LeaX: Class-Focused Explanations for Locating Leakage in Learning-Based Profiling Attacks

Qi Lei and Christian Wressnegger(✉)

KASTEL Security Research Labs, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
c.wressnegger@kit.edu

Abstract. Machine learning can significantly improve power side-channel attacks that derive cryptographic keys from hardware devices. The attacker learns a model that maps side-channel information (e.g., a device's power consumption) to the computation's intermediate states/values, which in turn serve as evidence for the actual key. The model implicitly learns which portions of the power traces leak information and which are mere noise. This knowledge is of great interest to the designer of the cryptographic hardware. Knowing the "leakage points" allows them to tweak the implementation to prevent the leakage. The community has thus investigated the use of "Explainable AI" (XAI) to derive the machine-learning model's knowledge about existing leakage. Unfortunately, with limited success for protected (masked) implementations as used in practice so far. In this paper, we show that very much like for a side-channel attack itself, model analysis using XAI must focus on intermediate values rather than keys, accumulate evidence for all intermediates, and make an informed choice at the end. Doing so successfully, however, requires class-discriminative explanations—a fact overlooked up to now. We present a novel analysis method, LeaX, that uses this observation to precisely pinpoint leakage, especially for masked cryptographic implementations, which prior work has failed to do.

1 Introduction

Machine learning has emerged as a powerful tool in power side-channel analysis (SCA) to retrieve secret keys from cryptographic hardware [3,6,7,18,19,39, 40]. Side-channel attacks correlate side-channel information, such as the power consumption [e.g., 16,35] or electromagnetic emissions [e.g., 3,4,25], with intermediate values of the cryptographic computation, which are used as evidence to derive secret keys. While traditional methods, such as differential power analysis (DPA) [15] and correlation power analysis (CPA) [5], perform statistical analysis to uncover this correlation, learning-based attacks train a machine-learning model to do so [26]. Proficiency in identifying these correlations is critical to reduce the number of traces required for a successful attack. For defense, in

M. Dalla Preda et al. (Eds.): ARES 2025, LNCS 15993, pp. 321–342, 2025.
https://doi.org/10.1007/978-3-032-00627-1_16

turn, we strive to additionally identify the source of the leakage and fix it (e.g., avoid moving the mask and the masked value to the same register [3]).

Side-channel traces represent the executed instructions over time. Thus, it is possible to precisely link leakage occurrence in time to the executed instruction in the implementations [28,30]. However, doing so is particularly difficult for learning-based attacks. While the used machine-learning models hold unique knowledge about leakage that the hardware designer requires to protect against (or at least increase the difficulty of) learning-based attacks, they are mostly operated as black-box oracle and do not expose their internal "reasoning" easily.

"Explainable AI" (XAI) holds great promise in this setting as it allows to reveal what the machine-learning model "looks at" during a successful attack, and thus, can point out where information leaks. Prior work [7,12,22,24,38,40] has explored transferring XAI techniques from the image domain to side-channel analysis, such as "Simple Gradients" [31], "Layer-wise Relevance Propagation" [2], or "Weight Visualization" [36,40]. We find that these methods fall short of accurately locating leakage points in protected (masked) implementations. The reason is two-fold: First, unlike in the image domain, neither instance-based explanations [2,31], that explain a single input trace, nor model-based explanations [36,40], that explain the underlying model using its parameters and/or activations, are directly applicable in side-channel analysis because the machine-learning models used here operate on comparably low confidence [25]. SCA-based attacks, hence, accumulate evidence gathered from intermediate values across multiple power traces. Second, applying this logic of accumulating results to explanations must not solely focus on the correct key [12] but all classes. As it is well known in the SCA community, all classes of all traces contribute to the ranking of each key candidate. Similarly, all classes of all traces contribute to the final explanation, and thus, all classes need to be considered for model analysis (without pre-filtering by key). We find that the success of such *class-focused aggregation of explanations* is, however, dependent on using class-discriminative explanations—a type of explanation method that can attribute feature-relevance specific to a single class and suppress the influence from other classes [11,17]. Most explanation methods [2,31,41] used in SCA so far [7,12,24,40] are specific to the predicted class but are *not* class-discriminative.

Contribution. In this paper, we revisit the use of XAI techniques in the field of side-channel analysis and provide new insights in how to use them for learning-based profiling attacks. We point out the limits of key-focused explanations (that filter explanations based on the correct key) for locating leakage points in protected/masked hardware implementations. Additionally, we propose a novel approach for locating leakage, LeaX, that uses class-focused explanations and is suitable for the particularities of side-channel analysis. We extensively evaluate our explanation strategy on the community benchmarks ASCAD [3], AES_HD [25], and DPA_v4 [4], showing that LeaX significantly outperforms instance-based and key-focused explanations across different learning-based SCA attacks.

2 Side-Channel Analysis

In this paper, *we focus on the classical setting where side-channel analysis is used to derive encryption keys from cryptographic hardware* [5,8,15]. Although the encryption scheme is provably secure, it is possible to derive the used key via information leaked during the hardware's computations. In this setting, the side-channel commonly is the hardware's power consumption, referred to as power traces. We specifically consider learning-based profiling attacks [3,6,7,19,40], meaning, attacks that learn a machine learning model on power traces observed during profiling a cloned device [26].

We elaborate on the considered attacker model and provide more details on learning-based attacks in Sects. 2.1 and 2.2, respectively. Moreover, side-channel attacks can also serve to locate leakage points, that is, the specific time samples that allow to derive the secret key. This information is of paramount importance for hardening the implementation, as we explain in Sects. 2.3.

2.1 Attacker Model

Side-channel attacks operate on cryptographic hardware, implementing a specific, known cryptographic function for which side-channel information can be gathered. We consider power consumption as the side-channel measured over time, yielding power traces. Moreover, the encryption key is not hardcoded in the device but is configurable, and the attacker cannot read the key from the device's memory.

In addition to the *"target device,"* the adversary is assumed to have an identical cryptographic device, the *"clone device,"* that can be reconfigured at will and unlimitedly queried. It hence is possible to gather side-channel information, such as power traces, for different configurations and keys as a preparatory step for the attack, giving rise to so-called profiling attacks. In contrast to non-profiling attacks [e.g., 5,15], profiling attacks better cope with defense strategies such as "masking." We refer the reader to related survey [26,27] for more details.

For the actual attack against the target device *for which the key is not known*, the adversary collects side-channel information to derive a ranking of possible key candidates. The more evidence the adversary collects (number of power traces), the higher the correct (actually used) key rises in the derived ranking and thus, can be identified. The likelihood of success can be estimated using the "guessing entropy," which in turn is estimated through the mentioned key rank. Below, we sketch the principle and again refer the reader to related work [21,33] for details.

2.2 Learning-Based Attacks

In recent years, learning-based attacks have yielded promising results in SCA [18,39,40]. Thereby, a machine learning model represented as its parameters θ and a decision function $f_\theta : \mathbb{R}^d \rightarrow [0,1]^{256}$, takes a single power trace $\mathbf{x}$ consisting out of d time samples as its input and predicts the probabilities scores for abstract representations of a cryptographic intermediate value used as label $y = \varphi(z)$.

An *intermediate value* z can result from any intermediate computational step, for instance, $z = S(p_i \oplus k_i)$, where S refers to an S-box. Its representation, in turn, can be thought as a map φ, implemented as either the hamming weight [24], least significant bit (LSB) [34], or simply the identity function [3,25,40]. In the remainder of the paper, we consider the latter as it is the most effective among the three with the least loss of information [3].

Note that a thoroughly trained model can still *not* predict the intermediate value perfectly but provides imprecise clues only. Hence, a successful learning-based profiling attack accumulates prediction results across multiple inputs. *In this setting, the above observation on the indirection from intermediate value z to key byte k_i is crucial.*

Depending on the plaintext byte p_i, a different intermediate value z represents the correct key byte k_i^* in the model's output. Hence, when cumulating results over a set of inputs $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ it is imperative to resolve this indirection per input trace $\mathbf{x}$ individually.

For the remainder of the paper, we abstract away details of the encryption scheme (e.g., the use of S-Boxes) and introduce a plaintext-dependent map ϕ, using a message $m_{\mathbf{x}}$ that represents the used plaintext of a specific input trace $\mathbf{x}$, so that $z = \phi(k_i\,; m_{\mathbf{x}})$. Next, we sum up the yield log-probabilities over all traces in X to measure the attack performance, forming a single experiment's "distinguishing vector":

$$\mathbf{v}_X = \sum_{\mathbf{x} \in X} \left(\log f_\theta(\mathbf{x})_{\phi(k_i\,; m_{\mathbf{x}})} \right)_{k_i \in \{0,\dots,255\}} , \tag{1}$$

where each element of the vector refers to the likelihood of a specific key byte k_i to be correct. The map ϕ links keys (associated with the vector's dimensions) with the classifier's predictions of intermediate values.

Finally, we approximate the guessing entropy and thus the success of a learning-based SCA attack after processing n traces in X (the attack budget) across N experiments as: $\mathit{Rank}(k_i^*, \mathbf{v}_X) = \frac{1}{N} \sum_{l=1}^{N} \mathit{rank}(k_i^*, {\mathbf{v}_X}^{(l)})$. For evaluation, one commonly shows the average rank on the y-axis over the number of traces in X used on the x-axis. Hence, the "faster" an attack approaches $\mathit{Rank} = 0$ (meaning the fewer input traces are needed to reach an approximated guessing entropy of 0), the better the attack.

2.3 Locating Leakage

Side-channel traces represent the executed instructions over time, and thus, it is possible to pinpoint the instruction responsible for the observed effect in the trace. Consequently, knowing the precise time samples where leakage occurs allows for identifying problematic instructions or constructs in the implementation [28,30]. These indicative time samples of a power trace that correspond to information leakage are commonly referred to as "leakage points" or "points of interest" [26].

Locating leakage thus is of particular interest to the analyst, striving to improve the concealment of information in hardware. A large strain of research on side-channel analysis hence attempts to precisely locate leakage points [12, 22, 28, 30]. Our paper also falls in this category, presenting methods to locate leakage based on learning-based profiling attacks.

Most straightforwardly, the analyst can use the signal-to-noise ratio (SNR) to point out leakage locations [20]. *However, such an analysis requires knowledge about the used encryption key and the masking scheme (including the involved random values) that is deployed as a defense.* In learning-based profiling attacks, the attacker has access to a clone device and thus, controls the key during profiling. The masking scheme, however, is obstructed in the implementation and initialized through randomness so that an attacker or external analyst *cannot* compute the SNR directly [1, 37].

3 XAI for Side-Channel Analysis

Explainable AI (XAI) can show what a machine learning model "looks at" when deriving the correct key. An explanation method $h_\theta(\mathbf{x}, c)$ provides a *prediction-specific* relevance vector $\mathbf{R} \in \mathbb{R}^d$ of the same dimensionality as the input trace $\mathbf{x}$ based on the model θ. This means that each input feature x_i receives a relevance value R_i, showing how important that feature is to class c. Consequently, these methods can indicate leakage points, and thus, the problematic instructions to be adjust in the implementation to prevent the leak [28, 30]. *However, prior work on using XAI methods for leakage localization [7, 12, 22, 24, 38, 40] had limited success in correctly pointing out points-of-interest in protected (masked) implementations as we show in our evaluation (cf. Fig. 2).*

Informative time samples including leakage are sparsely distributed. Hence, an explanation method needs to highlight specific time samples accurately rather than rough regions. Figure 1 provides an example of an ideal attribution $\mathbf{e}$ using XAI (orange) with the SNR-over-time as ground truth showing the masking's random value r and the masked S-box output depicted in red and gray color.

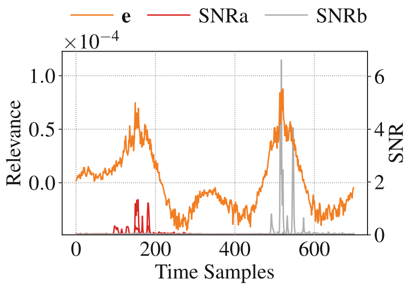


Fig. 1. Locating leakage points using XAI (orange) for a recent learning-based attack [12] on the ASCAD dataset [3]. Ground truth is provided in red and gray color based on the signal-to-noise ratio (SNR). *Note that information about the SNR is not available in practice but is provided by the benchmark.* (Color figure online)

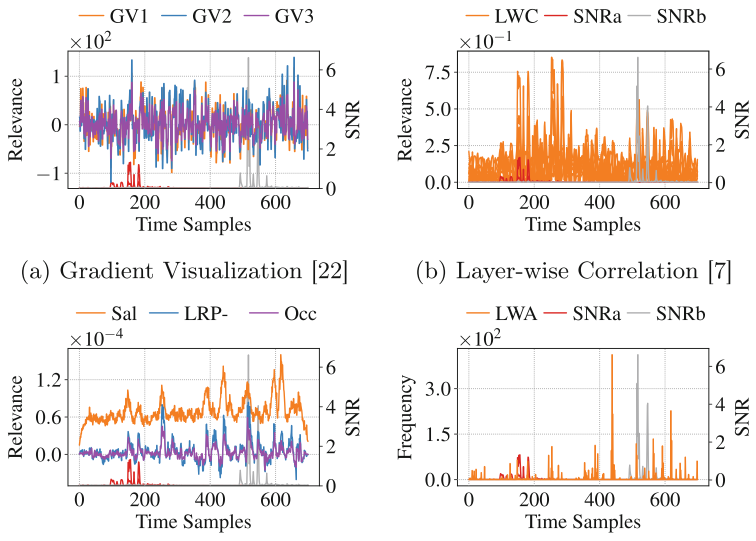
In further follow, we describe common strategies found in literature and highlight shortcomings of existing attempts to use XAI in side-channel analysis.

Explanation Strategies. In the past, various explanation methods and strategies have been used for locating leakage points [7,12,22,24,38,40], split into two rough categories: intrinsically explainable machine-learning models and "post-hoc explanations" of ML models.

While the former promises a tight coupling of explanations to the reasoning of the learning-based attack, these approaches are currently limited in handling low-dimensional inputs with as little as twelve time samples only [38]. The large majority [7,12,24,40], however, uses post-hoc explanations for locating leakage, which is also in line with the XAI community where most approaches fall into this category [2,29,31,41]. We distinguish three explanation strategies for SCA:

A. **Instance-based explanations.** Most straight-forwardly one takes a single input (a power trace) and uses XAI to highlight what the machine-learning model considers to be important for deriving the encryption key [22]. In the image domain, instance-based explanations are used with great success [2,29,41]. For side-channel analysis, however, a single input carries too little information to derive the correct key with certainty [25], and thus, the explanation based on one individual power trace cannot locate the true leakage as we demonstrate. As an example, Fig. 2a shows three traces explained using Gradient Visualization [22], exhibiting drastic variance from one to another.
B. **Model-based explanations.** In SCA, we strive to know what the model in its entirety has learned about all the individual power traces that contain little clues about the occurring leakage. Model-based explanations [7,40] promise to provide this insight so that their application in SCA feels somewhat natural. Unfortunately, the models in learning-based attacks provide predictions with comparably low confidence [25], and hence, model-based explanations replicate this low confidence, failing to satisfactorily locate leakage. For instance, Fig. 2b shows the "Layer-wise Correlation" method [7] indicating areas where no sensitive information/leakage exists.
C. **Aggregation-based explanations.** In line with the learning-based attack's common practice of cumulating predictions to derive the correct encryption key, another strain of research proposes to cumulate explanations to locate leakage [12,24]. This middle way uses instance-based explanations to extract model knowledge as striven for in model-based explanations. Such an explanation strategy is unique to the field of side-channel analysis and is not employed in the image domain. Figure 2c and 2d provide two examples: First, attribution-based explanations [12] fail to indicate clear positive or negative regions of relevance in the trace, and also the "Layer-wise Activation" method [24] equally struggles to highlight "points of interests."

Unfortunately, none of those directions has yielded satisfactory results for masked implementations so far, leaving the community with effective key extraction but without the means to use these tools to locate leakage.



(a) Gradient Visualization [22]

(b) Layer-wise Correlation [7]

(c) Attribution Methods [12]

(d) Layer-wise Activation [24]

Fig. 2. Explanations of pior work: (a) Instance-based "Gradient Visualization" [22], (b) model-based "Layer-wise Correlation" [7], and two aggregation-based techniques using (c) different attribution methods [12] including salicency maps ("Sal"), LRP-ϵ, and Occlusion ("Occ"), and (d) Layer-wise Activation [24].

4 Revisiting Aggregation-Based Explanations

In this section, we specifically look at aggregation-based explanations that cumulate relevance values across multiple input traces similar to how attacks in SCA are typically conducted [3]. We detail key-focused explanations as presented in related work [12] and derive our proposal of *class-focused explanations* that better resembles the community's knowledge of SCA-based attacks.

Key-Focused Explanations. Hettwer et al. [12] propose to average explanations across all inputs $\mathbf{x} \in X$ whilst considering the mapping from the correct key byte k_i^* to the respective intermediate value $z = \varphi(\phi(k_i^*\,; m_{\mathbf{x}}))$. However, we assume $\varphi(.)$ to be the identity map in further follow, so that $z = \phi(k_i^*\,; m_{\mathbf{x}})$. An instance-based, prediction-specific explanation method $h_\theta(\mathbf{x}, z)$ then derives the time samples' contribution to the decision for value z:

$$\mathbf{e}_{k_i^*} = \frac{1}{|X|} \sum_{\mathbf{x} \in X} h_\theta(\mathbf{x}, \phi(k_i^* \,; m_\mathbf{x})). \tag{2}$$

This derivation, unfortunately, does not allow us to conclusively determine the correct leakage points in masked implementations, as we show in our evaluation. However, it gives rise to connecting explanation strategies with the common attack strategy in SCA. In line with the computation of the key ranking to approximate the guessing entropy, this approach can be extended to finding the correct averaged explanation based on label-to-key mapping ϕ among all key candidates $k_i \in K = \{0, \ldots, 255\}$:

$$E_K = \left\{ \frac{1}{|X|} \sum_{\mathbf{x} \in X} h_\theta(\mathbf{x}, \phi(k_i \,; m_\mathbf{x})) \mid k_i \in K \right\}. \tag{3}$$

Note that $\mathbf{e}_{k_i^*} \in E_K$ because k_i^* is one (of multiple) key hypothesis.

Class-Focused Explanations. We propose an alternative explanation strategy for side-channel analysis: Rather than focusing too rigidly on the label-to-key mapping, ϕ, it instead is better to explain the neural network's prediction, $f_\theta(\mathbf{x})_c$ for the specific classes $c \in C$, irrespective of the assigned key byte $k_i = \phi^{-1}(c \,; m_\mathbf{x})$. This rationale follows the intuition that all predicted classes (intermediate values) of all traces contribute to the ranking of the individual key candidates [26]. However, as we are not interested in the final key but in the model's insights/explanation, we consider all predicted classes of all traces *without* pre-filtering by keys. There are two crucial observations to make:

1. *While the classifier might be wrong overall, the individual locations it "looks at" (i.e., the corresponding features) can still be correct.*
2. *Predictions are influenced by features from the correct class and any other (wrong) class. Hence, indicative features can be obstructed by wrong features.*

A class-discriminative explanation method $h'_\theta(\mathbf{x}, c)$ that subtracts the influence of classes not matching the actual prediction [11] can highlight these indicative features. Cumulating over multiple inputs $\mathbf{x} \in X$ then gathers all the class-discriminative clues/features that the model is aware of, no matter how small:

$$E_C = \left\{ \frac{1}{|X|} \sum_{\mathbf{x} \in X} h'_\theta(\mathbf{x}, c) \mid c \in C \right\} . \tag{4}$$

Moreover, without relying on the correct key byte k_i^* each class $c \in C$ seems equally likely, so that we require a method for finding the most indicative explanation $\mathbf{e} \in E_C$. Formally, we denote this method as $H_\theta(X)$ and describe a novel approach in Sect. 5.

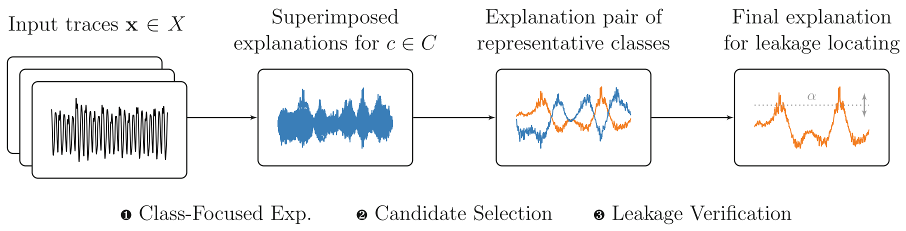

Input traces $\mathbf{x} \in X$
Superimposed explanations for $c \in C$
Explanation pair of representative classes
Final explanation for leakage locating
α
❶ Class-Focused Exp.
❷ Candidate Selection
❸ Leakage Verification

Fig. 3. Schematic depiction of LeaX: We ❶ employ our class-focused explanation strategy, ❷ select two maximally contradicting groups of candidates, and ❸ verify which of those describe leakage by analyzing different cut-off thresholds α.

5 LeaX: Locating Leakage Using XAI

Our method LeaX determines the most indicative explanation $\mathbf{e} = H_\theta(X)$ that shows leakage points ("points of interest"), thus implementing the function H_θ on an attack dataset X. Often, there is not a single explanation, though, but a group of similar explanations that stand out so that we average across the most indicative candidates from E_C in practice.

Locating leakage with out method, LeaX, involves three steps as depicted in Fig. 3:

❶ We determine the class-focused explanations for all possible classes $c \in C$ as described in Sect. 4. We do so by averaging across an attack dataset of power traces X gathered from the clone device (Sect. 5.1).

❷ Interestingly, these (aggregated) class-focused explanations occur in two opposing groups with varying intensity. We choose the explanations that differ the most, yielding the most promising "explanation candidates" (Sect. 5.2).

❸ These candidates have low/high relevance values for the correct leakage points already. However, we cannot tell which one of the two is correct, so that we verify leakage by incrementally removing least to most relevant time samples (according to the explanations) and inspect the progression of attack success. Only the candidate yielding more effective attacks across increasing α indicates the correct leakage points (Sect. 5.3).

5.1 Class-Focused Explanations

While the reasoning for class-focused explanations is based on difficile details of side-channel analysis (cf. Sect. 4), the computation itself is rather straightforward. However, the specific choice of the used explanation method h_θ is crucial, though: We require so-called class-discriminative explanation methods as we strive to focus on the involved classes (and their individual prediction scores). Interestingly, most explanation methods are *not* class-discriminative.

Methods such as Simple Gradients [31], SmoothGrad [32], or "Layer-wise Relevance Propagation" (LRP) [2,23] indicate features relevant to the decision *but* do not focus on features distinguishing it from other classes [11]. Consequently, these are unsuitable for our use case. There exist class-discriminative extensions to some of the mentioned methods, though, e.g., "Softmax Gradient LRP" (SGLRP) [14] or "Contrastive Layer-wise Relevance Propagation" (CLRP) [11]. For LeaX, we use the latter and provide further details below.

In our evaluation (cf. Sect. 6.3), however, we show results for both variants (LeaX with CLRP and SGLRP) to shed light on the differences in practice.

Contrastive Layer-Wise Relevance Propagation (CLRP). As the name suggests, CLRP [11] builds upon LRP [2,23], which is founded on the so-called conservation property, stating that the relevance of all the units of a neural network's layer l need to sum up to the relevance values of all the units in the next layer $l+1$ for all L layers of the network:

$$\sum_i R_i^{(1)} = \sum_i R_i^{(2)} = \ldots = \sum_i R_i^{(L)}.$$

Here, $R_i^{(l)}$ denotes the relevance of unit i in layer l. LRP determines relevance for all classes, which unfortunately results in low sensitivity to the predicted class [11]. To highlight features for the target class, CLRP introduces negative values to non-target classes and conducts multiple runs of the above calculation. CLRP initialize the relevance values of the last layer as follows:

$$R_i^{(L)} := \begin{cases} g_\theta(\mathbf{x})_c & i = c \\ -\frac{g_\theta(\mathbf{x})_c}{|C|-1} & \text{otherwise}, \end{cases}$$

where $g_\theta(.)$ represents the logits of the neural network before any softmax operation. The currently investigated class is denoted as c, and $|C|$ refers to the total number of classes. Note that only the relevance for the particular class c is set to the model's output, and all remaining values of classes $c' \neq c$ are set so that the overall sum of relevance values is zero:

$$R_c^{(L)} + \sum_{c' \neq c} R_{c'}^{(L)} = 0.$$

This construction guarantees that features indicative for class c' are highlighted rather than other classes' features. Moreover, similarly to LRP, positive values mean that the features speak in favor of the classification of class c, while negative values (from the wrong classes' "penalty values") speak against it. Eventually, we will use positively valued input features for determining leakage points with LeaX as these speak in favor of the classification.

5.2 Candidate Selection

Due to the use of class-discriminative explanations and the aggregation of input traces, class-focused explanations$\mathbf{e} \in E_C$ exhibit some interesting properties:

(1) Positive values that speak in favor of the classification and negative values that speak against the classification cancel each other out when averaged across classes $c \in C$. (2) Positive and negative values are spread out over the entire input range due to the noise involved in learning-based SCA attacks. Considering positive values only and averaging these, hence, is not helpful. Consequently, we have to indeed select one/some of the classes rather than average across E_C. (3) Class-focused explanations form opposing clusters with varying intensity, that is, we have multiple pairs of c_1 and c_2, so that $\mathbf{e}_{c_1} \approx -\mathbf{e}_{c_2}$. In other words, some explanations show very high while other explanations show very low relevance values for the same time samples.

The third and last property is crucial for identifying the most indicative class (representative) and its class-focused explanation, respectively. We calculate the pair-wise Euclidean distance for all explanations $\mathbf{e}_1, \mathbf{e}_2 \in E_C$ to find the opposing (groups of) explanations. We then choose the pair, P_{max}, with the largest distance. However, it may happen that doing so captures stark outliers that are not good candidates. We hence determine the l-nearest neighbors[1] [9] to form two groups/neighborhoods of explanations, E_{C1} and E_{C2}, that show similar explanation patterns. Finally, we use the center class of E_{C1} and E_{C2} as our explanation candidates $\mathbf{e}_a$ and $\mathbf{e}_b$, respectively, for the final step.

5.3 Leakage Verification

At this stage, the candidates $\mathbf{e}_a$ and $\mathbf{e}_b$ are equally likely, and we cannot decide based on the mere explanation output which one shows true leakage points. We, thus, take recourse to established techniques for evaluating XAI methods [13] and adapt them to class-focused explanations.

We follow the intuition that a learning-based attack will not succeed if we remove time samples that leak information from the considered input traces. Put differently, if we remove time samples that do *not* leak information, the attack still works. However, in learning-based SCA attacks, we may have multiple leakage points (e.g., repetitive leakage [4] or combined leakage [3]) that may serve as an indicator for the used key. Even worse, the machine-learning model may or may not consider this redundancy. Hence, the mere removal of certain time samples is insufficient.

We propose to take/keep presumably indicative (regions of) time samples according to a "take ratio" α while rejecting the rest and retrain an attack model on this data. We refer to this process as "Take and Retrain" (TART) that we describe below in more detail. Based on this, we then demonstrate how to identify the correct explanation candidate by observing attack performance over increasing values of α.

Take and Retrain (TART). We construct two datasets $X_{\mathbf{e}_a}$ and $X_{\mathbf{e}_b}$ based on the explanation candidates $\mathbf{e}_a$ and $\mathbf{e}_b$, and train separated models θ_a and θ_b on

[1] Note that we use the well-known k-nearest neighbors algorithm [9] but avoid using k to prevent confusion with the encryption key as used before.

these datasets. More specifically, we filter out time samples with low relevance according to a “take ratio” α yielding $T_{\mathbf{e}} = \{i \mid e_i > \alpha \cdot \max(\mathbf{e})\}$ for an explanation $\mathbf{e}$. This set of highly relevant time samples serves as a reference to judge which time samples (i.e., input features) to take/keep in the new datasets:

$$X_{\mathbf{e}} = \left\{ (x_i)_{i \in T_{\mathbf{e}}} \mid \mathbf{x} \in X \right\}.$$

For each candidate $\mathbf{e}_a$ and $\mathbf{e}_b$, we construct individual training datasets, $X^{(train)}_{\mathbf{e}_a}$ and $X^{(train)}_{\mathbf{e}_b}$, and attack datasets $X_{\mathbf{e}_a}$ and $X_{\mathbf{e}_b}$, respectively. While the former is used for training, the latter we use to measure the attack performance of the new models θ_a and θ_b. The model architecture remains identical except for the input size. Comparing the performances of these models then allows us to pick the candidate that shows the leakage points used by the original model.

Note that retraining on these datasets is crucial for the success of TART, and the original model must not be used to classify the modified data. Without retraining, it is impossible to tell whether the dataset modification itself causes a change in performance or it stems from the removed input features [13]. In other words, retraining allows the model to adapt to the changed dataset distribution.

Assuming $\mathbf{e}_a$ as the more suitable explanation, we have three possible outcomes of the final comparison of models:

1. One model performs a successful attack while the other does not:

$$Rank(k_i^*, \mathbf{v}_{X_{\mathbf{e}_a}}) = 0 \ \wedge \ Rank(k_i^*, \mathbf{v}_{X_{\mathbf{e}_b}}) \neq 0,$$

   where $\mathbf{v}_{X_{\mathbf{e}_a}}$ and $\mathbf{v}_{X_{\mathbf{e}_b}}$ are the “distinguishing vectors” of the respective models, and *Rank* determines the averaged rank of the correct key byte k_i^* as discussed in cf. 2. A successful attack is indicated by a rank of 0, meaning that the model has learned the leakage correctly as revealed by the explanation candidate.
2. Machine-learning models do not necessarily capture redundancy if a subset suffices for successful prediction. In our scenario, a model may derive the correct key using one of multiple leakage points. This circumstance may lead to the situation where both models are successful:

$$Rank(k_i^*, \mathbf{v}_{X_{\mathbf{e}_a}}) = 0 \ \wedge \ Rank(k_i^*, \mathbf{v}_{X_{\mathbf{e}_b}}) = 0,$$

   which means that both explanations capture leakage points. Here, we are in favor of the explanation enabling faster rank convergences to 0.
3. Finally, it may happen that both models fail to derive the correct key:

$$Rank(k_i^*, \mathbf{v}_{X_{\mathbf{e}_a}}) \neq 0 \ \wedge \ Rank(k_i^*, \mathbf{v}_{X_{\mathbf{e}_b}}) \neq 0.$$

   In this case, either the underlying explanation method cannot show the real leakage or the take ratio α is too restrictive/high.

Verification. For verifying explanation candidates, we repeat TART for different take ratios $\alpha \in [0, 1]$ applied to $\mathbf{e}_a$ and $\mathbf{e}_b$. We measure the average number of traces required for a successful attack (normalized to the attack budget n) indicated on the y-axis per take ratio on the x-axis and determine the area under this curve denoted as TART-AUC$\mathbf{e}_a$ and TART-AUC$\mathbf{e}_b$, respectively. The lower the ..., the preciser the explanation—thus, the explanation candidate yielding the lower value indicates the correct leakage points.

Obviously, the ...can only be approximated in discrete steps for increasing the α in practice. In the remainder of the paper, we thus use steps of 0.1. Examples are provided in Figs. 4, 5 and 6. *Note that the figures in the remainder of the paper show the absolute number of traces used on the y-axis rather than the normalized value to better compare the attack success to related work.*

6 Evaluation

In this section, we evaluate LeaX across different model structures and leakage scenarios. We begin by introducing the used datasets in Sect. 6.1, before presenting our main results of LeaX under different leakage scenarios in Sect. 6.2. Finally, the ablation study in Sect. 6.3 investigates the choices of LeaX's major components and elaborates on the insufficiency of key-focused explanations.

6.1 Experimental Setup

Benchmarks used in the SCA community typically consider implementations of the "Advanced Encryption Standard" (AES) [10] to compare side-channel attack methods. In our experiments, we apply our method to attack models from related work that use (a) convolutional neural networks (CNNs) [12,40][2] and (b) simple multi-layer perceptrons (MLPs) [36]. We evaluate LeaX on three datasets covering different types of leakage. Note that each dataset has a specific MLP and a CNN model, and that all models applied to demonstrate the general applicability of LeaX have been proposed in related work.

Datasets. In our evaluation, we cover three leakage types from varying software and hardware implementations described below:

1. *Combined Leakage* means that information from multiple, disjunct time frames must be combined to infer sensitive data. The hardware implementation from the ASCAD dataset [3] uses first-order masking so that the adversary requires both the mask r and the masked S-box output, $S(m_{\mathbf{x}} \oplus k^*) \oplus r$, to obtain the correct key.

[2] The model architecture for AES_HD is inspired by Zaid et al. [40]. It has one convolutional layer with kernel size 1 and a pooling size of 25. The model is trained *without* (0,1)-scaling [36].

2. *Single Leakage* refers to a situation where one specific time frame reveals sensitive information. The AES_HD dataset [25] shows power traces of an FPGA-based, unprotected (i.e., no masking) implementation, where leakage occurs through writing a register in the last round of the AES decryption.
3. *Repetitive Leakage* occurs if multiple time frames leak identical sensitive information. The DPA Contest v4.2 (DPA_v4) dataset [4] has been recorded based on a masked software implementation of AES for a smart card. Here, the masking values are know, so that the community [e.g., 25,40] often treats it as unprotected implementation and reduces it to a subset of 4000 time samples per trace. Each input trace contains three groups of time samples that leak the S-box output in the first round of AES.

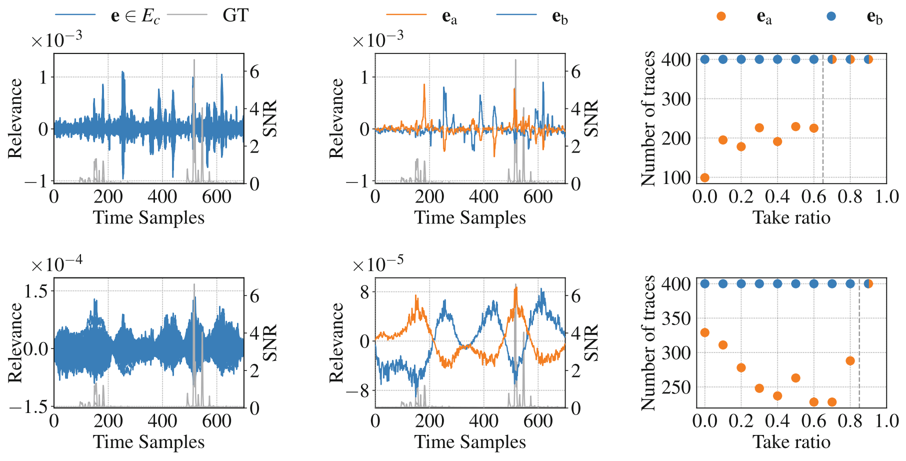
Among the public benchmark datasets, ASCAD dataset [3] is most commonly used for demonstrating machine learning-based side-channel attacks [3,7,18,25,39,40] and explanation techniques in this domain [12,22,24,38].

Learning Setup. We follow the common setup in deep-learning based SCA, where attack models are customized to each dataset. 90% of a training set (the "profiling dataset") is used for actual training and 10% for validation. Each benchmark dataset mentioned above additionally provides a test dataset to measure attack performance. The input traces are preprocessed to have zero mean and unit variance before training. For all datasets, we average the guessing entropy across 100 repetitions of the attack and shuffle the attack traces randomly for each experiment.

6.2 Explanation Performance

We apply LeaX to different MLP-based and CNN-based attack models [12,36,40] and use the same explanation parameterization/configuration across different datasets. More specifically, we use CLRP for ❶ class-focused explanation, set the number of nearest neighbors to 5 for ❷ candidate selection, and present explanation results indicated by ❸ leakage verification. The choice of these parameters is further explained in the ablation study in Sect. 6.3.

Combined Leakage. We show LeaX on the most commonly investigated dataset ASCAD [3] that records combined leakage where the mask and the masked output are required to infer sensitive data. We present results of LeaX on an MLP model [36] and CNN model [12] in Fig. 4 and discuss the details in the subsequent paragraphs.



(a) Class-Focused Exp. (b) Candidate Selection (c) Leakage Verification

Fig. 4. Our method using CLRP for ASCAD on the MLP model [36] (top row), and CNN model [12] (bottom row).

MLP. Figure 4a (top row) shows class-focused explanations using CLRP for each class, that is, for each of the 256 possible key bytes. Based on these, LeaX's candidate selection obtains the representative explanation pair $(\mathbf{e}_a, \mathbf{e}_b)$ as presented in Fig. 4b (top row), that show opposing relevance scores across the x-axis. Consequently, only one of them can be correct.

We apply leakage verification with different take ratios $\alpha \in [0, 1]$ and present the attack performances in Fig. 4c (top row). Orange and blue dots represent models trained on time samples positively indicated by $\mathbf{e}_a$ and $\mathbf{e}_b$, respectively. When the take ratio α increases to 0.7 to construct $X_{\mathbf{e}_a}$, the attack fails as we start to exclude time samples corresponding to the masking at time sample 180. Therefore, there is not sufficient information to learn the real leakage based on $\mathbf{e}_a$ anymore. In contrast, when taking time samples based on $\mathbf{e}_b$, the trained models fail consistently across take ratios (blue dots in Fig. 4c top row). We use the attack budget of 400 traces to indicate failure.

In summary, leakage verification yields a ...for $\mathbf{e}_a$ and $\mathbf{e}_b$ of 0.64 and 1.0, respectively, indicating that $\mathbf{e}_a$ is the correct explanation. The presence of two peaks (close to time samples 180 and 514) in $\mathbf{e}_a$ suggests that we are dealing with a combined leakage scenario, where two distinct sets of leakage information are required for a successful attack.

CNN. Again, we show all class-focused explanations and the distilled representative explanations $(\mathbf{e}_a, \mathbf{e}_b)$ in Figs. 4a and 4b (bottom row), respectively. We then apply leakage verification to reveal which explanation matches the leakage. The attack performance in Fig. 4c (bottom row) shows that taking time samples based on $\mathbf{e}_a$ allows us to reliably identify the correct key, whereas $\mathbf{e}_b$ does not lead to a successful attack. We yield a ...for $\mathbf{e}_a$ and $\mathbf{e}_b$ of 0.70 and 1.0, respectively, indicating that $\mathbf{e}_a$ is the correct explanation. Also, $\mathbf{e}_a$ nicely overlaps with the ground truth visually. Only when the take ratio α equals 0.9, the critical leak close to time sample 149 (i.e., the mask) is omitted. Therefore, the model is unable to learn the correlation between traces and target classes.

Single Leakage. Next, we present how LeaX locates a single leakage on the AES_HD dataset [25] in Fig. 5. We show results on a MLP-based [36] and a CNN-based attack model [40] in three steps as before.

MLP. We retrained attack models based on $\mathbf{e}_a$ and $\mathbf{e}_b$, as extracted from the 256 class-focused explanations shown Fig. 5a (top row), using different take ratios $\alpha \in [0, 1]$. In Fig. 5c (top row), we present the number of traces on the y-axis needed to perform a successful attack for different values of α on the x-axis. Here, TART-AUC($\mathbf{e}_a$) = 0.55 and TART-AUC($\mathbf{e}_b$) = 0.96, meaning that selecting time samples based $\mathbf{e}_a$ highlights the leaking samples best. Visually, there also is a high peak in $\mathbf{e}_a$ close to time sample 960 in accordance to the ground truth.

CNN. Fig. 5 (bottom row) shows the three steps of LeaX. During leakage verification, we obtain TART-AUC($\mathbf{e}_a$) = 0.52 and TART-AUC($\mathbf{e}_b$) = 1.0, and thus, $\mathbf{e}_a$ as the correct explanation. This is a similar difference in attack performance between $\mathbf{e}_a$ and $\mathbf{e}_b$ as in the MLP model (cf. Fig. 5 top row). The results on MLP and CNN models demonstrate LeaX can address a single leakage scenario.

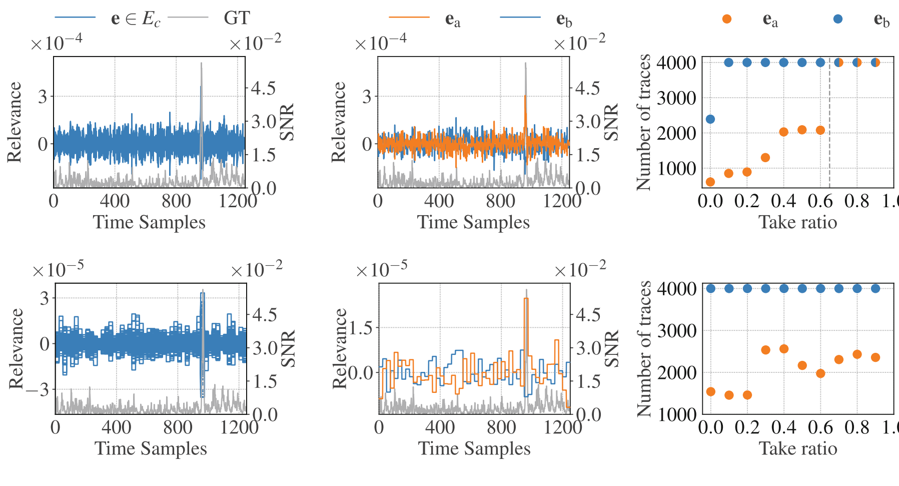


Fig. 5. LeaX using CLRP for AES_HD on the MLP model [36] (top row), and CNN model [40] (bottom row).

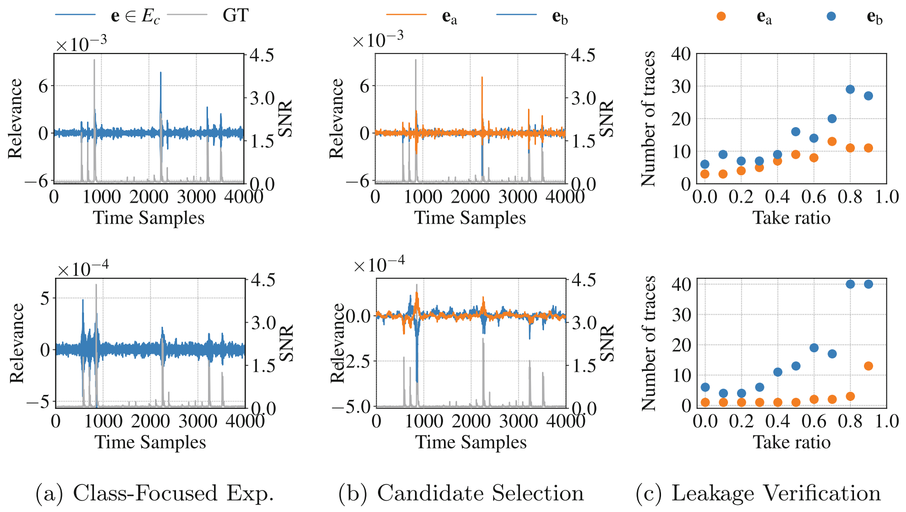


Fig. 6. LeaX using CLRP for DPA_v4 on the MLP model [36] (top row), and CNN model [12] (bottom row).

Repetitive Leakage. DPA_v4 dataset [4] has three groups of leakage representing the same information leak. We present the results of LeaX on an MLP model [36] and a CNN model [12] in Fig. 6.

MLP. After applying CLRP to all classes, we obtain a representative explanation pair $(\mathbf{e}_a, \mathbf{e}_b)$ as shown in Fig. 6b (top row). Leakage verification identifies $\mathbf{e}_a$ as the correct explanation, with a ...of 0.18. The explanation candidate $\mathbf{e}_b$, in turn, yields a ...of 0.36. As can be seen in Fig. 6c (top row), trimming time samples according to both explanations does not make the attack fail, indicating the presence of multiple leakages within the dataset.

CNN. Leakage verification for the selected candidates $\mathbf{e}_a$ and $\mathbf{e}_b$ (cf. Fig. 6b bottom row) yields TART-AUC$(\mathbf{e}_a) = 0.07$ and TART-AUC$(\mathbf{e}_b) = 0.40$, and thus, identifies $\mathbf{e}_a$ as the final explanation result. In Fig. 6c (bottom row), the model retrained on $X_{\mathbf{e}_a}$always performs better than $X_{\mathbf{e}_b}$, meaning the model learns the leakage mainly from time samples close to 850.

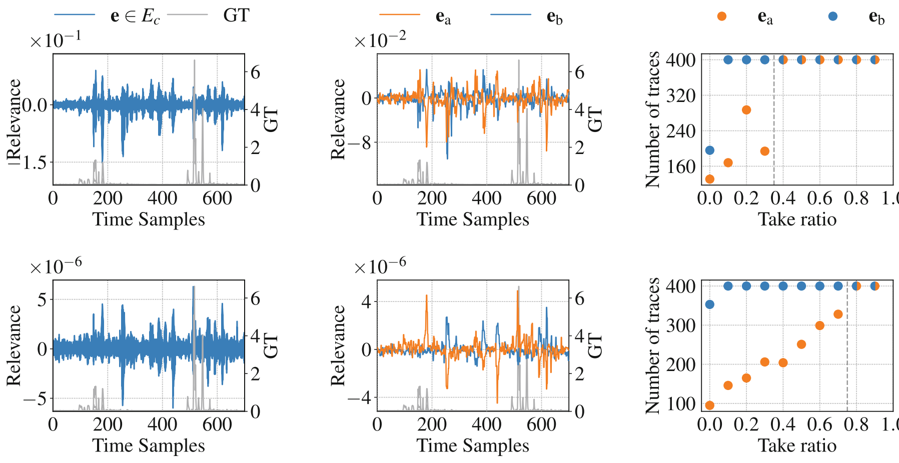
6.3 Ablation Study

We proceed to analyze each component of LeaX, focusing on the influence of class-discriminative explanation methods and the cluster size used for candidate selection. Finally, we also discuss the insufficiency of key-focused explanations.

Class-Discriminative Explanations. To clarify why class-discriminative methods are crucial to class-focused explanations, we apply class-discriminative methods and not class-discriminative methods to explain all classes on the benchmark dataset ASCAD. Additionally, we investigate LEAX when using a method that is not class-discriminative, Grad × Input, and another class-discriminative method, SGLRP, and present the results in Fig. 7.

When using Grad × Input, LEAX yields a ...of 0.80 and 0.95 for $\mathbf{e}_a$ and $\mathbf{e}_b$, respectively. The former thus is selected as the correct explanation. However, both ...values are relatively high, implying that both explanations are subpar. As shown in Fig. 7c (top row), the attack based on $\mathbf{e}_a$ fails with a take ratio of 0.4 already. Also visually comparing with the ground truth (cf. Fig. 7b top row) reveals that neither explanation candidate highlights the masked output at time sample 514.

For SGLRP, in contrast, leakage verification results in ...values of 0.62 and 0.99 for $\mathbf{e}_a$ and $\mathbf{e}_b$, respectively, indicating $\mathbf{e}_a$ as the correct explanation. Explanations based on SGLRP are clearly more indicative: The TART-AUC($\mathbf{e}_a$) is significantly lower when using SGLRP than it was with Grad × Input. In Fig. 7b (bottom row), $\mathbf{e}_a$ again highly overlaps with the ground truth.



(a) Class-Focused Exp. (b) Candidate Selection (c) Leakage Verification

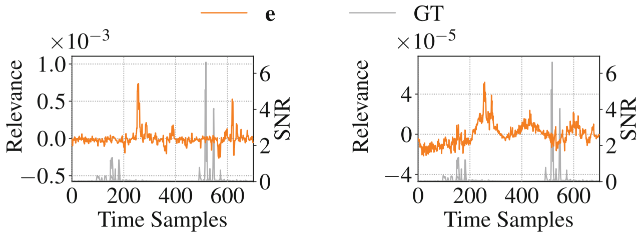
Fig. 7. LEAX using Grad × Input (top row) and SGLRP (bottom row) on the MLP model [36] for ASCAD dataset.

Size of Candidate Clusters. Assuming that high positive values in the explanation indicate high classification confidence, we select explanations of high relevance to indicate leakage time samples. Given the limited size of only 256 classes, exhaustively searching for suitable candidates emerged as a suitable options. Our candidate selection seeks to identify class pairs with the highest Euclidean distance P_{max} as potential representatives. Since explanations across different classes are similar, we use simple nearest neighbors clustering to find similar explanation groups based on P_{max}.

While the number of clusters is fixed to two (to find two opposing groups of explanations), we perform a grid search for the best neighborhood size in $[5, 10, 20, 40, 60, 80, 100, 120, 140]$. Interestingly, all these values perform reasonably well for locating leakage when selecting the center explanation. Averaging across classes, however, is not. The reason is that the larger the neighborhood gets, the more low-relevance explanations are included, flattening any explanation peaks present. In our experiments, we thus fix the neighborhood size to 5.

Insufficiency of Key-Focused Explanations. With the successful use of class-discriminative explanations, a natural thought is, if we can improve key-focused explanations using class-discriminative explanation also. We present two case studies: (1) using CLRP in key-focused explanations and (2) using LeaX's candidate selection based on key-focused explanations.

Case Study 1. We incorporate CLRP into the key-focused explanation strategy. Specifically, we choose the class corresponding to the correct key for each trace and aggregate the relevance scores to obtain $\mathbf{e}_{k_i^*}$ as described in Eq. (2). Figure 8 shows the key-focused explanations on MLP and CNN models retrieved this way. Both fail to indicate the correct leakage as compared to the ground truth. The explanation of the MLP points to non-leakage time samples $[200, 300]$ and $[600, 700]$, the one for the CNN highlights time samples $[200, 300]$ and $[400, 500]$.



(a) MLP model [36] (b) CNN model [12]

Fig. 8. Key explanations using CLRP on ASCAD with the correct key.

Case Study 2. Finally, we generate key-focused explanations for all 256 key hypotheses (i.e., $\mathbf{e}_{k_i} \in E_K$) and apply candidate selection to observe how well the explanation candidates align with the ground truth. Again, the key-focused explanation strategy fails to highlight leakaging time samples on MLP and CNN models. Compared to class-focused explanations, they simply are not as indicative. Key-focused explanations cannot help locate leakage even when equipped with class-discriminative explanations or LeaX's candidate selection.

7 Conclusion

Existing approaches for locating leakage points using learning-based attacks fall short in precision. We analyze the reasons and revisit the use of XAI techniques for this task. Two aspects are crucial: First, naively applying instance-based or model-based explanations is prone to fail as the accuracy of the SCA attack models is low in comparison to other classification tasks where XAI is applied. Second, accumulating explanations just like we do for attacks to compensate for low classification accuracy, in turn, must focus on the model's classes (the intermediate values) rather than the key, giving rise to *class-focused explanations*. The feasibility of class-focused explanations, however, is fundamentally dependent on class-discriminative explanations (e.g., CLRP). We have developed a thorough understanding of the relations of SCA attacks and traditional XAI techniques, effectively bridging disciplines. The resulting technique, LeaX, allows to precisely locate leakage, improving the state of the art decisively.

Acknowledgments. We gratefully acknowledge funding by the Helmholtz Association (HGF) within topic "46.23 Engineering Secure Systems."

Data Availability Statement. For the sake of reproducibility, we make the implementations of our method, LeaX, publicly available at:https://intellisec.de/research/leax.

References

1. Azouaoui, M., et al.: A systematic appraisal of side channel evaluation strategies. In: Proceedings of the International Conference on Security Standardisation Research (SSR) (2020)
2. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS ONE **10**(7), 1–46 (2015)
3. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ASCAD database. JCEN **10**(2), 163–188 (2020)
4. Bhasin, S., Bruneau, N., Danger, J., Guilley, S., Najm, Z.: Analysis and improvements of the DPA contest v4 implementation. In: Proceedings of the International Conference on Security, Privacy, and Applied Cryptography Engineering (SPACE) (2014)
5. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES) (2004)
6. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without preprocessing. In: Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES) (2017)
7. Cao, P., Zhang, C., Lu, X., Gu, D., Xu, S.: Improving deep learning based second-order side-channel analysis with bilinear CNN. IEEE Trans. Inf. Forensics Secur. **17**, 3863–3876 (2022)

8. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES) (2002)
9. Cove, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
10. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography, Springer, Berlin, Heidelberg (2002). https://doi.org/10.1007/978-3-662-60769-5
11. Gu, J., Yang, Y., Tresp, V.: Understanding individual decisions of CNNs via contrastive backpropagation. In: Proceedings of the IEEE/CVF Asian Conference on Computer Vision (ACCV) (2018)
12. Hettwer, B., Gehrer, S., Güneysu, T.: Deep neural network attribution methods for leakage analysis and symmetric key recovery. In: Proceedings of the International Conference on Selected Areas in Cryptography (SAC) (2019)
13. Hooker, S., Erhan, D., Kindermans, P., Kim, B.: A benchmark for interpretability methods in deep neural networks. In: Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS) (2019)
14. Iwana, B.K., Kuroki, R., Uchida, S.: Explaining convolutional neural networks using softmax gradient layer-wise relevance propagation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops (2019)
15. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Proceedings of the Annual International Cryptology Conference (CRYPTO) (1999)
16. Kwon, D., Hong, S., Kim, H.: Optimizing implementations of non-profiled deep learning-based side-channel attacks. IEEE Access **10**, 5957–5967 (2022)
17. Lee, J.R., Kim, S., Park, I., Eo, T., Hwang, D.: Relevance-CAM: your model already knows where to look. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
18. Lu, X., Zhang, C., Cao, P., Gu, D., Lu, H.: Pay attention to raw traces: a deep learning architecture for end-to-end profiling attacks. IACR TCHES **3**, 235–274 (2021)
19. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: Proceedings of the International Conference on Security, Privacy, and Applied Cryptography Engineering (SPACE) (2016)
20. Mangard, S.: Hardware countermeasures against DPA – a statistical analysis of their effectiveness. In: Proceedings of the Cryptographers Track at RSA Conference (CT-RSA) (2004)
21. Massey, J.L.: Guessing and entropy. In: IEEE International Symposium on Information Theory (ISIT) (1994)
22. Masure, L., Dumas, C., Prouff, E.: Gradient visualization for general characterization in profiling attacks. In: Proceedings of the International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE) (2019)
23. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.: Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recogn. **65**, 211–222 (2017)
24. Perin, G., Ege, B., Chmielewski, L.: Neural network model assessment for side-channel analysis. IACR Cryptology ePrint Archive, p. 722 (2019)
25. Picek, S., Heuser, A., Jovic, A., Bhasin, S., Regazzoni, F.: The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. IACR TCHES **2019**(1), 209–237 (2019)

26. Picek, S., Perin, G., Mariot, L., Wu, L., Batina, L.: SoK: deep learning-based physical side-channel analysis. ACM CSUR **55**, 1–35 (2023)
27. Prouff, E., Rivain, M.: Masking against side-channel attacks: a formal security proof. In: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2013)
28. Pundir, N., Park, J., Farahmandi, F., Tehranipoor, M.M.: Power side-channel leakage assessment framework at register-transfer level. IEEE Trans. VLSI **30**(9), 1207–1218 (2022)
29. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2017)
30. Shelton, M.A., Samwel, N., Batina, L., Regazzoni, F., Wagner, M., Yarom, Y.: Rosita: towards automatic elimination of power-analysis leakage in ciphers. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) (2021)
31. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: visualising image classification models and saliency maps. In: Proceedings of the International Conference on Learning Representations (ICLR) Workshop Track Proceedings (2014)
32. Smilkov, D., Thorat, N., Kim, B., Viégas, F.B., Wattenberg, M.: SmoothGrad: removing noise by adding noise. arXiv preprint arXiv:1706.03825 (2017)
33. Standaert, F., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT) (2009)
34. Timon, B.: Non-profiled deep learning-based side-channel attacks with sensitivity analysis. IACR TCHES **2019**(2), 107–131 (2019)
35. Wei, L., Luo, B., Li, Y., Liu, Y., Xu, Q.: I know what you see: power side-channel attack on convolutional neural network accelerators. In: Proceedings of the Annual Computer Security Applications Conference (ACSAC) (2018)
36. Wouters, L., Arribas, V., Gierlichs, B., Preneel, B.: Revisiting a methodology for efficient CNN architectures in profiling attacks. IACR TCHES **2020**(3), 147–168 (2020)
37. Wu, L., Perin, G., Picek, S.: Not so difficult in the end: breaking the lookup table-based affine masking scheme. In: Proceedings of the ACM Symposium on Applied Computing (SAC), vol. 14201 (2023)
38. Yap, T., Benamira, A., Bhasin, S., Peyrin, T.: Peek into the black-box: interpretable neural network using SAT equations in side-channel analysis. IACR TCHES **2023**(2), 24–53 (2023)
39. Zaid, G., Bossuet, L., Carbone, M., Habrard, A., Venelli, A.: Conditional variational autoencoder based on stochastic attacks. IACR TCHES **2023**(2), 310–357 (2023)
40. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient CNN architectures in profiling attacks. IACR TCHES **2020**(1), 1–36 (2020)
41. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Proceedings of the European Conference on Computer Vision (ECCV) (2014)