Universiteit Leiden
The Netherlands

Bachelor's Programme in Data Science & Artificial Intelligence

Mechanistic Interpretability in Deep Learning-Based Side-Channel Attacks

Anna Sips

Supervisors:
Nele Mentens & Sengim Karayalçin

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

01/06/2025

Abstract

Deep learning has shown strong performance in side-channel attacks (SCA) but remains difficult to interpret due to the black-box nature of neural networks. Recent research introduced mechanistic interpretability (MI) techniques to understand what deep learning models learn. This thesis applies several MI methods to convolutional neural networks (CNN) and multi-layer perceptrons (MLP) trained on the ASCAD variable key dataset using the Identity leakage model. Results show that both models can extract meaningful internal features. However, the clarity of internal representations is weaker than in prior work. This thesis provides insights into the interpretability of deep learning in SCA and provides suggestions for further research.

Contents

1 Introduction

In recent years, side-channel attacks (SCA) have emerged as a powerful technique for extracting secret information from cryptographic implementations by analyzing physical leakages such as power consumption [KJJ99], electromagnetic radiation [AARR03], or timing [Koc96]. Using deep learning for SCAs has proven highly effective [WRAp+24]. A big challenge with deep learning models is that they operate as a black box. Even when a model successfully recovers a cryptographic key, it is typically unclear which part of the trace was used, what features were extracted, and why certain decisions were made [KKP25]. This lack of interpretability limits the practical usefulness of such models in security evaluation settings, where understanding the attack mechanism is just as important as achieving key recovery.

Mechanistic interpretability (MI) tries to reverse engineer the internal behavior of neural networks. Recent research has applied MI techniques such as logit analysis, activation visualization, and activation patching to better understand what features deep learning models are using during SCA. These techniques have shown promising results on specific datasets and model configurations [KKP25]. This study investigates whether the mechanistic interpretability techniques from Karayalçin et al. [KKP25] work on different models and datasets in DLSCA.

The research question is: *How well do the techniques proposed in the paper "It's Not Just a Phase: On Investigating Phase Transitions in Deep Learning-based Side-channel Analysis" perform on other models and datasets from the literature?*

1.1 Contributions

This thesis contributes the following:

- This thesis extends previous work from Karayalçin et al. (2025) [KKP25]. It uses simpler MLP models from [PWP21] and introduces a CNN architecture from [PWP21] into mechanistic interpretability analysis of profiling SCA, to explore how these techniques apply to convolutional models.
- It focuses on the identity (ID) leakage model in the ASCAD variable key dataset. Unlike the LSB-biased traces used in [KKP25], this setup presents a more challenging target, as no strong bit biases are present. The dataset also differs in trace preprocessing: 1400 carefully selected points of interest are used rather than 2000 resampled points from a larger window of 20,000 samples.
- It evaluates how well mechanistic interpretability techniques generalize to different architectures (CNN and MLP) and leakage models (ID).

1.2 Thesis overview

This undergraduate thesis was written at the Leiden Institute of Advanced Computer Science (LIACS) under the supervision of Nele Mentens and Sengim Karayalçin. The thesis is structured as follows: Section 2 provides background on side-channel analysis, deep learning, and interpretability. Section 3 gives definitions of key concepts and terms. Section 4 discusses related work. Section 5

describes the experimental setup and results. Section 6 discusses the results. Section 7 concludes the thesis and discusses future research directions.

2 Background Information

This section provides the necessary background for understanding mechanistic interpretability (MI) in deep learning-based side-channel analysis (DLSCA). It begins with an introduction to SCA and leakage models, then describes the ASCAD dataset and deep learning models, and closes with an explanation of MI.

2.1 Side-channel Analysis

Side-channel analysis (SCA) is a form of attack that extracts secret information from cryptographic implementations by observing unintended physical leakages such as power consumption [KJJ99], electromagnetic radiation [AARR03], or timing variation [Koc96]. Instead of exploiting weaknesses in the cryptographic algorithm itself, SCA focuses on the physical behavior of devices while executing cryptographic operations [PPM+23, KKP25]. In a DLSCA attack, an attacker records multiple power traces during encryption and uses these to infer intermediate computations inside the cryptographic algorithm. Machine learning techniques can be trained to derive parts of the secret key by analyzing power consumption patterns, for example. SCA has proven successful even against strong cryptographic schemes such as Advanced Encryption Standard (AES) [oSND+01]. AES is a byte-oriented block cipher that operates over multiple rounds, each consisting of several nonlinear and linear operations. Due to its widespread use when encrypting information, AES is also a usual target in side-channel analysis research [KKP25]. A common target point in AES is the output of the S-box operation because an S-box combines key and plaintext information in a nonlinear way. As a result, many attacks focus on recovering one byte of the key (subkey) by analyzing this intermediate value:

$$IV = S\text{-box}(p \oplus k)$$

where p is a plaintext byte and k is the corresponding key byte. In SCA, if the attacker recovers a subkey byte, this is often enough because this process can be repeated to reconstruct the entire secret key [KKP25]. The essence of SCA is to match secret-dependent predictions of physical leakages with actual measurements to determine the most likely data being processed [PPM+23].

There are two forms of SCA: profiling and non-profiling [Tim19]. In profiling attacks, the attacker has access to a duplicate of the target device and could collect labeled data with known keys to train a model. This model is later applied to unlabeled traces from the target device. Profiling attacks are generally more powerful, especially when using deep learning [WRAp+24]. In non-profiling attacks, the attacker learns directly from the target's traces. In this paper, we look at profiling attacks.

To defend against SCA, many implementations include countermeasures such as masking (randomizing internal values) and hiding (flattening power signatures). However, recent work shows that even masked implementations can be vulnerable to modern attacks, like those based on deep learning [KKP25, HGG19].

So, the dataset contains measurement data and labels, but these labels are based on something: a leakage model. Such a model determines how the physical leakages are related to secret values.

2.1.1 Leakage Models

To apply supervised machine learning techniques in SCA, labeled data is needed. Each power trace must be associated with a specific value that the model should learn to predict. Since the secret key

itself is not directly observable, researchers define leakage models: as functions that map internal cryptographic states (such as intermediate values in AES) to labels that approximate how physical leakage behaves [WRAp+24].

Several standard leakage models that are commonly used in deep learning-based SCA (DLSCA) are Hamming Weight (HW), Most/Least Significant Bit (MSB/LSB), and Identity (ID). The HW model suggests that the amount of leakage is correlated with the number of '1' bits in a processed byte. For example, the byte value 0xF0 (11110000) has an HW of 4. It maps each byte to an integer. The HW model has 9 possible output classes (0 to 8). This leakage model assumes that each bit has the same contribution to the leakage. The Most/Least Significant Bit assumes that the leakage depends on a specific bit position, especially the most or least significant bit. Lastly, the Identity leakage model assumes that the leakage is proportional to the values at the intermediate variable, here the output of the S-box. So, this assumes that each bit has a different importance. This model has 256 output classes: one for each byte. So each of these models makes different assumptions about the hardware's physical leakage. [KKP25, WRAp+24].

Finally, it is important to note that the true leakage behavior of a device may not perfectly match any of the models. In short, leakage models tell us what the model is supposed to learn.

To evaluate side-channel leakages, a dataset is necessary. The next section introduces the most widely used benchmark in this domain: the ASCAD dataset.

2.2 ASCAD Data Set

To evaluate deep learning-based side-channel attacks in a controlled and reproducible setting, the ASCAD dataset introduced in 2018 has become the standard benchmark [BPS+20]. ASCAD stands for Advanced Side-Channel Analysis Database.

The dataset contains power traces captured from an 8-bit ATmega microcontroller while it performs AES-128 encryption. Each trace corresponds to a single encryption and consists of measurements of the device's power consumption over time. These traces capture the physical leakage that SCA tries to exploit [BPS+20]. To avoid processing lots of data, a window of 1400 points of interest is extracted around the leakage spot. The ASCAD dataset includes two main labeled datasets:

- **200 000 profiling traces:** Used to train machine learning models, with known plaintexts and keys.
- **100 000 Attack traces:** Used to evaluate the performance of the previously trained model on unseen traces.

In most side-channel evaluations, the point of interest is the output of the S-box, which combines key and plaintext in a nonlinear way. The goal is to recover this intermediate value or infer its structure based on physical traces. The ASCAD dataset thus provides a structured way to test whether machine learning models can learn these relationships. Deep learning, a subfield of machine learning, has become more popular in SCA in recent years. The next section provides an introduction to deep learning models.

2.3 Deep Learning Models

Deep learning is a subfield of machine learning that is inspired by how the human brain processes information. It uses multi-layered artificial neural networks to model complex relationships in high-dimensional data. Deep learning has already been successfully applied to many fields like image recognition [HZRS16] and natural language processing [YHPC18], and more recently in side-channel analysis [Tim19, PKWP22].

A deep learning model is trained in multiple epochs. In each epoch, the model processes the entire training dataset and optimizes its internal parameters to minimize a loss function, using optimization techniques such as gradient descent. The gradient indicates how the function changes, and the gradient descent uses that information to adjust the parameters and improve the model [PBPPM09]. In this section, we discuss two common types of deep learning used in SCA: the Multi-Layer Perceptron (MLP) and the Convolutional Neural Network (CNN).

2.3.1 Multi-layer Perceptron

A Multi-layer Perceptron (MLP) is one of the most commonly used deep learning models. It consists of fully connected layers, where each neuron in a layer is connected to all neurons in the next layer [PBPPM09, GD98]. Inspired by biological neurons, these artificial neurons perform simple mathematical operations in parallel.

An MLP is a feedforward neural network: the signals flow from the input layer through one or more hidden layers to the output layer, without cycles. Each neuron applies a weighted transformation to its input, adds a bias term, and passes the result through a nonlinear activation function [GD98, GBC16]. Figure 1 [CASQ+23] (left) illustrates a basic MLP architecture, where the input vector $X = x_1, x_2, \ldots, x_n$ is transformed into an output vector $Y = y_1, y_2, \ldots, y_m$ through hidden layers. While relatively simple, MLPs are capable of learning complex nonlinear patterns.

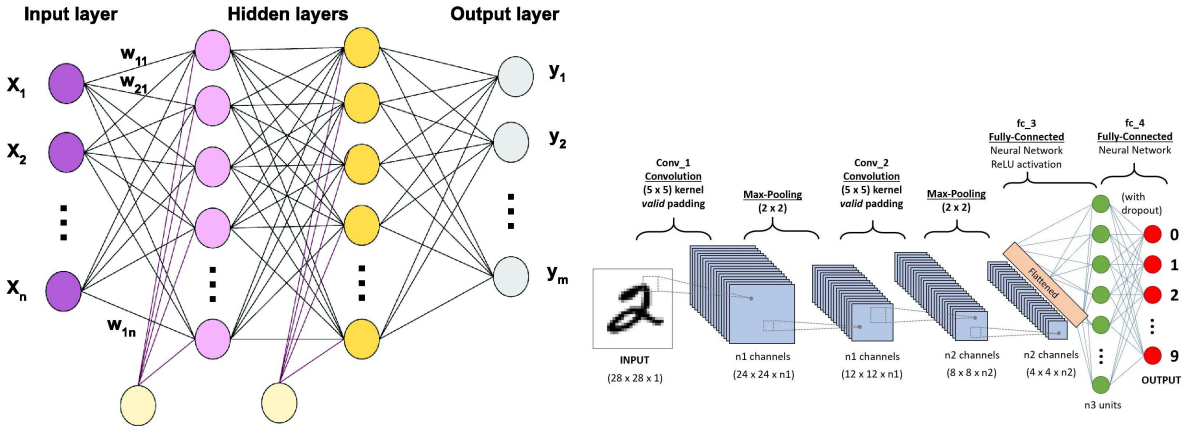

Figure 1: Left: an example of a Multi-layer Perceptron, with input layers, hidden layers, and output layers [CASQ+23]; Right: example of a Convolutional neural network that classifies handwritten digits [Sah23].

2.3.2 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a deep learning model designed to detect local patterns in structured data, such as images or time series. In contrast to a fully connected architecture like the MLP, CNNs consist of different types of layers: convolutional layers, pooling layers, and fully connected layers. While an MLP treats each input value independently, a CNN is better at processing spatial regions simultaneously by applying filters (kernels) [MPP16, ON15]. The convolutional layer applies a set of learnable filters that slide over the input data, where each filter is designed to detect specific features, such as edges or local patterns [ON15]. The output of the convolution layer is a feature map, highlighting where the detected patterns occur in the input [MPP16]. Following the convolutional layer, there is a pooling layer, which reduces the spatial dimensionality of the feature maps, a process known as downsampling. This reduces the computational cost and increases robustness to small shifts in the input. A common pooling operation is max pooling, where the maximum value within a local window is selected [ON15, MPP16] At the final stage, the network includes fully connected layers, which function similarly to those in a standard MLP and produce a final class decision at the end [ON15]. An example of a CNN architecture is shown in Figure 1 [Sah23], which classifies handwritten digits. Similar CNNs can be used to classify power traces in SCA.

CNNs are particularly effective in SCA because they can handle desynchronization: small variations in the timing of power traces. This makes them more robust than MLPs in real-world SCA scenarios [MPP16].

2.3.3 Deep Learning Models for SCA

Deep learning has significantly advanced SCA in recent years. When the ASCAD dataset was introduced in 2018 [BPS+20], key recovery required many traces and was computationally intensive. Now, deep learning models can often break these targets using only a single trace [WRAp+24] There are two main advantages of deep learning-based SCA (DLSCA). First, it can effectively break targets even when protected with countermeasures, such as masking and hiding, as said in 2.1. Second, it has the ability to learn directly from raw traces, often eliminating the effort for pre-processing. These advantages make deep learning powerful in SCA [PPM+23]. An interesting phenomenon observed in DLSCA training is phase transitions or grokking effect [NCL+23]. After several epochs with minimal improvement, the model may suddenly generalize and start recovering key information. This turning point often corresponds to internal changes in the learned representations. Understanding how and where these transitions occur could improve both the design of robust countermeasures and our understanding of deep learning itself. The next section introduces the concept of Mechanistic Interpretability, which aims to reverse-engineer the internal behavior of deep learning models.

2.4 Mechanistic Interpretability

The goal of mechanistic interpretability (MI) is to reverse-engineer deep neural networks. This is done by trying to uncover internal structures and processes that lead to model predictions [Ola22]. Unlike black-box interpretability approaches, MI treats the network as a mechanistic system: a composition of understandable parts, such as neurons, weights, and circuits. The goal is to decompose network behavior into interpretable components, allowing researchers to understand not just what the model predicts but also why it makes those predictions.

A challenge in MI is to overcome the *curse of dimensionality*, as neural networks operate in high-dimensional spaces. Olah [Ola22] suggests two strategies for this challenge: (1) studying networks with low-dimensional inputs, and (2) analyzing local behavior around specific data points. This allows for a manageable examination of parameters, activations, and memory representations, ideally segmenting them into variables that can be interpreted independently.

In a related work, Olah et al. [OCS+20] propose a more detailed approach to understanding neural networks, suggesting that rather than seeking high-level, simplified explanations, the focus should be on "zooming in" on individual neurons and their connections. By closely looking at the relationships between neurons and the flow of activations, the authors demonstrate that it is possible to uncover meaningful algorithmic structures embedded in the weights of a network. They argue that neural networks may be made up of meaningful, reusable circuits that are groups of neurons and weights that implement specific functions. Examples of those circuits are curve detectors in vision models or frequency filters. These circuits, formed by weighted connections between interpretable features, can be studied analogously to biological neural systems. The authors propose that understanding these circuits may eventually enable a full mechanistic understanding of neural models in tasks and domains.

Building on these ideas, Karayalçin et al. [KKP25] apply MI techniques to Deep Learning-based Side-channel Analysis (DLSCA). The hypothesis is that deep learning models perform things as some kind of meaningful interpretable algorithms, and then reverse engineer at the location of or after the phase transitions [NCL+23]. Below, the main techniques used in this work are described.

2.4.1 Probing

Probing is a method used to study the internal representations of neural networks [AB18, PKWP22]. The key idea is to test whether certain features are encoded in the activations of a trained model. Features can be specific bits of an intermediate value for example. A probe is a small, typically linear classifier trained on the frozen activations of a network layer to predict a selected feature. If the probe can predict the feature with accuracy significantly above chance (0.5 for binary bits), this indicates that the feature is represented within that layer's activations. By training probes across layers and at different epochs, it is possible to analyze where and when a model begins to encode relevant features [PKWP22].

2.4.2 Logit analysis

This technique analyzes the model output (before applying softmax) for different classes immediately after phase transitions. By examining these outputs, it becomes possible to identify which classes the model distinguishes most clearly and which are commonly confused. The height of the class indicates how strongly the model "considers" that class as a possible outcome. This helps to assess how strongly the model associates certain inputs with specific outputs and whether certain patterns are well represented internally.

2.4.3 Activation Analysis (PCA)

This technique explores the internal representations of the model by applying Principal Component Analysis (PCA) to activation vectors in intermediate layers. PCA reduces the dimensionality of the activations while preserving variance, so the learned 'structure' can be visualized in 2D plots.

Clusters in PCA plots can indicate that the model internally represents certain input patterns or bit combinations. For example, diagonal or grid-like patterns may correspond to interactions between secret shares or masked intermediaries. This form of visualization helps trace how meaningful representations emerge during training.

2.4.4 Activation patching

After identifying the relevant principal components, activation patching tests whether these components are causally responsible for the predictions. The idea is to fix all directions in the activation space except one and vary that one direction across inputs. This allows for controlled interventions: if changing a single component alters the model's output, that component is causally relevant. Finally, the Signal-to-Noise Ratio (SNR) is used to assess whether patched outputs align with known physical leakage points.

2.5 Evaluation Metrics

To assess what a model has learned and how information is represented internally, evaluation techniques:

- **Perceived Information (PI)**: A measure that estimates how much information about the key (or subkey) is present in the model's output probabilities. If the model assigns a low probability to the correct subkey, it has probably not captured the true leakage relationships, making the internal representations less meaningful. A higher PI indicates better recovery of secret information.
- **Probe Accuracies**: Probe accuracies measure how well individual bits of secret intermediate values can be linearly predicted from hidden activations. Hidden activations in this experiment are S-box input/output bits. Linear probes are trained for each bit, using frozen activations of the model as input. Accuracy above random (0.5) indicated that the bit is represented in the network's internal representations. For example, a probe trained to predict bit 1 of the S-box output with an accuracy of 0.8 implies that this bit is actively encoded and likely used in decision making [PKWP22].
- **Logit Distributions**: Logits are analyzed per class and per trace group (Y_i) to detect common misclassifications and output patterns. In each group, the median logit values are plotted for the 256 output classes. Ideally, the red markers that represent the expected classes for that group should coincide with the highest peaks in the blue logit lines. When this alignment occurs, it indicates that the model strongly associates that input group with the correct output classes, suggesting that it has learned a structured internal representation.
- **Principal Component Analysis (PCA)**: Used to visualize activation structure in a lower-dimensional space. Clustering or alignment of activation projections with specific input/output bit combinations suggests an interpretable internal structure.
- **Activation Patching and Signal-to-Noise Ratio (SNR)**: Activation patching is used to test whether specific principal components are causally responsible for predictions. The

resulting model outputs are then analyzed using SNR to quantify the strength of signal-leakage alignment.

3 Definitions

This section defines key concepts and terms used throughout the thesis.

Activation Patching A mechanistic interpretability technique where activation components are replaced to test their causal role in predictions.

ASCAD Widely used SCA dataset for evaluating side-channel attacks.

Probe accuracy A metric that indicates how well a linear classifier (probe) can predict specific bits of internal intermediate values (e.g., S-box output) from hidden layer activations. Higher accuracy indicates stronger internal encoding [PKWP22].

Hamming Weight (HW) A leakage model that assumes that leakage is proportional to the number of '1' bits in a byte.

Identity (ID) A leakage model that assumes that leakage is proportional to the actual value of the intermediate variable.

Leakage model An assumption about how the physical leakage relates to the secret internal state (e.g HW, ID, and LSB/MSB models)

Logit The raw output of the final layer of a neural network, before applying softmax. The higher the logit, the more confident the model is in predicting the class.

NIST Advanced Encryption Standard (AES) cipher A widely used algorithm used for symmetric encryption and decryption [oSND+01].

Perceived Information (PI) A measure that estimates how much information about the key is present in the model's output probabilities [NCL+23, KKP25].

Phase Transition, Grokking A point during training where a sudden increase of learned information and where the model starts to generalize and extract information after several training steps, where there was no improvement [KKP25].

Principal Component Analysis (PCA) A dimensionality reduction technique used to visualize internal activations by projecting them in directions of maximum variance.

Side-Channel Attack (SCA) A form of attack where the attacker looks for physical leaks (e.g., time, power consumption) to recover secret keys.

Softmax An activation function that converts logits into a probability distribution over classes, used as an output for classification networks.

4 Related Work

Several studies have investigated the application of deep learning in SCA. When the ASCAD dataset was proposed in 2018, recovering secret keys still required significant effort, and research on deep learning for SCA was limited [BPS+20]. Now, secret keys can often be recovered using single attack traces, even when countermeasures are present [WRAp+24]. Because DLSCA is a relatively new field, most studies have only focused on optimizing deep learning networks rather than on explainability and interpretability [PKWP22]. This section discusses studies that specifically investigate interpretability in deep learning-based profiling SCA, with an emphasis on techniques that try to explain the internal workings of neural networks.

We begin with studies that are not so focused on the interpretability of all layers, but are nevertheless concerned with understanding neural networks in a broad sense. We start with Zaid et al. (2020) [ZBHV20], who applied visualization techniques, such as weight visualization and feature maps to better understand feature selection and the role of hyperparameters. Their goal was to investigate whether classification complexity could be reduced by simplifying feature selection in the early stages of training. Their work did not aim to localize side-channel leakages, but rather to optimize network efficiency. They concluded that CNNs do not need to be highly complex to achieve good performance in SCA. This insight is important, as it suggests that simpler models may offer a better trade-off between accuracy and efficiency in practical applications. Masure et al. (2019) [MDP19] focused on input gradients to identify important sample points. This method provides a way to visualize which parts of the input data had the most influence on the model's decisions. We can conclude that both studies [ZBHV20, MDP19] focus on the input layer, they provide valuable insights but not much information about the internal workings of the hidden layers.

Some studies focused more directly on internal representations and explaining them, but are limited in how good they are. For example van der Valk et al (2021) [vdVPB21]. [vdVPB21] tries to explore what hidden layers in neural networks learn from side-channel traces in profiling SCA by using Singular Vector Canonical Correlation Analysis (SVCCA). They applied this method to both MLPs and CNNs and found that internal representations differ across datasets and masking schemes. Interestingly, they also observed similarities between datasets that, on the surface, appear unrelated. SVCCA proved useful for comparing neural network layers and models and revealed that datasets sharing the same leakage model had more similar internal structures than datasets with or without countermeasures. This finding suggests that leakage models play a more crucial role in shaping internal representations than the presence of countermeasures. Wu et al. [WWJ+24] used ablation techniques to evaluate the sensitivity and resilience of individual layers, by systematically removing components to assess their impact. However, they noted that interpreting the results of such analyses remains complex and challenging.

Other research has applied information-theoretic approaches to understand internal processes. Perin et al (2020) [PBP20] used mutual information to make an early stopping mechanism, allowing the model to stop training if they have captured enough information about the secret key. In another work, Perin et al. (2022) [PKWP22] studied how deep learning models encode internal information during profiling SCA. They introduced the concept of ExDL-SCA, a framework that uses information-theoretic metrics to better understand what is represented in the hidden layers. Their approach builds on the information bottleneck theory (IB), which views deep learning training as a process consisting of first fitting the data and then compressing unnecessary information. They

employed perceived information to quantify how much information about hidden values, such as masks, is encoded in internal representations. This work is directly relevant to our study because it also wants to explain the role of hidden layers in profiling SCA, although it focuses on analyzing internal layers rather than linking them to specific trace points or physical leakage locations.

Finally, Karayalçin et al. (2025) [KKP25] introduced mechanistic interpretability techniques that are central to our work. They applied methods such as logit analysis, PCA, and activation patching to reverse engineer trained models and identify which features contribute to key recovery and what leakages are being exploited. Their work provided insights into how specific internal structures of neural networks relate to side-channel leakage. Our study builds on these methods but applies them to different parts of the ASCAD dataset and to models with slightly different architectures, to assess how well these techniques generalize.

In summary, prior studies show valuable first steps in the interpretability of deep learning in profiling SCA. However, much of the existing work focuses on input layers or provides limited insight into internal processes. To improve countermeasures in the future, there remains a need for techniques that systematically explain the internal workings of models and provide insight into how and where side-channel leakages are exploited.

5 Experiments

5.1 Experimental Setup

This experiment follows the approach of Karayalçin et al. (2025) [KKP25], assuming that the correct subkey is already known during evaluation. This allows for assessment of whether the model is learning meaningful features, and whether its predictions become increasingly aligned with the correct subkey values. To investigate this, the **Perceived Information (PI)** is tracked over training epochs, which measures how much information about the correct subkey is present in the model's output distribution. A rising PI indicates that the model is learning something relevant.

Next, several mechanistic interpretability analyses are applied that were proposed in the original work to investigate how and where the model internally encodes this information:

1. **Probe accuracy** to evaluate to what extent individual bits of the S-box input and output are encoded in intermediate layers. For each bit, a separate linear probe is trained to predict the bit value based on layer activations. Accuracy above chance (here 0.5) suggests meaningful internal coding [PKWP22].

2. **Logit analysis** examines the model's raw output scores before softmax for trace groups Y_i, grouped by specific input and output combinations.

3. **Principal Component Analysis (PCA)** of hidden activation, where activations are visualized and colored by relevant input and output bit values to assess clustering structure.

The model was trained on the ASCAD variable key from the GitHub repository ASCAD GitHub repository from [BPS+20] using the Identity (ID) leakage model. The variable key dataset has a more complex masking scheme, so this is a harder and more realistic dataset than the fixed key. The input consisted of 1400 points of interest (POIs), selected from the interval [80945, 82345]. The labels correspond to the S-box output byte, and the model is trained with a softmax output layer containing 256 units (one per possible byte value).

Three neural network architectures were evaluated:

- **Using a CNN and ID leakage**: The CNN was based on the best-performing configuration proposed in the GitHub repository at feature_selection_dlsca repository described in the [PWP21] article. The network architecture of the CNN consists of three convolutional layers with 'selu' activation, each followed by max pooling and batch normalization, followed by two dense layers and a softmax output layer. The training was performed for 100 epochs, and the Adam optimizer with a learning rate of 0.001 was used. Furthermore, a batch size of 300 and a categorical cross-entropy loss.

- **Using an MLP and ID leakage**: The MLP used in this experiment was also based on the best-performing configuration proposed in the GitHub repository at feature_selection_dlsca repository described in [PWP21]. The model consists of three hidden layers, each with 20 neurons and `selu` activation, followed by a softmax output layer with 256 classes. The weights were initialized with the `random_uniform` initializer, and the model was trained using the Adam optimizer with a learning rate of 0.0005 and a batch size of 100.

5.2 Results

5.2.1 CNN

Figure 2 shows the perceived information (PI) over training epochs for both train and test sets. PI increases during the first 30 epochs, so there is a phase transition. After that, it rises steadily afterward, indicating that the network successfully learns relevant features under the ID leakage model.

The middle panel of Figure 2 shows bitwise accuracy over training epochs. Input bits 0 and 1 show significantly higher accuracy (≈ 0.6) compared to other bits, which remain near chance. Still, it has a low accuracy in general when compared to the paper [KKP25]. It can be concluded that this dataset makes inference more difficult. Most bits perform near chance level, with bit 1 of the S-box output showing particularly low accuracy. As a result, it was excluded from subsequent logit plots. Overall, the dataset contains less exploitable information, which explains the reduced accuracy.

In the right panel of Figure 2, the median logits are plotted per class for 8 subsets Y_i, where each Y_i is defined by a combination of bits 0 and 1 of the S-box input and bit 0 of the S-box output. Red markers indicate the expected S-box values (i.e., output classes) for each Y_i group, and the blue lines indicate the median logit value per output class. The peaks of the lines match the red markers well, implying that the model strongly associates these trace groups with the correct output classes. Notably, $i = 2, 3, 4, 5$, the peaks are higher than for the other four groups, suggesting a stronger distinction. This uneven performance is probably due to more irregular leakage patterns in this part of the ASCAD dataset.

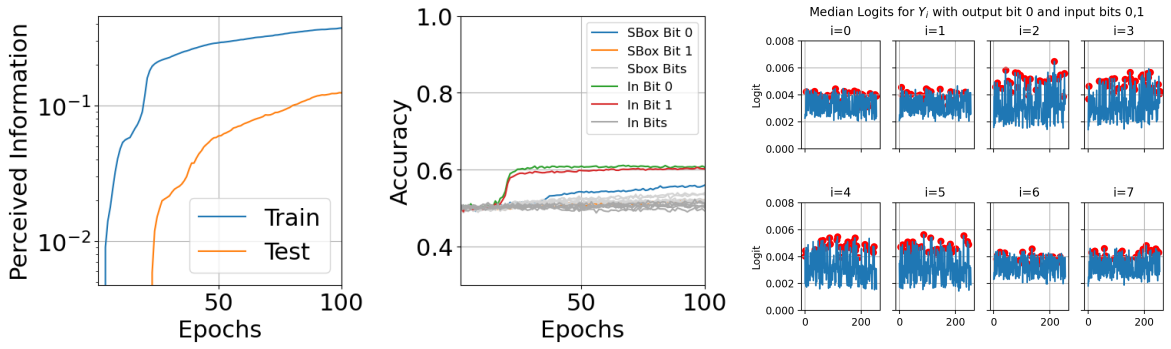

Figure 2: Left: Perceived Information evolution over epochs for the CNN trained with the ID leakage model. Middle: Probe accuracy over epoch for selected S-box and input bits. Right: Median logits per group Y_i.

Figure 3 presents PCA projections of the activations across CNN layers. From layer 5 onward, clustering begins to emerge, especially in the S-box input bit. While four distinct clusters were not consistently visible, several projections showed two or three clear groupings as seen in the left and middle picture of 3. Notably, values like 00 and 11 tend to appear close together, suggesting that the model may encode bitwise parity rather than full identity. In the attack label bit, the clusters are significantly less distinct, as shown in Figure 3.

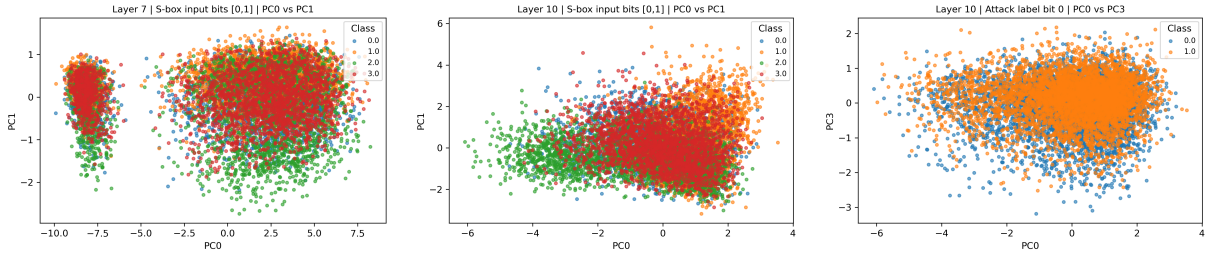

Figure 3: PCA projections of CNN activations. Left: layer 7 (S-box input bits). Middle: layer 10 (S-box input bits). Right: layer 10 (attack bit 0).

5.2.2 MLP

Figure 4 shows the training progression and internal behavior of the model. The Perceived Information (PI) increases sharply in the first 20 epochs and then flattens out. The accuracy of probes shows that input bits 0 and 1 reach around 0.6, while other bits remain close to chance level. This indicated that the MLP can extract some secret information, but less effectively than in the paper [KKP25]. The logit analysis in Figure 4 (right) shows that the red markers (expected classes for each Y_i group) align only partially with the high logits. In particular, groups Y_2 to Y_5 show more confident predictions, while others remain flat and uninformative. This suggests that the model only learns distinguishable structures for a subset of trace groups.

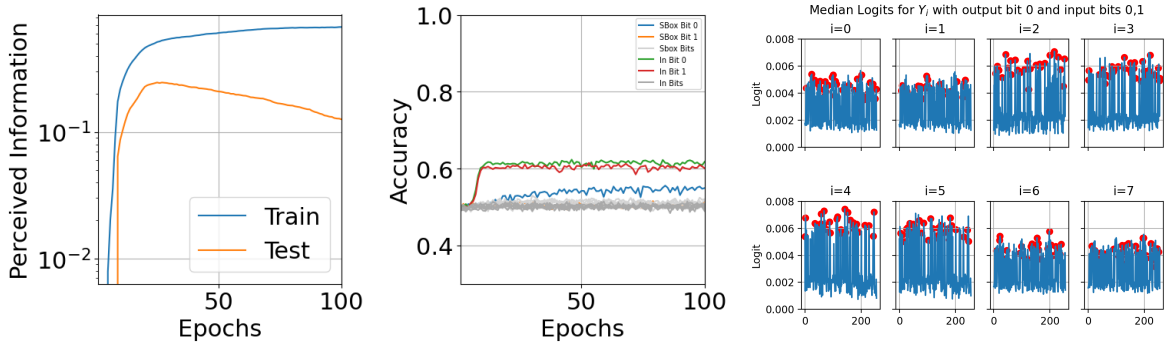

Figure 4: Left: Perceived Information and probe accuracy over epochs for the MLP trained with ID leakage. Middle: Probe accuracy over epoch for selected S-box and input bits. Right: Median logits per group Y_i at epoch 30.

Finally, Figure 5.2.2 shows PCA projections of activations from the first and second layers of the MLP. There is some clustering in the S-box input bit, indicating that the MLP captures some relevant features. But, just like the CNN, these are noisier and less structured than in the paper, which indicates that the internal representations are less well-structured. For the S-box output bit used as an attack label, only weak clustering is observed as shown in Figure 5.2.2 on the right.

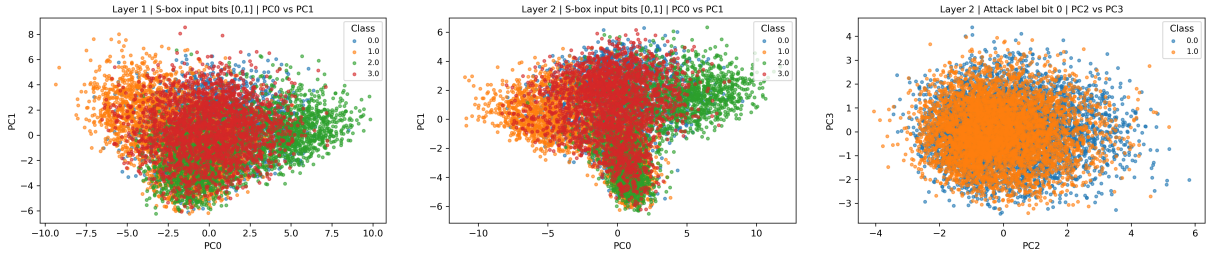

Figure 5: PCA projections of MLP activations. Left: layer 1 (S-box input bits). Middle: layer 2 (S-box input bits). Right: layer 2 (attack bit 0).

6 Discussion

The results demonstrate that both the CNN and MLP models are capable of learning meaningful features from side-channel traces when trained on the ASCAD variable key dataset using the Identity leakage model. Across both architectures, similar trends were observed in multiple interpretability metrics. For instance, the Perceived Information (PI) curves followed nearly identical patterns over training epochs, and both models show above-random probe accuracy (around 0.6) for S-box input bits 0 and 1. These findings suggest that both architectures are able to encode information about the key internally, despite their structural differences.

Interestingly, this study found no major interpretability advantage for one of the deep learning models. The logits of both models exhibited comparable behavior: in both cases, the expected class for each group Y_i tended to correspond to the peak of the logit distribution, although, for the MLP, these peaks appeared slightly higher. Similarly, PCA projections showed the emergence of some clustering structures for both models, though the patterns were generally less clean and more diffuse than those reported in the original study by Karayalçin et al. [KKP25]. The differences in internal representation clarity and strength likely stem from variations in the datasets, model complexity, or training, which would be valuable to investigate further.

Overall, the similarity in results between CNN and MLP suggests that for this dataset and leakage model, the architectural choice may play a smaller role in interpretability than expected. However, more extended studies across leakage models and datasets are needed to validate this finding. Furthermore, while probe accuracies and clustering provide useful signals, deeper causal techniques such as activation patching could strengthen the conclusions about what internal structures are truly responsible for model behavior. The fact that both models achieve similar probe accuracies for specific bits suggests that their internal representations, though trained independently, encode similar types of information.

7 Conclusions and Further Research

This thesis applied mechanistic interpretability techniques from Karayalçin et al. [KKP25] both CNN and MLP models trained on the ASCAD variable key dataset using the ID leakage model. The goal was to evaluate how well these interpretability techniques generalize to different model and dataset combinations within DLSCA.

The results indicate that the interpretability methods provide valuable insights: probe accuracies showed higher accuracies for individual bits, logit values matched, and clustering patterns were observed in the PCA plots. However, the insights were less robust than those in the original paper. This suggests that the effectiveness of these interpretability techniques is sensitive to both the dataset and the models.

For future work, it would be valuable to explore the use of different leakage models (e.g., HW) to provide different insights into model behavior, look at more models with other architectures, or look at other (parts of) datasets like the ASCAD fixed key dataset.

References

[AARR03] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The em side—channel(s). In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 29–45, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[AB18] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2018.

[BPS+20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ascad database. *Journal of Cryptographic Engineering*, 10, 06 2020.

[CASQ+23] Kit Yan Chan, Bilal Abu-Salih, Raneem Qaddoura, Ala' M. Al-Zoubi, Vasile Palade, Duc-Son Pham, Javier Del Ser, and Khan Muhammad. Deep neural networks in the cloud: Review, applications, challenges and research directions. *Neurocomputing*, 545:126327, 2023.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GD98] M.W Gardner and S.R Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14):2627–2636, 1998.

[HGG19] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. Deep neural network attribution methods for leakage analysis and symmetric key recovery. Cryptology ePrint Archive, Paper 2019/143, 2019.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[KKP25] Sengim Karayalçin, Marina Krček, and Stjepan Picek. It's not just a phase: On investigating phase transitions in deep learning-based side-channel analysis, 2025.

[Koc96] 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, page 104–113, Berlin, Heidelberg, 1996. Springer-Verlag.

[MDP19] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Gradient visualization for general characterization in profiling attacks. In *Constructive Side-Channel Analysis and Secure Design: 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3–5, 2019, Proceedings 10*, pages 145–167. Springer, 2019.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings 6*, pages 3–26. Springer, 2016.

[NCL+23] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

[OCS+20] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, March 2020.

[Ola22] Chris Olah. Mechanistic interpretability, variables, and the importance of interpretable bases, 2022. Informal note.

[ON15] Keiron O'shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[oSND+01] National Institute of Standards, Technology (NIST), Morris J. Dworkin, Elaine Barker, James Nechvatal, James Foti, Lawrence E. Bassham, E. Roback, and James Dray Jr. Advanced encryption standard (aes), 2001-11-26 00:11:00 2001.

[PBP20] Guilherme Perin, Ileana Buhan, and Stjepan Picek. Learning when to stop: a mutual information approach to fight overfitting in profiled side-channel analysis. *Cryptology ePrint Archive*, 2020.

[PBPPM09] Marius-Constantin Popescu, Valentina Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8, 07 2009.

[PKWP22] Guilherme Perin, Sengim Karayalcin, Lichao Wu, and Stjepan Picek. I know what your layers did: Layer-wise explainability of deep learning side-channel analysis. Cryptology ePrint Archive, Paper 2022/1087, 2022.

[PPM+23] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55(11), February 2023.

[PWP21] Guilherme Perin, Lichao Wu, and Stjepan Picek. Exploring feature selection scenarios for deep learning-based side-channel analysis. Cryptology ePrint Archive, Paper 2021/1414, 2021.

[Sah23] Sumit Saha. A guide to convolutional neural networks—the eli5 way. *Sturn Cloud Blog. Accessed: Jan*, 17, 2023.

[Tim19] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 107–131, 02 2019.

[vdVPB21] Daan van der Valk, Stjepan Picek, and Shivam Bhasin. Kilroy was here: The first step towards explainability of neural networks in profiled side-channel analysis. In Guido Marco Bertoni and Francesco Regazzoni, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 175–199, Cham, 2021. Springer International Publishing.

[WRAp+24] Lichao Wu, Azade Rezaeezade, Amir Ali-pour, Guilherme Perin, and Stjepan Picek. Leakage model-flexible deep learning-based side-channel analysis. *IACR Communications in Cryptology*, 1(3), 2024.

[WWJ+24] Lichao Wu, Yoo-Seung Won, Dirmanto Jap, Guilherme Perin, Shivam Bhasin, and Stjepan Picek. Ablation analysis for multi-device deep learning-based physical side-channel analysis. *IEEE Transactions on Dependable and Secure Computing*, 21(3):1331–1341, 2024.

[YHPC18] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018.

[ZBHV20] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 1–36, 2020.