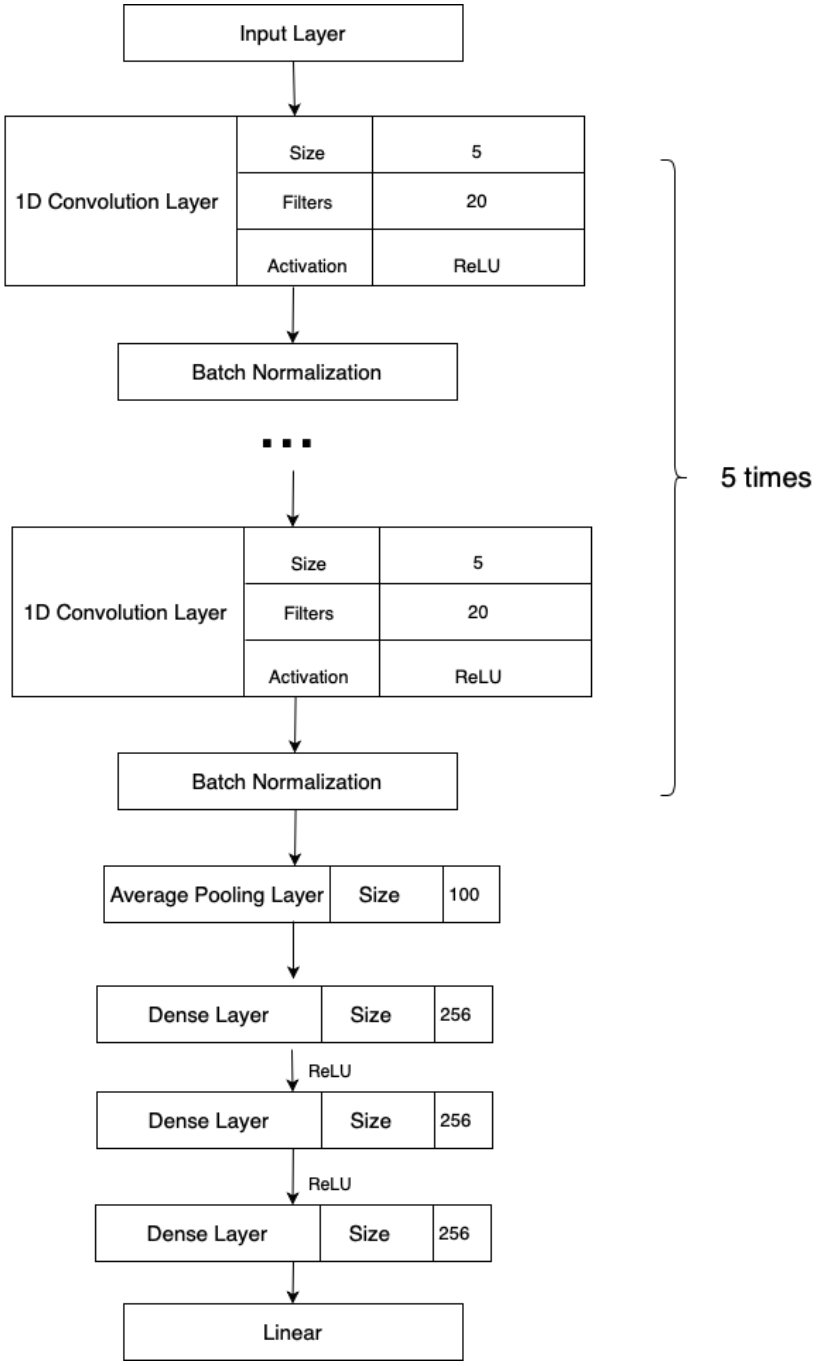


Figure A.5: The proposed MOR-CNN architecture.