Uncertainty Estimation in Neural Network-enabled Side-channel Analysis and Links to Explainability

Seyedmohammad Nouraniboosjin and Fatemeh Ganji

Worcester Polytechnic Institute, Worcester, USA
snouraniboosjin@wpi.edu,fganji@wpi.edu

Abstract. Side-channel analysis (SCA) has emerged as a critical field in securing hardware implementations against potential vulnerabilities. With the advent of artificial intelligence (AI), neural network-based approaches have proven to be among the most useful techniques for profiled SCA. Despite the success of NN-assisted SCA, a critical challenge remains, namely understanding how confident NNs are in their predictions. NNs often exhibit uncertainty in their predictions, leading to incorrect key guesses with high probabilities, corresponding to a higher rank associated with the correct key. This uncertainty stems from multiple factors, including measurement errors, randomness in physical quantities, and variability in NN training. Understanding whether this uncertainty arises from inherent data characteristics or can be mitigated through better training is crucial. Additionally, if data uncertainty dominates, identifying specific trace features responsible for misclassification becomes essential.

We propose a novel approach to estimating uncertainty in NN-based SCA by leveraging Rényi entropy, which offers a generalized framework for capturing various forms of uncertainty. This metric allows us to quantify uncertainty in NN predictions and explain its impact on key recovery. We decompose uncertainty into epistemic (model-related) and aleatoric (data-related) components. Given the challenge of estimating probability distributions in high-dimensional spaces, we use matrix-based Rényi α-entropy and α-divergence to better approximate leakage distributions, addressing the limitations of KL divergence in SCA. We also explore the sources of uncertainty, e.g., desynchronization, randomized keys, as well as hyperparameters related to NN training. To identify which time instances (features in traces) contribute most to uncertainty, we also integrate Shapley value-based explanations with our framework, overcoming the limitations of conventional sensitivity analysis. Lastly, we show that predictive uncertainty strongly correlates with standard SCA metrics like rank, offering a complementary measure for evaluating attack complexity. Our theoretical findings are backed by extensive experiments on available datasets and NN models.

Keywords: NN-assisted SCA. · Uncertainty. · Metrics.

1 Introduction

Assessing the security of an implementation against side-channel analysis (SCA) is a complex task. Since their introduction in the 1990s [Koc96, KJJ99], extensive research has been dedicated to analyzing physical leakage in the form of time, power, electromagnetic emanation, etc. This line of research focuses on evaluating mechanisms based on a solid

The code is available at https://github.com/vernamlab/SCA_Uncertainty_Decomposition

theoretical foundation. A key advancement toward establishing rigorous, information-theoretic security has been the introduction of profiled SCA, which uses an open sample to estimate the device's leakage model [CRR02]. Profiled SCA represents the worst-case security scenario for an implementation, as there is no universal attack strategy capable of extracting secret information from physical observables without prior knowledge of the device's leakage distribution [SMY09, WOS14, MCHS23]. [HRG14] has established that an optimal attack strategy from an information-theoretic perspective relies on distinguishing the correct key with regard to the probability distribution of leakage conditioned on the targeted secret. Establishing such an optimal distinguisher is known to be highly non-trivial due to the unknown true leakage distribution of a device that is complex to even estimate, especially in the presence of countermeasures like masking [CJRR99]. In fact, if one can perfectly learn the probability density function (pdf) of the leakage conditioned on the targeted secret, it can be further involved in a maximum likelihood distinguisher to obtain the optimal solution cf. [MDP20]. In the absence of knowledge about this pdf, supervised classification using deep neural networks (NNs) has emerged as a sub-optimal solution.

Despite major breakthroughs and increases in the effectiveness of NN-enabled attacks, obstacles should be faced to link NNs' concepts widely adopted in machine learning and SCA. One of the already known missing links in this regard is the link between performance metrics in both domains. While key rank, guessing entropy, and key discrimination success rate [SMY09] have been used as metrics in SCA, accuracy serves as the primary metric in machine learning. The lack of a direct equivalent for accuracy in SCA severely limits not only the evaluation of SCA effectiveness cf. [BPS+18, PHJ+19, MDP20], but also hinders interpretability and error attribution in NN-based attacks. This limits our ability to *explain* why an NN makes a correct/wrong key guess.

In fact, regardless of their applications, NN classifiers often assign a high prediction probability to an incorrect label, i.e., NNs are overconfident in their predictions [TB24, GTA+23]. This means that even when a model produces a high predictive probability at the softmax output, it does not necessarily reflect its true level of uncertainty. In the context of SCA, the uncertainty can be translated to wrong guesses with high probabilities delivered by the NN (i.e., the correct key has a high rank instead of being ranked first). While losses such as L_2 or NLL penalize prediction errors, they do not measure the uncertainty in predictions. SCA-specific metrics, unfortunately, fail to provide a clear assessment of the model's predictive uncertainty.

The root cause of uncertainty in NN-enabled SCA can be measurement errors, randomness in physical quantities, and errors in the NN's architecture specification or its training. Calibration issues, limitations in precision, as well as human errors during measurement, account for measurement errors. Randomness in physical quantities encompasses thermal noise, manufacturing, or environmental variations and can influence the consistency of patterns in traces, making the traces less exploitable [DCEM18, MHK+24, GOKT16]. On the other hand, the training process of an NN involves numerous parameters, e.g., batch size, learning rate, stopping criteria, and regularization. Other stochastic elements like batch generation and weight initialization introduce more randomness during training [LBH15]. These directly lead to a strong factor for uncertainties in NNs' predictions.

The key question to address here is whether it is possible to *explain* why an NN's key guess is incorrect. In other words, one seeks to understand whether the uncertainty stems from the inherent characteristics of the data or if it can be mitigated through improved NN training. The second question that arises is: as data characteristics contribute to the predictive uncertainty in NN-assisted SCA, can one identify which trace features led to ambiguity in guessing the correct key?

Contribution. Our paper positively answers these questions by making the following contributions. We stress that we do not treat the concepts borrowed from the machine learning field as simple, "plug-and-play" solutions; instead, we adapt and tailor them to

meet the specific needs of SCA (see Section 3.4). Through extensive experimentation, we validate our proposed uncertainty estimation framework and its implications for NN-based SCA, as detailed below.

1. We introduced the predictive uncertainty metric quantified by looking at the entropy of the classification output conditioned on the input traces. In the context of SCA with high-dimensional data, the precise probability density function (PDF) estimation required to approximate the entropy is infeasible. Therefore, we employ the matrix-based Rényi α-entropy. Moreover, as the side-channel leakage can follow a multimodal and/or multivariate distribution [LPMS18, LM18, RBAF24, MMOS16, BGH+17, GLRP06, RBFA21, DCGRP19, Riv08, SMY09], conventional measures for approximating probability distributions cannot be employed. After discussing why one of the most popular measures, i.e., Kullback-Leibler divergence, fails to approximate the leakage distribution, we propose using α-divergence [Ama12, ZR95] to approximate the PDFs and then compute the predictive uncertainty. In doing so, we also examine the relationship between uncertainty and standard SCA evaluation metrics, such as the average rank. Our findings reveal a strong relationship between uncertainty levels and rank. In machine learning, higher predictive uncertainty is often associated with a flatter output distribution over classes [GG16], where the correct class (correct key in SCA) may still receive non-negligible probability, but not rank at the top.

2. The second contribution of our paper is related to how explainability links to predictive uncertainty in SCA. Here, explainability refers to the ability to identify and attribute the causes of model mispredictions in NN–based SCA. This enables analysts to move beyond surface-level accuracy metrics and instead perform systematic diagnosis of failure cases, such as recognizing overfitting, poor generalization, or data ambiguity. Our goal is to turn uncertainty into actionable insight, providing root-cause explanations for why a model fails on certain traces, rather than relying on anecdotal examples or post-hoc interpretations. As a prime example, we evaluate the impact of desynchronization and key randomization on model uncertainty. Using datasets that introduced such characteristics, we show that they not only reduce attack performance, but also increase uncertainty in model predictions. We further take into account hyperparameters closely related to the NN training, including the number of epochs, the number of training/evaluation traces, and the network configuration. In this regard, we highlight the fact that the uncertainty can explain whether better hyperparameters could improve the attack. Such an explanation helps the evaluators understand whether the uncertainty is reducible, i.e., can be resolved by, e.g., feeding more traces to the NN. Hence, the uncertainty metric directly connects the NN performance to the attack cost and effectiveness.

3. Our third contribution deals with which time instance(s) in a trace cause uncertainty. We first motivate the problem by noting that traditional uncertainty sensitivity analysis methods (e.g., [DHLUR17]) assume independence across input dimensions, making them unsuitable for side-channel traces, where strong temporal correlations exist. In fact, consecutive samples of a trace are statistically dependent [CRR02, BCM+23], so the independence assumption behind derivative-based methods remains invalid in SCA settings. Therefore, we conduct an in-depth analysis of time sample contributions using SHapley Additive exPlanations (SHAP) [LL17] to identify the most critical time features influencing model uncertainty. Given that side-channel traces contain thousands of time features, not all contribute equally to classification tasks. By integrating SHAP values with uncertainty estimations, we determine which time samples are most responsible for the uncertainty in model predictions, providing valuable insights for optimizing SCA models and improving attack efficacy.

1.1 Organization of the Paper

The remainder of this paper is organized as follows. Section 2 provides a detailed background on profiled SCA and its evaluation metrics. In this section we also introduce uncertainty quantification techniques, SHAP values and the datasets that we used for our experiments. Section 3 details our methodology for uncertainty analysis in NN-enabled SCA. Section 4 presents extensive experimental results on benchmark datasets, focusing on uncertainty analysis under various conditions, such as desynchronization, calibration or different training setups. Section 5 concludes the paper by summarizing our findings, discussing their implications for SCA, and suggesting directions for future research.

2 Background

2.1 Notations

In this paper, we adopt calligraphic letters such as $\mathcal{X}$ to represent sets. Correspondingly, uppercase letters, such as X, denote random variables (random vectors are represented as $\mathbf{X}$), while lowercase letters like x (or $\mathbf{x}$ for vectors) indicate specific realizations of these random variables. Matrices are distinguished using bold capital letters. The i-th component of a vector $\mathbf{x}$ is expressed as $\mathbf{x}[i]$, whereas the i-th observation of a random variable X is written as x_i. The pdf of a continuous random variable X is denoted as f_X, where $f_X(x)$ denotes the probability density at x. The expected value is represented by $\mathbb{E}[\cdot]$, which may be sub-scripted by a random variable, as in $\mathbb{E}_X[\cdot]$, or by a probability distribution, written as $\mathbb{E}_{f_X}[\cdot]$, to specify the context in which the expectation is computed. Formulation of Shannon's conditional entropy is $H(Y \mid X) = H(Y, X) - H(X)$, whereas mutual information (MI) and conditional mutual information (CMI) are $I(X;Y) = H(X) + H(Y) - H(X,Y)$ and $I(X;Y|X') = H(X,X') + H(Y,X') - H(X,Y,X') - H(X')$. Here, H denotes entropy or joint entropy: $H = -\sum_{x \in X} p(x) \log p(x)$ and $p(x)$ denotes the probability of random variable X taking value x.

2.2 Datasets

This paper uses two versions of the ASCAD dataset, which serve as benchmarks for evaluating the performance of NN-based SCA models.

ASCAD Fixed (ASCAD-f). The ASCAD fixed dataset targets an 8-bit AVR microcontroller running a masked AES-128 implementation. Electromagnetic (EM) emanations are recorded as side-channel traces. The dataset consists of a profiling set with 50,000 traces and an attack set with 10,000 traces. Each trace contains 700 features extracted from a fixed time window, and the profiling and attack sets use the same encryption key. The attack focuses on recovering the third key byte, which corresponds to the first masked intermediate value. This dataset also has traces with parametrized desynchronization to check the efficiency of the algorithm against jitter, here we also use traces with 50 and 100 desynchronization [BPS+20].

ASCAD Random (ASCAD-r). The ASCAD random dataset is an extension of the ASCAD family, featuring random encryption keys for the profiling phase. This dataset contains a profiling set with 200,000 traces recorded with random keys and an attack set with 100,000 traces using a fixed key. The attack still focuses on the third key byte, but the traces contain a larger window of 1,400 features to account for increased variability due to random keys. ASCAD-r is more challenging than ASCAD-f due to the presence of key randomness, which increases noise and reduces the effectiveness of simple leakage models.

Both ASCAD-f and ASCAD-r provide a controlled environment for evaluating NN-based profiled attacks. The use of masked AES-128 ensures that these datasets are representative

of real-world cryptographic implementations [BPS+20].

CHES CTF 2018. The database refers to the CHES Capture-the-flag (CTF) AES-128 trace set, which was released in 2018 for the Conference on Cryptographic Hardware and Embedded Systems (CHES) [Ris18]. The traces are derived from masked AES-128 encryption executed on a 32-bit STM microcontroller. In our experiments, we utilize 45,000 traces for the training set, which incorporates a fixed key. The test set comprises 5,000 traces, with the key used in the training set being different from the key configured for the test set. Additionally, every trace contains 2,200 features [CHE18].

2.3 Profiled Side-Channel Analysis

SCA exploits the observations $\mathbf{L}$ obtained during the operation of an algorithm to extract its secret [BPS+18]. When an SCA adversary gains control over an open device (i.e., a device where she can manipulate at least some inputs, including the secret), they can execute a specific class of attacks known as profiling attacks. A profiling SCA consists of two primary phases: a profiling phase and an attack phase [BPS+18]. For each possible key $k \in \mathcal{K}$, the attacker computes an estimated probability distribution function $\hat{g}_k$ as follows:

$$g_k : (\boldsymbol{\ell}, p) \mapsto p(\mathbf{L} = \boldsymbol{\ell} \mid (P, K) = (p, k)). \tag{1}$$

The estimated function $\hat{g}_k$ is obtained from a profiling set $\mathcal{D}_{\text{profiling}} \doteq \{(\boldsymbol{\ell}_i, p_i, k_i)\}_{i=1}^{N_p}$, where N_p represents the number of traces ℓ_i. These traces are collected under known plaintext chunks p_i and key values k_i. Throughout this paper, the trace set is denoted by $\mathcal{L}$, while the corresponding set of inputs is referred to as $\mathcal{X}$.

In the attack phase, the attacker collects an attack set from a closed device, denoted as $\mathcal{D}_{\text{attack}} \doteq \{(\boldsymbol{\ell}_i, p_i)\}_{i=1}^{N_a}$, where the unknown secret, say k^*, remains fixed but undisclosed. The attacker's goal is to recover this key. For traces in $\mathcal{D}_{\text{attack}}$, a score based on the maximum-likelihood of each hypothetical key can be obtained as

$$\mathbf{d}_{Na}[k] = \prod_{i=1}^{N_a} \Pr((P, K) = (p_i, k) \mid \mathbf{L} = \boldsymbol{\ell}_i) = \prod_{i=1}^{N_a} \frac{\Pr(\mathbf{L} = \boldsymbol{\ell}_i \mid (P, K) = (p_i, k)) f_{\mathbf{L}}(\boldsymbol{\ell}_i)}{f_{P,K}(p_i, k)}. \tag{2}$$

Here, $\Pr(\mathbf{L} \mid P, K)$ denotes the likelihood estimated from the generative model built during the profiling phase, and the posterior $\Pr((P, K) \mid \mathbf{L})$ is obtained via Bayes' theorem.

Metrics for SCA. In the realm of SCA, model performance is often assessed using the rank function. Let $k^* \in \mathcal{K}$ denote the true key used during $\mathcal{D}_{\text{profiling}}$ acquisition. The rank function of a model $\hat{g}$ trained on $\mathcal{D}_{\text{profiling}}$ and tested on $\mathcal{D}_{\text{attack}}$ is:

$$\text{rank}(\hat{g}, \mathcal{D}_{\text{profiling}}, \mathcal{D}_{\text{attack}}, n) = |\{k \in \mathcal{K} \mid \mathbf{d}_{Na}[k] > \mathbf{d}_{Na}[k^*]\}|, \tag{3}$$

where $\mathbf{d}_{Na}[k]$ represents the score for candidate k, as defined in Equation 2. The score is derived from the conditional probability model built on $\mathcal{D}_{\text{profiling}}$ and evaluated on the first n traces in $\mathcal{D}_{\text{attack}}$. If k^* achieves the highest score, the rank is 0; if it has the lowest, the rank is $|\mathcal{K}| - 1$. Since the rank varies based on the traces used, it is common practice to calculate it across different dataset chunks and determine the average rank, also known as guessing entropy [MPP16].

2.4 Matrix-based Estimation of Rényi Entropy

When using a generalized version of Shannon entropy, specifically Rényi entropy, information-theoretic quantities can be computed through functionals applied to normalized positive definite matrices. More concretely, Rényi entropy is computed from the normalized Gram matrix $\mathbf{A}$ obtained by evaluating a kernel on all data pairs. A kernel is a function

$\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that takes two inputs from a space $\mathcal{X}$ and returns a real number representing their similarity. It is symmetric, i.e., $\kappa(x, x') = \kappa(x', x)$, and positive definite, meaning that for any finite set $x_1, \ldots, x_n \subset \mathcal{X}$, the Gram matrix $K \in \mathbb{R}^{n \times n}$ with entries $K_{ij} = \kappa(x_i, x_j)$ satisfies $c^\top K c \geq 0$ for all $c \in \mathbb{R}^n$.

A kernel is called infinitely divisible if for every real number $r > 0$, the function $\kappa_r(x, y) = \kappa(x, y)^r$ is still a valid positive definite kernel [Bha06, YP19]. This property guarantees closure under Hadamard (element-wise) powers and products, which is essential for defining multivariate entropy over multiple variables using kernels.

Let $\mathbf{A} = \mathbf{K}/tr(\mathbf{K})$ denote the normalized Gram matrix, where $tr(\cdot)$ is the trace of a square matrix, i.e., the sum of its diagonal elements. The eigenvalues of $\mathbf{A}$ are used in the functional[1] $\mathbf{S}_\alpha(\mathbf{A})$ to compute Rényi entropy, thus avoiding explicit density estimation. When $\alpha = 1.01$, this closely approximates Shannon entropy [YP19, GRP14, YWJP20]. These matrices are implicitly constructed in a reproducing kernel Hilbert space (RKHS), where the positive definiteness of the kernel ensures that all eigenvalues are non-negative, making entropy and similarity computations well-defined and numerically stable. This formulation allows information-theoretic measures to be computed directly from data samples without assuming a parametric distribution. This procedure is formulated in the following definition.

Definition 1. *(cf. [YP19]) Consider n samples $\{x_1^i, x_2^i, \cdots, x_k^i\}_{i=1}^n$, where each sample contains k ($k \geq 2$) measurements, we define the kernel $\kappa_1 : \mathcal{X}_1 \times \mathcal{X}_1 \mapsto \mathbb{R}$, $\cdots$, $\kappa_k : \mathcal{X}_k \times \mathcal{X}_k \mapsto \mathbb{R}$ ($X_z = x_z^1, \cdots, x_z^n$ with $1 \leq z \leq k$).*

The Rényi's α-order joint-entropy for k variables is

$$\mathbf{J}_\alpha(\mathbf{X}_1, \cdots, \mathbf{X}_k) = \mathbf{S}_\alpha \left(\frac{\mathbf{A}_1 \odot \cdots \odot \mathbf{A}_k}{tr(\mathbf{A}_1 \odot \cdots \odot \mathbf{A}_k)} \right), \tag{4}$$

Where $(\mathbf{A}_1)ij = \kappa_1(x_1^i, x_1^j)$, $\cdots$, $(\mathbf{A}_k)ij = \kappa_k(x_k^i, x_k^j)$. Here, $\odot$ denotes the Hadamard product operators, Furthermore, the function $\mathbf{S}_\alpha(\cdot)$ is defined as follows.

$$\mathbf{S}_\alpha(\mathbf{A}) = \frac{1}{1-\alpha} \log_2(tr(\mathbf{A}^\alpha)) = \frac{1}{1-\alpha} \log_2 \sum_{i=1}^{n} \lambda_i(\mathbf{A}^\alpha), \tag{5}$$

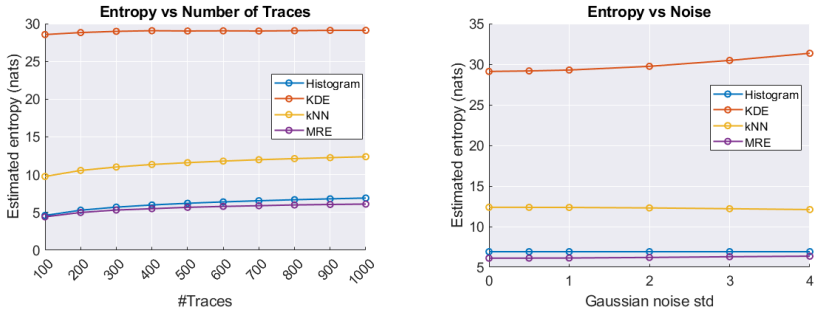
where $\lambda_i(\mathbf{A})$ denotes the ith eigenvalue of $\mathbf{A}$.

The relationship between the matrix $\mathbf{A}$ in Equation (5) and the Gram matrix $\mathbf{K}$ can be formulated as $\mathbf{A}_{ij} = \frac{\mathbf{K}_{ij}}{n\sqrt{\mathbf{K}_{ii}\mathbf{K}_{jj}}}$. In order to approximate Shannon entropy, the information quantities are computed by setting $\alpha = 1.01$ in the matrix-based Rényi entropy formula [GP13, GRP14].

Empirical comparison between Matrix-based Rényi entropy (MRE) and off-the-shelf entropy estimators. We compare widely used entropy estimation techniques, histogram [Pan03], kernel density estimation (KDE) [Sil18], and k-nearest neighbors (kNN) [Koz87], against matrix-based Rényi entropy (MRE). Histogram-based methods discretize the input space and compute empirical symbol probabilities. KDE smooths the distribution via kernel functions to estimate a continuous density, while kNN relies on local distances among samples to estimate entropy.

To mitigate the curse of dimensionality, which affects KDE, kNN, and histograms due to sparsity and bandwidth sensitivity, we reduce the dimensionality of the ASCAD-f side-channel traces to 10 using principal component analysis (PCA) [GFA+21]. MRE, by contrast, operates on Gram matrices computed from pairwise similarities and is naturally suited to high-dimensional inputs, avoiding explicit density estimation. We emphasize that PCA is *not* ideal for side-channel leakage preservation [HB10]. This is due to the fact that

[1] Here, a functional is a function that takes a matrix as input and returns a scalar.



(a) (b)

Figure 1: Entropy estimates obtained using histogram [Pan03], KDE [Sil18], kNN [Koz87], and matrix-based Rényi entropy (MRE) on real side channel traces of ASCAD dataset [BPS+18] projected to 10 dimensions via PCA. Entropy is shown as a function of sample size (a) and additive Gaussian noise level (b), illustrating estimator behavior under finite-sample and noisy conditions.

side-channel traces often contain overlapping leakage, i.e., multiple operations contribute to the same region of the trace. PCA produces linear combinations of all features, which affect temporal locality and can merge or obscure distinct leakage points. Figure 1(a) shows entropy as a function of sample size. MRE provides stable estimates even for small numbers of traces and tracks, and histogram-based values more closely than KDE or kNN. In Figure 1(b), we fix the sample size to 1000 traces and add Gaussian noise to assess estimator robustness. MRE remains the least sensitive to increasing noise, maintaining entropy estimates comparable to the histogram baseline. These results highlight MRE's suitability for noisy, finite-sample settings typical in side-channel analysis.

2.5 SHAP Values

SHAP (SHapley Additive exPlanations) values provide an interpretable method to understand the contributions of individual features in a machine learning model. Based on cooperative game theory, SHAP fairly attributes the prediction of a model to its input features by considering their marginal contributions across all possible feature subsets [SREG+25].

Definition 2. *For a model $f : \mathcal{X} \to \mathbb{R}$ and a set of input feature indices $\mathcal{N}$, the SHAP value ϕ_i for feature $i \in \mathcal{N}$ is defined as:*

$$\phi_i = \sum_{S \subseteq \mathcal{N} \setminus \{i\}} \frac{|S|! \cdot (|\mathcal{N}| - |S| - 1)!}{|\mathcal{N}|!} \left[f_{(S \cup \{i\})} - f_S \right], \tag{6}$$

where S represents a subset of feature indices. f_S denotes the expected model output when only the features in S are known and the others are marginalized or replaced by reference values, i.e., $f_S = \mathbb{E}[f(x_S, x_{\bar{S}})]$. Thus, $f_{(S \cup \{i\})}$ represents the model output when feature i is added to the subset S.

Choice of explainer. Although SHAP provides fair and theoretically grounded feature attributions, this interpretability comes with computational cost. To address this, SHAP includes model-specific explainers optimized for different model structures. Among those, we employ Kernel SHAP, which is well-suited for non-parametric settings [LL17]. Unlike tree- or deep-learning-specific SHAP variants, Kernel SHAP builds a locally linear surrogate

model using weighted linear regression over multiple perturbed inputs to approximate feature attributions.

Advantages and limitations of SHAP. In the landscape of post-hoc explanation methods, techniques generally fall into three broad categories: perturbation-based, gradient-based, and example-based [SF24]. SHAP belongs to the perturbation-based class, offering additive and locally accurate attributions by quantifying each feature's marginal contribution to a scalar output. This makes SHAP particularly suitable for our framework, where the goal is to explain predictive uncertainty rather than a specific class label. In contrast, gradient-based methods such as saliency maps, integrated gradients, or Grad-CAM rely on output sensitivity with respect to inputs and are typically used to explain a predicted class [NDR+22]. While computationally efficient, these methods often lack properties such as consistency or faithfulness. Example-based methods, e.g., influence functions, explain predictions by referencing similar training examples but are neither easily quantitative nor decomposable, which limits their usefulness for uncertainty analysis [BPF20].

More specifically, SHAP has been chosen for its model-agnostic nature and theoretical grounding, making it suitable for explaining predictive uncertainty. One of SHAP's key strengths is its foundation in cooperative game theory, ensuring it uniquely satisfies three essential properties. First, SHAP ensures local accuracy, meaning that the sum of SHAP values across time samples equals the uncertainty score for that input trace. Furthermore, missingness (missing features receive zero importance), and consistency (more influential features never get less attribution) are offered by SHAP [LL17]. This makes SHAP unique among post-hoc explanation methods in satisfying all three features (see Section 3.5 for a comparison with gradient-based methods). SHAP also generalizes and unifies various explanation techniques such as [RSG16] and DeepLIFT [SGK17] under a single formal framework [LL17], providing theoretical rigor and interpretability coherence across models and contexts. Moreover, SHAP can capture interactions between features, such as groups of time samples in a trace, rather than evaluating each feature in isolation. This provides a more comprehensive understanding of how different parts of the trace jointly influence the model's behavior in SCA.

Yet, SHAP does have limitations. [SHJ+20] has showed that perturbation-based explainers like SHAP and LIME [RSG16] can be fooled by adversarial classifiers crafted to yield misleading explanations. However, in SCA, evaluators themselves train the models for interpretability and analysis, rather than relying on third-party black-box models. This ensures alignment between model behavior and explanation goals. Moreover, certain practical considerations arise depending on the application context. For instance, Kernel SHAP assumes local linearity when approximating the model around an input, which generally works well in practice, but may be less accurate for highly non-linear models with sharp decision boundaries. In cases where the model exhibits strong hierarchical or non-local dependencies, e.g., deeper convolutional networks, using model-specific explainers such as Deep SHAP may provide fidelity.

3 Uncertainty Analysis in SCA

Our main objective in this section is to study the uncertainty associated with the predictions of NNs in SCA. To achieve this, we start with formulation of NN-enabled SCA to understand sources of uncertainty.

3.1 NN-enabled Profiled SCA

As explained in Section 2.3, the attacker aims to disclose the secret k^*, where she must determine which of the estimated probability distributions $\hat{g}_k$ (for each $k \in \mathcal{K}$) best fits the attack set. Given reasonable assumptions about the distributions, the most efficient

way to determine this is through a maximum likelihood strategy. This strategy involves computing the following likelihood function $\mathbf{d}_{Na}[k]$ for each key candidate $k \in \mathcal{K}$ and selecting the most probable candidate; see Equation 2. Estimating $\Pr(\mathbf{L} \mid (P, K) = (p, k))$ for a given pair $(p, k) \in \mathcal{Y}$ with $\mathcal{Y} \doteq \mathcal{P} \times \mathcal{K}$ is recognized as a prediction problem (also referred to as a generation problem). Conversely, estimating $p((P, K) = (p, k) \mid \mathbf{L})$ is known as a classification problem. Bayes' theorem provides means to relate the simpler classification problem to the more complex generative one, making the two problems closely interconnected [BPS+18, LTR17]. This connection is important because, in practice, NN classifiers estimate posterior probabilities, but are used in a generative manner by accumulating likelihoods across traces.

NNs have found application in SCA since they provide tools for handling classification problems, enabling a discriminative approach [LTR17]. This means that the classification problem is directly addressed by estimating $\Pr((P, K) = (p, k) \mid \mathbf{L})$ without relying on Bayes' inversion [BN06]. This methodology aims at constructing an approximation $\hat{f}_{\mathbf{L},P}$ of the function $f_{\mathbf{L},P} : (\ell, p) \mapsto \Pr((P, K) = (p, k) \mid \mathbf{L} = \boldsymbol{\ell})_{k \in \mathcal{K}}$. The classification of a new trace $\boldsymbol{\ell}$ observed for an input p is carried out by computing $\mathbf{y} = \hat{g}_{\mathbf{L},P}(\boldsymbol{\ell}, p)$ and selecting the key candidate $\hat{k}$ (or equivalently, the corresponding label in ML terminology) as: $\hat{k} = \arg\max_{k \in \mathcal{K}} \mathbf{y}[k]$. When key discrimination is conducted over N_a pairs $(\boldsymbol{\ell}_i, p_i)$, the maximum likelihood approach is applied as follows:

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

where $\mathbf{y}_i$ represents the output of the function $\hat{f}_{\mathbf{L},P}$ when evaluated with the pair $(\boldsymbol{\ell}_i, p_i)$. Using the same notation, we have:

$$\mathrm{acc}(\hat{g}, \mathcal{D}_{\mathrm{train}}, \mathcal{D}_{\mathrm{test}}) = \frac{|\{(\boldsymbol{\ell}_i, p_i, k^*) \in \mathcal{D}_{\mathrm{test}} \mid k^* = \arg\max_{k \in \mathcal{K}} \mathbf{y}_i[k]\}|}{|\mathcal{D}_{\mathrm{test}}|}, \tag{8}$$

where $\mathbf{y}_i$ is the output vector from $\hat{g}(\boldsymbol{\ell}_i, p_i)$ cf. [BPS+18]. Note that this definition assumes a fixed key byte k^* across all test traces, as is common in SCA benchmarks. While this definition resembles standard classification accuracy, it is more accurately described as a rank-1 key prediction rate, measuring how often the correct key byte k^* receives the highest score from the model. This single-trace metric correlates with key recovery performance, as higher values typically lead to faster reduction in key rank under multi-trace attacks (cf. Equation 7).

Notably, Equation 7 is a straightforward reformulation of Equation 2, where the conditional probability $\Pr((P, K) = (p_i, k) \mid \mathbf{L} = \boldsymbol{\ell}_i)$ is replaced by its approximation $\mathbf{y}_i[k] = \hat{f}_{\mathbf{L},P}(\boldsymbol{\ell}_i, p_i)[k]$ cf. [BPS+18]. This approximation of $\hat{g}_{\mathbf{L},P}$ can be performed by NNs from a family of functions (often referred to as models in ML terminology) whose types are usually specified a priori by the data analyst to suit the problem's specific characteristics. Selecting the appropriate type of NNs in SCA is the first step, where the first attempts have been devoted to combining convolutional NNs (CNNs) with data augmentation in order to enhance attack performance, even against protected designs [BPS+18, PYW+17, WAGP20, ZBHV20]. Later, studies have shown similar performance, and sometimes even superiority, when using multi-layer perceptrons (MLPs) [WPP22, AGF23]. Regardless of NN types, the approximated function $\hat{g}_{\mathbf{L},P}$ is selected from a predefined class of parameterized hypotheses established by the evaluator (i.e., the adversary). This class can be thought of as a collection of models $G_{\boldsymbol{\theta}} = \{g_{\boldsymbol{\theta}}(x) \mid g_{\boldsymbol{\theta}} : x \mapsto g_{\boldsymbol{\theta}}(x, \boldsymbol{\theta})\}$ with $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^q$, where q is the dimension of the vector gathering all the parameters cf. [MDP20].

Within the ML community, various evaluation frameworks are utilized to measure model performance and select optimal parameters within a given family of models. According to Bayes error in supervised classification, accuracy maximization is the optimal strategy for

setting model parameters $\boldsymbol{\theta}$. However, finding the global optimum is NP-hard [SPAQ06, BPS+18]; hence, practical training only approximates this objective, and high accuracy does not necessarily guarantee good SCA metrics [BPS+18, MDP20]. In other words, maximizing the accuracy is equivalent to finding only the model that maximizes the probability that the right key is ranked first [BPS+18, MDP20].

3.2 Uncertainty in NNs' Predictions

The prediction process constitutes three principal steps. First, the data acquisition phase is devoted to capturing observations, whereas during the NN building phase, the NN is designed and trained to learn patterns from the measurements. The trained model is then deployed in the applied inference phase, where it is utilized to make predictions. The predictive uncertainty deals with the uncertainty sources within the NN and the data to quantify whether the model's outputs should (not) be trusted [GTA+23]. (see Section 1 for sources of uncertainty in SCA).

Model uncertainty accounts for the uncertainty arising from limitations in the model itself, which may result from errors in the training process, an inadequate model architecture, or insufficient knowledge due to the presence of unknown samples or an incomplete representation of the training data, see Figure 2. In contrast, data uncertainty is linked to uncertainties inherent in the data itself. It arises from the loss of information when capturing real-world phenomena within a dataset. Noteworthy is that, although conceptually related to classical ML decompositions such as Bayes/approximation/estimation/optimization errors or the bias–variance trade-off, our approach adopts a distinct probabilistic interpretation of predictive uncertainty. While model uncertainty can, in principle, be mitigated by refining the model architecture, training process, or dataset quality, data uncertainty is intrinsic to the data and cannot be eliminated [KG17].



Figure 2: Uncertainty decomposition in a binary classification task.

The Bayesian framework provides an elegant approach for modeling uncertainty in deep learning [GG16, TB24]. The underlying principle of this framework is that to estimate y^*, one commonly employs the maximum a posteriori (MAP) approach, which determines the most probable outcome as $y^* = \arg\max_y p(y|x^*, \boldsymbol{\theta})$. Since the underlying model relies on an unknown latent variable θ, an approximation is constructed using a sampled training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, which contains N training samples. In the Bayesian framework, model uncertainty is expressed as a probability distribution over the model parameters $\boldsymbol{\theta}$, whereas data uncertainty is represented as a probability distribution over predictions y^*, given a parameterized model $f_{\boldsymbol{\theta}}$. When predicting y^* for a new sample x^*, the distribution of prediction is given by:

$$p(y^*|x^*, \mathcal{D}) = \int p(y^*|x^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}, \tag{9}$$

$p(\boldsymbol{\theta})$ is the prior distribution over model parameters. $p(\mathcal{D} \mid \boldsymbol{\theta})$ is the likelihood, representing how well the model explains the data. $p(\mathcal{D})$ is the marginal likelihood (evidence). In a classification problem, predictive uncertainty is measured by the entropy of the softmax output distribution as $H(y^*|x^*, \mathcal{D})$ [G+16, GG16, GIG17, HHGL11]. Moreover, in Equation 9, the term $p(\boldsymbol{\theta}|\mathcal{D})$, known as the posterior distribution, corresponds to the uncertainty in the model parameters given the training dataset $\mathcal{D}$. However, this posterior distribution is often intractable. Applying the Bayes theorem to reformulate the posterior distribution yields $p(\boldsymbol{\theta}|\mathcal{D}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})/p(\mathcal{D})$, with $p(\boldsymbol{\theta})$ represents the prior distribution over model parameters, incorporating general knowledge about $\boldsymbol{\theta}$ without

specific observations [BN06]. The likelihood term $p(\mathcal{D}|\boldsymbol{\theta})$ measures the probability that the dataset $\mathcal{D}$ aligns with the distribution defined by a model parameterized by $\boldsymbol{\theta}$. Many standard loss functions, such as cross-entropy and mean squared error, are derived from or directly related to this likelihood function [GTA+23].

Despite this reformulation, the predictive uncertainty remains intractable. To approximate that, various methods have been proposed, offering different trade-offs in accuracy and computational complexity. Next, we discuss how we can use Bayesian methods to estimate uncertainty in the context of SCA. For that, we use the matrix-based estimation of Rényi Entropy introduced next.

3.3 Uncertainty Analysis through Bayesian Modeling

In this work, we develop tools to derive practical uncertainty estimates in NNs by employing Bayesian models, without altering the NN models themselves or the optimization process. Bayesian modeling offers a principled way to estimate uncertainty by treating model parameters as probability distributions rather than fixed values [GG16, G+16, Cha18, JLB+22, SG05, JLB+22][2]. This modeling does not aim to enhance NN's performance, but to explain why a prediction is made by the NN [TB24]. Our approach captures both *epistemic* (model-related) and *aleatoric* (data-dependent) uncertainties more effectively. While the terms epistemic and aleatoric uncertainty are widely used in the machine learning community, for simplicity, we refer to them as model and data uncertainty, respectively. As shown in Equation 9, estimating the posterior distribution of the weights is essential to computing the predictive uncertainty. Therefore, given an already-trained NN, the following steps should be taken cf. [VdSDK+21]: (1) representing existing knowledge about the NN's parameters in a statistical model through a prior distribution $p(\boldsymbol{\theta})$, usually defined before data is collected; (2) defining the likelihood function based on information about the parameters contained in the observed data ($p(\mathcal{D}|\boldsymbol{\theta})$); and (3) combining the prior and likelihood using Bayes' theorem to obtain the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$.

3.3.1 Defining the Prior Distribution

We emphasize that we approximate the posterior over weights, $p(\boldsymbol{\theta} \mid \mathcal{D})$, for a fixed model architecture (i.e., fixed function class). This is standard in parametric Bayesian modeling, where the uncertainty lies over the weights rather than across different architectures (functions). For this, we first need to determine the appropriate distribution for the model's parameters. A reasonable assumption is that the weights follow a normal (Gaussian) distribution. Interestingly, the trained weights often resemble a Gaussian distribution due to the accumulation of stochastic updates. This is supported by the central limit theorem, which states that the sum of many independent variables tends to be normally distributed [Ros14]. In NNs, weight updates during training (e.g., via stochastic gradient descent) can be viewed as the accumulation of many small, approximately independent contributions from mini-batches [LYK24]. As a result, the final weight values, being the sum of these updates, are often approximately Gaussian distributed, especially in large networks or after many updates. More importantly, it is known that NNs with random Gaussian weights can create a stable embedding of the data, serving as a universal classifier [GSB16]. Additionally, common training techniques, such as stochastic gradient descent (SGD), introduce randomness that can contribute to a normal-like distribution of the trained weights [MHB17]. Another reason to assume normality is the weight initialization method used in SCA-related literature, e.g., [WAGP20, AGF23], which follows He initialization [HZRS15]. This initialization method relies on sampling weights from a normal distribution, confirming our assumption.

[2]Note that we do not train Bayesian NNs, but employ principles of Bayesian modeling to compute the uncertainty.

Toy example. To examine whether the trained model's weights follow a normal distribution in real-world scenarios, we use quantile-quantile (Q-Q) plots [WG68]. We stress that the assumption of normally distributed weights in NNs is independent of the input traces. It stems from fundamental properties of how neural networks are initialized and trained. To examine this in the context of SCA, we perform an experiment explained below. The Q-Q plots visually compare the quantiles of our sample distribution (the model's weights) with those of an ideal normal distribution. If the points align along the diagonal, the weight distribution is approximately normal. Deviations from this diagonal indicate skewness, heavy tails, or other irregularities in the distribution. Our use of Q–Q plots is not to define the prior empirically, but to verify whether trained weights exhibit Gaussian-like behavior, often expected due to stochastic optimization dynamics. We used a single model and did not train multiple instances. For each plot, we flattened all weights in a given layer (or the entire model) into a vector and compared them to a normal distribution. This offers a valid empirical check for assuming a Gaussian prior over $\boldsymbol{\theta}$ in Bayesian modeling. Each Q–Q plot analyzes individual weight values rather than summary statistics, thereby avoiding the layer collapsing into a single variable and faithfully capturing parameter and level variation.

The Q-Q plots of the models trained on ASCAD-f, ASCAD-r, and CHES-CTF and used in this paper are presented in Figure 3 (for more information about the models, see Section 4.1). For this experiment, we intentionally select the initialization methods so that one can observe how the Gaussianity of the weights are affected by that. For ASCAD-f, the model uses He initialization [Kum17], which is specifically tailored for ReLU activations and encourages broader weight exploration by drawing from a higher-variance distribution. This enables smoother and more symmetric gradient updates during training, leading to weight distributions that naturally evolve toward Gaussianity. In contrast, ASCAD-r and CHES-CTF adopt Glorot (also known as Xavier) initialization [GB10], the default in many SCA frameworks, which assumes tanh-like nonlinearities and employs lower variance, resulting in more compressed or anisotropic updates. The Q-Q plots show that weight distributions across layers closely follow a Gaussian trend. Nevertheless, the difference in initialization is particularly pronounced in deeper or final layers, where task-specific constraints can amplify deviations. Thus, the more visible Gaussianity seen in ASCAD-f is not merely incidental, but a consequence of the initialization methods.

In our scenario, once we confirm the normality assumption for our model's weights, the next step is to determine the mean and variance of the distribution. By fitting a Gaussian distribution, we experiment with different values of mean and variance to find a distribution that closely matches the actual weight distribution. This means that Gaussian fitting algorithm determines the mean and variance automatically. The blue curve in Figure 4 illustrates the weight distribution for the second layer of the trained model. The orange curves represent distributions close to the original distribution, and we observe that the one on the right provides a closer match to the actual distribution, yielding a higher similarity score (see Section 3.3.3).

3.3.2 Defining the Likelihood Function

In order to determine the likelihood function, one specifies a data-generating model and includes observed data [VdSDK+21]. The likelihood $p(\mathcal{D}|\boldsymbol{\theta})$ is a function of $\boldsymbol{\theta}$ for a given dataset $\mathcal{D}$; hence, the likelihood function summarizes a model that stochastically generates all the traces and a range of possible values for $\boldsymbol{\theta}$ explaining the observations. This specified model, represented by the likelihood function, is the foundation for the uncertainty analysis, as the posterior distribution is the result of the prior distribution determined in this step.
Background knowledge offered by SCA. In various fields, specifying a likelihood function can be nontrivial; however, SCA provides insight into this problem. In the context of SCA, it is assumed that each trace is measured independently from others. Hence, the

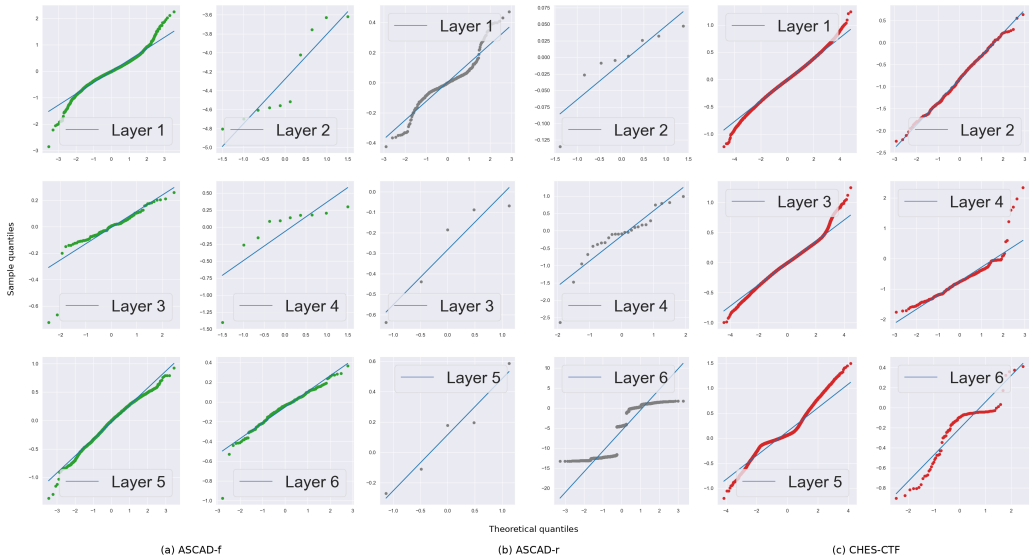


Figure 3: Q-Q plots showing the weight distributions of models trained on different datasets.

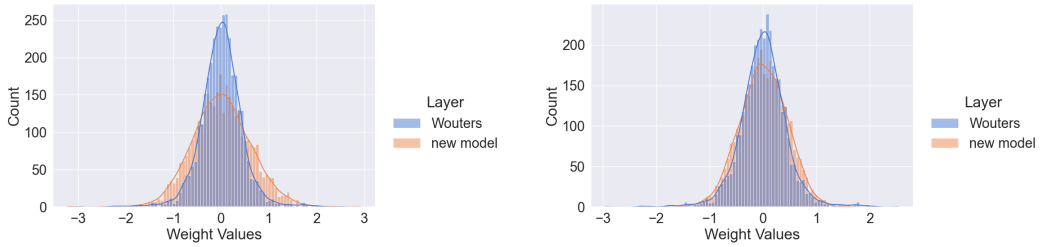


Figure 4: Weight distribution for the second layer of the trained model [WAGP20]. Two different parameter settings are used for sampling (orange curves). The right distribution, which has a closer mean and variance to the actual weights, produces more accurate results.

likelihood function can be written as $p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{y}_i \mid \boldsymbol{\theta}, p_i)$ for a given key. The likelihood $p(\mathbf{y}_i \mid \boldsymbol{\theta}, x_i)$ can be thought of as the output of the output layer in the NN, where logits are translated to probabilities. This is thanks to the connection between NNs in classification and generative settings (see Section 3.1).

The choice of the output layer is closely tied to the selection of the loss function. In line with deep learning principles, classification NNs that employ a softmax output layer tend to learn more efficiently and robustly when trained with the negative log-likelihood (NLL) loss function [GBC16]. It has been shown that minimizing the NLL function (similarly, the cross-entropy loss) during training is asymptotically equivalent to maximizing the estimated mutual information between the side-channel traces and the leakage profiling model (i.e., the machine learning model trained on the traces) [MDP20]. Hence, for networks employing the softmax function and trained on side-channel traces, predictions of the NN can be directly used as the likelihood $p(\mathbf{y}_i \mid \boldsymbol{\theta}, p_i)$.

3.3.3 Bayesian Inference

Defining the likelihood function is followed by *Bayesian inference*[3]. Bayesian inference requires evaluating the true posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$, which cannot be evaluated analytically, as explained before. To address this, we use variational inference [Cha18], a powerful technique in Bayesian modeling that allows us to approximate the complex, intractable posterior distributions of the network's parameters. During the profiling phase, this distribution is learned by the model $\hat{g}$ given the profiling traces. Intuitively, variational inference approximates the intractable posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ with a tractable distribution over the model weights, $q_{\boldsymbol{\nu}}(\boldsymbol{\theta})$, with variational parameters $\boldsymbol{\nu}$. In other words, variational inference makes it feasible to use Bayesian methods to transform the problem into an optimization task. The approximated distribution should be as close as possible to the posterior distribution obtained from the original model. Hence, the approximated distribution is usually minimized by calculating the Kullback-Leibler (KL) divergence with regard to $\boldsymbol{\theta}$:

$$\mathrm{KL}(q_{\boldsymbol{\nu}}(\boldsymbol{\theta}) \parallel p(\boldsymbol{\theta}|X)) := \int q_{\boldsymbol{\nu}}(\boldsymbol{\theta}) \log \frac{q_{\boldsymbol{\nu}}(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|X)} \, d\boldsymbol{\theta}. \tag{10}$$

To solve Equation 10, an approximating variational distribution $q_{\boldsymbol{\nu}}(\boldsymbol{\theta})$ should be defined first, and then the KL divergence can be minimized to determine the variational parameter $\boldsymbol{\nu}$. A common choice is a mean-field approximation, where the variational distribution factorizes into independent Gaussian distributions over the weights [HVC93]:

$$q_{\boldsymbol{\nu}}(\boldsymbol{\theta}) = \prod_{l=1}^{L} \prod_{i=1}^{V_{l-1}} \prod_{j=1}^{V_l} \mathcal{N}\left(\boldsymbol{\theta}_{ijl}; \mu_{ijl}^{\boldsymbol{\theta}}, (\sigma_{ijl}^{\boldsymbol{\theta}})^2\right), \tag{11}$$

where L is the number of layers, and V_ℓ is the number of parameters per layer. The parameters $\mu_{ijl}^{\boldsymbol{\theta}}$ and $\sigma_{ijl}^{\boldsymbol{\theta}}$ define the mean and standard deviation of the weight distributions. This approximation assumes the independence of each weight scalar in each layer from all other weights to simplify the Bayesian inference. It is clear that if this assumption does not hold true in practice, the approximation loses important information about weight correlations. To resolve the issue, Monte Carlo sampling [Gra11] and using a mixture of Gaussians prior to each weight [BCKW15] have been introduced.

Why SCA does not comply well with KL divergence. KL divergence-based variational inference suffers from issues known to the machine learning community, including scalability and computational complexity [G+16]. Besides, when it comes to SCA, leakage can follow a multimodal and/or multivariant distribution [LPMS18, LM18, RBAF24, MMOS16, BGH+17, GLRP06, RBFA21, DCGRP19, Riv08, SMY09]. Assuming Gaussian additive noise, which is a standard and widely accepted assumption in the SCA literature, it has been theoretically verified that the estimated leakage function $\hat{g}_{L,P}$ is always composed of some Gaussian PDFs (i.e., following a Gaussian mixture model–GMM), where the key hypothesis k determines how the Gaussian PDFs are mixed cf. [LRP07, PR09, PR10, UKM+17, BGP+11]. In such scenarios, $q_{\boldsymbol{\nu}}(\boldsymbol{\theta})$ fitted with the KL divergence criterion fails to capture the global behavior of the posterior as it fits a local mode of that. In other words, KL divergence neither exhibits "mass-covering property", nor offer the ability to adjust the impact of likelihood ratios as in GMMs [HLLR+16, LG17]. These lacks can result in less accurate approximations, particularly in the case of complex models.

α-divergence. The KL-divergence is one of the options for measuring the distance between the posterior and the tractable distribution $q_{\boldsymbol{\nu}}$. A more general alternative is the α-divergence, defined as:

[3] It is important to note that the term "inference" means differently in Bayesian modeling and deep learning. In Bayesian modeling, "inference" refers to the process of integrating over model parameters $\boldsymbol{\theta}$, which often involves approximation techniques. In contrast, in deep learning, inference typically refers to evaluating the model on test data after training is complete.

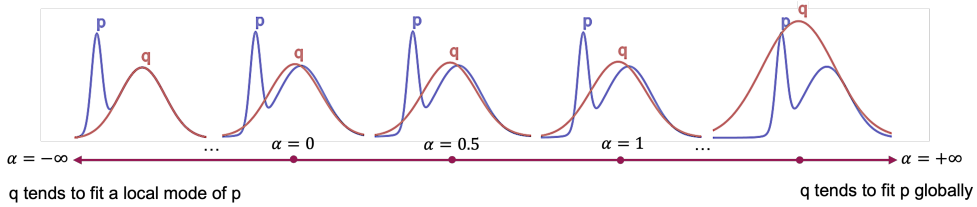


Figure 5: Solutions for minimizing the α-divergence between the complicated posterior distribution p (shown in blue) and the tractable, unnormalized Gaussian approximation q (shown in red) [DHLDVU16, M+05].

$$D_\alpha(p(x) \parallel q(x)) := \frac{1}{\alpha(1-\alpha)} \left(1 - \int p(x)^\alpha q(x)^{1-\alpha} dx\right). \tag{12}$$

α-divergence measures the "similarity" between two distributions. To explore how the choice of α influences the outcome of inference, we consider the task of approximating a complex distribution p with a more manageable Gaussian distribution q by minimizing the divergence between the two; see Figure 5. When α is large and positive, the approximation q tends to cover all the modes of p, whereas for $\alpha \to -\infty$ (assuming the divergence remains finite), q is drawn toward the mode with the highest probability mass cf. [HLLR+16]. If the true posterior is multimodal, a global approximation using $\alpha \geq 1$ will capture multiple modes; however, it also assigns large enough probability to regions where the true posterior has low density. Note that while α-divergence and KL divergence are related, $\alpha = 1$ does not yield KL divergence in the general α-divergence formulation used in our work. Instead, α-divergence with $\alpha = 1$ corresponds to a specific member of the α-divergence family, distinct from KL. As shown in Figure 5, even small variations in α influence how the approximation covers the posterior, with $\alpha \geq 1$ yielding more mass-covering behavior than KL. The choice of $\alpha = 1$ supports a more conservative approach to uncertainty quantification across multiple posterior modes and balances robustness and tractability, which is critical in SCA settings with ambiguity.

3.4 Uncertainty Decomposition in SCA

Understanding what a model does not know is a critical part of NN-enabled SCA. If one is able to assign a high level of uncertainty to the NN's erroneous predictions, then poor attack performance can be explained and, to some extent, improved by taking the right measures into account. The ultimate goal of decomposing the uncertainty in SCA is to link the high rank of the correct key (as one of the possible metrics) to inherent noise in traces and/or uncertainty in the model parameters. While the former cannot be reduced even if more data were to be collected, the latter can be resolved by considering different means, e.g., better hyperparameter tuning, regulation, etc. In this regard, as in the machine learning field, uncertainty can be decomposed into *model* and *data* uncertainties [DKD09, KG17]. As described in Section 3.2, the predictive uncertainty of an attack trace x^* and its output y^*, denoted as $H(y^*|x^*, \mathcal{D})$, is expressed as [DHLDVU18, KG17]:

$$H(y^*|x^*, \mathcal{D}) = I(y^*, \boldsymbol{\theta}|x^*, \mathcal{D}) + \mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|\mathcal{D})}(H(y^*|x^*, \boldsymbol{\theta})). \tag{13}$$

The terms in Equation 13 are explained below.

3.4.1 Predictive Uncertainty Estimation for SCA

The core idea of employing the variational distribution $q_{\boldsymbol{\nu}}(\boldsymbol{\theta})$ is that, after minimizing the divergence to select $\boldsymbol{\nu}$, it closely approximates the posterior weight distribution while remain-

ing simple to sample from. To this end, because the exact posterior $p(\boldsymbol{\theta}|\mathcal{D})$ is intractable, we substitute it with an approximate variational distribution $q(\boldsymbol{\theta})$, once the variational parameters $\boldsymbol{\nu}$ have been optimized. This yields $p(y^*|x^*, \mathcal{D}) \approx \int p(y^*|x^*, \boldsymbol{\theta})q(\boldsymbol{\theta})d\boldsymbol{\theta}$ (see Equation 9). Instead of performing direct integration over the weight space, a common approach is to use Monte Carlo sampling. In this method, m samples are drawn from the weight distribution $\boldsymbol{\theta}_m \sim q(\boldsymbol{\theta})$, and each sample is used to compute the likelihood $p(y^*|x^*, \boldsymbol{\theta}_m)$. With this approximation, the output distribution can be written as: $p(y^* \mid x^*, \mathcal{D}) \approx (\sum_{m=1}^{M} p(y^* \mid x^*, \boldsymbol{\theta}_m))/M$ with each $\boldsymbol{\theta}_k$ being a sample drawn from the approximate weight distribution $q(\boldsymbol{\theta})$.

In the context of SCA, to calculate the predictive uncertainty $H(y^*|x^*, \boldsymbol{\theta}_m)$, we use matrix-based Rényi entropy [YWJP20] (see Figure 6). The reason for our choice is twofold.

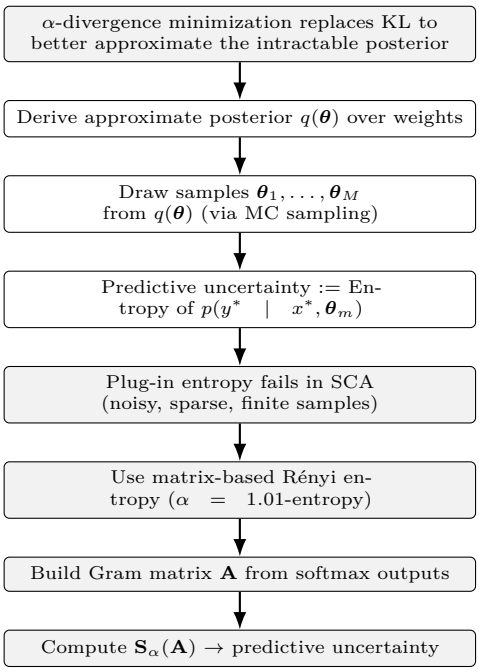


Figure 6: Workflow for predictive uncertainty estimation in NN–based SCA. The blocks in gray are devised especially to deal with SCA settings.

First, large datasets with high dimensions in SCA can be analyzed more straightforwardly when employing matrix computations, which effectively reduce computational complexity. The second reason is more delicate. Information-theoretic quantities, e.g., entropy, heavily depend on the underlying probability distribution of the data. When these probability distributions are unknown, and only a finite set of samples is available, such quantities must be estimated. A common method is the "plug-in" approach, which first fits a model to the available data and then uses this model to compute the desired quantity [GRP14]. While intuitive and (sometimes) easy to compute, this method relies on accurately estimating the data distribution, which can be intractable when, for instance, dealing with noisy data. On the other hand, parametric statistical models require selecting an appropriate model and balancing complexity and tractability, while non-parametric statistical approaches often involve tuning an extra hyperparameter. Moreover, non-parametric statistical models may suffer from overfitting, where smoothing and regularization techniques must be employed to control statistical model capacity. This challenge is especially severe in SCA settings, where leakage traces are often noisy, structured, and misaligned, making density estimation unreliable and entropy estimation unstable. Our matrix-based approach avoids density estimation altogether by using the similarity between NN outputs to compute entropy. This makes the method more stable and well-suited for softmax-based predictions, even in high-dimensional or noisy settings.

Interestingly enough, if a generalized form of Shannon entropy, namely Rényi entropy, is considered, the information-theoretic measures can be quantified using functionals on normalized positive definite matrices. To put it simply, the matrices are obtained by evaluating a positive definite kernel on all pairs of data points [GRP14, YWJP20]. This implicitly maps the data to a reproducing kernel Hilbert space (RKHS) of functions.

To calculate $H(y^* \mid x^*, \boldsymbol{\theta}_m)$ for a given model with parameters $\boldsymbol{\theta}_m$ using matrix-based

Rényi entropy as in Definition 1, we have

$$H(y^* \mid x^*, \boldsymbol{\theta}_m) = \mathbf{S}_\alpha(\mathbf{A}) = \mathbf{S}_\alpha\left(\frac{\mathbf{A}_1 \odot \cdots \odot \mathbf{A}_k}{\mathrm{tr}(\mathbf{A}_1 \odot \cdots \odot \mathbf{A}_k)}\right). \tag{14}$$

In Equation 14, $\mathbf{A}_1, \cdots, \mathbf{A}_{k_t}$ denote the Gram matrices evaluated over the output of the softmax function in the NN.

3.4.2 Estimating Data Uncertainty

Data uncertainty is defined as the expected entropy of the model's predictions over the sets of weights. To estimate that, we perform [Cha18]

$$\mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|\mathcal{D})}\left[H(y^*|x^*, \boldsymbol{\theta})\right] = -\int p(\boldsymbol{\theta} \mid \mathcal{D}) H(y^*|x^*, \boldsymbol{\theta}) d\boldsymbol{\theta} \tag{15}$$

Since the true posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$ is intractable, we approximate it with the variational distribution $q(\boldsymbol{\theta})$, and use Monte Carlo sampling with M samples: $\mathbb{E}_{\boldsymbol{\theta}\sim p(\boldsymbol{\theta}|\mathcal{D})}\left[H(y^*|x^*, \boldsymbol{\theta})\right] \approx -(\sum_{m=1}^{M}\sum_c H(y^* = c|x\cdot\boldsymbol{\theta}_m))/M$, where $\boldsymbol{\theta}_m \sim q(\boldsymbol{\theta})$ are samples from the approximate posterior. This estimates the data uncertainty by averaging over the model's predictive distributions conditioned on sampled weights.

3.4.3 Estimating Model Uncertainty

Model uncertainty $I(y^*, \boldsymbol{\theta}|x^*, \mathcal{D})$ is computed as the difference between predictive entropy and data entropy [Cha18]. This captures the uncertainty due to model limitations, highlighting areas where the model requires more information for improved predictions.

3.5 Sensitivity Analysis in SCA

After understanding how much the model contributes to the predictive uncertainty, one can be interested in explaining which measurement in time (i.e., a feature in the side-channel trace) can account for the uncertainty. Determining the contribution of the time features x_j $(1 \leq j \leq k)$ to predictive uncertainty allows us to pinpoint areas where profiling data is sparse [ABA+20]. A feature exhibiting high data sensitivity suggests a strong relationship with unobserved or latent variables, e.g., noise in side-channel traces. In such cases, collecting more traces could be helpful. On the other hand, a feature with high model sensitivity indicates that the model's confidence heavily depends on it, and thus, careful feature selection can reduce the attack cost and complexity. Hence, sensitivity analysis helps establish the relationship between leakage points and attack cost. Unlike prior works [MDP19, HGG19, ZBHV20, YPB24] that use sensitivity or attribution maps to visualize leakage or identify point-of-interests (PoIs), our approach aims to analyze how time features contribute to predictive uncertainty components; thus, it links explainability with uncertainty quantification rather than feature relevance alone.

In the machine learning domain, sensitivity analysis is a straightforward technique that offers insights into how variations in the input influence the network's predictions. The importance score S_i for the measurement x_i^* quantifies this influence. This score measures the overall significance of an input dimension with respect to a selected uncertainty metric through summing linear approximations centered around each test data point as formulated below cf. [DHLUR17].

$$S_i = \frac{1}{|\mathcal{D}_{\text{test}}|}\sum_{n=1}^{|\mathcal{D}_{\text{test}}|}\left|\frac{\partial H(y|x)}{\partial x_{n,i}}\right|;\ 1 \leq i \leq k. \tag{16}$$

One can leverage the chain rule to compute these derivatives.

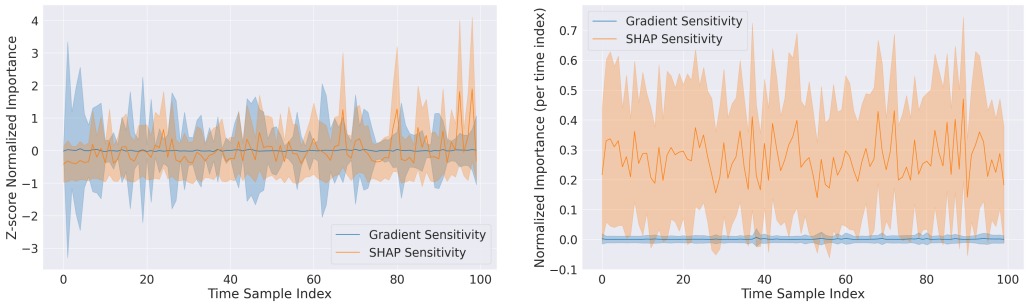
However, the gradient-based approaches, as in Equation 16, assume that other time features remain constant, thereby failing to account for interactions between features. In fact, this equation is unsuitable for SCA due to correlations between successive leakage points [CRR02, MM13, BCM+23]. Given that side-channel traces can be inherently multivariate, the contributions of time samples to uncertainty are interdependent, necessitating a method that captures their interconnections. Yap et al. [YBBP23] have similarly noted that gradient-based methods assume differentiability and feature independence, providing only local explanations and thus failing to capture the interdependence between correlated time samples in side-channel measurements. This has been indeed well-established in the explainable-AI literature since gradient-based attribution methods (such as derivative-based sensitivity) often yield noisy and locally constrained explanations due to the dependency on non-smooth gradients and being sensitive to input scaling; see, e.g., [ACÖG17].

To address this limitation, we use SHAP values, as discussed in Section 2.5. SHAP values assess the contribution of a time feature by considering all possible subsets of other features [LC23]. Unlike gradient-based approaches that provide only local explanations, SHAP captures global interactions among correlated samples by integrating out the influence of remaining features through conditional expectations. More precisely, SHAP aims to allocate credit for a model's output among its input features based on principles from cooperative game theory. The algorithm associates input features with players in a game and aligns the model function with the game's rules. In game theory, players can choose to participate or not, which translates in machine learning to whether a feature "influences" the uncertainty.

When applying SHAP [LL17] in the context of side-channel traces, for each feature, we track changes in uncertainty when that feature is included or excluded from all possible subsets of features. SHAP aims to allocate credit for a model's output among its input features, based on principles from cooperative game theory. The algorithm associates input features with players in a game and aligns the model function with the game's rules. In game theory, players can choose to participate or not, which translates in machine learning to whether a feature influences the uncertainty.

Concretely, each input feature corresponds to a time sample x_i in the leakage vector $x \in \mathcal{D}_{\text{test}}$ and $\mathbf{x} = [x_1, \ldots, x_k]$. To compute the SHAP value ϕ_i, we evaluate how including or excluding x_i from subsets of other time samples affects the model's predictive uncertainty. Formally, ϕ_i is obtained as in Definition 2, where $S \subseteq \{1, \cdots, k\} \setminus \{i\}$ denotes a subset of time features, $f_S = \mathbb{E}_{x \in \mathcal{D}_{\text{test}}}[f(x_S, x_{\bar{S}})] = \mathbb{E}_{x \in \mathcal{D}_{\text{test}}}[H(y \mid x_S)]$, where f_S represents the expected predictive uncertainty when only the time samples in S are set. To quantify each feature's overall influence on the uncertainty, the remaining features are integrated out using a conditional expectation $\mathbb{E}_{x \in \mathcal{D}_{\text{test}}}[H(y \mid x_i)]$. Thus, $f_{(S \cup \{i\})} - f_S$ measures the marginal contribution of time sample x_i to the model's uncertainty, weighted by $\frac{|S|!(k-|S|-1)!}{k!}$ and averaged over all possible subsets. This formulation captures the cooperative influence among correlated leakage points and provides an attribution that respects temporal dependencies in traces, yielding a more faithful explanation of uncertainty propagation. Unlike gradient-based approaches that are sensitive to noise and local non-smoothness [ACÖG17], SHAP produces stable, model-agnostic attributions by averaging feature contributions across all possible subsets.

Simulation results for comparing gradient- and SHAP-based sensitivity analysis. The goal is to examine to what extent the gradient-based sensitivity (Equation 16) fails to identify relevant features when temporal correlations exist, while SHAP correctly attributes uncertainty to correlated time samples. Direct experimentation on datasets such as ASCAD is not feasible for this comparison, as the temporal correlations among leakage samples are inherent to the measurement process and cannot be explicitly controlled or varied;



(a) (b)

Figure 7: (a) Z-score and (b) Min-Max normalized sensitivity profiles (mean ± std) computed across 30 runs using SHAP and gradient-based methods. While gradient sensitivity remains flat and uniform, SHAP reveals a richer and more dynamic structure across time samples. This contrast illustrates SHAP's superior ability to capture non-linear and sample-specific feature importance in side-channel traces.

hence, we perform a simulation. For this, a synthetic SCA dataset is generated to enable controlled evaluation. Each trace $x \in \mathbb{R}^k$ with $k = 100$ has been formed according to $x_t = s + \rho x_{t-1} + \epsilon_t$, where s denotes the intermediate value. To better mimic a SCA scenario, we have defined s as the output of an AES S-box. ρ controls inter-sample correlation (e.g., $\rho = 0.8$), and $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ represents additive noise. The traces have been labeled by the sensitive bit ($y = s$) following the ID leakage model, and the leakage has been intentionally made temporally correlated to mimic the overlapping behavior of consecutive operations.

To launch the attack, we trained an MLP model with three dense layers, the first two layers have 128 and 64 neurons with ELU activation, and the last layer has 256 neurons and softmax activation. For this attack, we generated 10,000 traces, of which 9,000 were used for the profiling phase and 1,000 for the attack phase. Through the learning curve diagnosis, we found the correct number of epochs at which the model is well-trained and not overfitted. The model reached rank zero with 6 traces ($T_{GE0} = 6$), after training with 180 epochs.

Results shown in Figure 7 reveal critical differences in attribution behavior under the temporally correlated leakage model. In Figure 7(a), we present the results for normalizing sensitivity vectors (SHAP and gradient-based) across the time sample axis. This figure enables direct pattern comparison and highlights relative importance variation within each method. In this figure, both SHAP and gradient-based sensitivities are centered and scaled, allowing comparison of structural behavior. SHAP's sensitivity profile also reveals localized peaks of influence. Besides this, SHAP exhibits broader, higher-amplitude fluctuations, especially toward later time indices, where leakage accumulates due to recursive correlation. This demonstrates its ability to capture overlapping effects and attribute uncertainty to the most informative regions. In contrast, gradient-based sensitivity remains flatter and more temporally uniform, failing to track this accumulation. This suggests that gradients could not follow temporally extended influence, offering a limited view of sensitivity when leakage is structured and temporally correlated, as in realistic side-channel traces.

Figure 7(b) presents a case where the min–max normalization is applied point-by-point across samples. This view is informative because it preserves the temporal structure of sensitivity across time while standardizing the scale per time index. Here, the gradient-based method shows low variance across traces and trials, consistent with its deterministic nature. SHAP, on the other hand, shows higher variance due to its stochastic sampling of background data. SHAP can highlight subtle temporal interactions, while the gradient-based

one shows similar importance overall. Moreover, it emphasizes where the methods disagree and how much they diverge in attribution per feature. In this regard, SHAP consistently assigns higher relative importance across time, with especially strong attributions toward the end of the trace, precisely where correlated leakage dominates. The wide confidence bands reflect increased discriminative signal, not instability. The gradient-based scores' overall flatness indicates that it fails to highlight which features contribute most to the model's predictions, potentially masking key regions of importance. Moreover, the smooth and low-variance profile suggests an over-smoothing effect, where gradient-based methods lack sufficient expressiveness to capture actual feature relevance. This discrepancy highlights the limitations of local derivative information in capturing global or multi-step dependencies. In the context of SCA, where key-dependent leakage may emerge nonlinearly and propagate over time, SHAP's ability to marginalize over perturbations allows it to surface attribution patterns aligned with ground-truth dynamics. Hence, although visually more variable, SHAP provides a more faithful and temporally aware attribution, making it a superior choice for explaining predictive uncertainty in side-channel profiling.

4 Results and Discussion

This section begins by introducing a toy example to visually illustrate how uncertainty decomposition works and how its components should be interpreted. In doing so, results in Section 4.2 obtained for CIFAR10 dataset [Kri09] serve as introductory examples to build intuition before transitioning to the more domain-specific challenges of side-channel analysis in Sections 4.3-4.6. CIFAR10 is a widely known benchmark in machine learning, with its moderate complexity allowing clean separation of model and data uncertainty. This helps illustrate the core ideas of decomposition without more involved factors like low SNR found in SCA datasets.

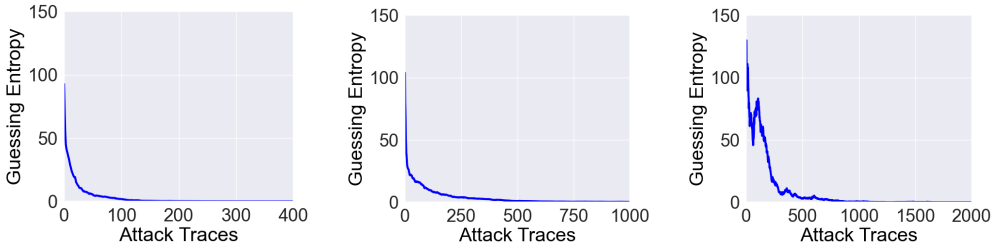
4.1 Experimental Setup

The experiments were conducted on a high-performance computing system running Ubuntu 22.04.3 LTS with a 6.5.0-44-generic Linux kernel. The system was equipped with dual Intel(R) Xeon(R) Silver 4216 CPUs, each with 16 cores and 32 threads, providing a total of 64 logical processors. The system had 256 GB of RAM. For GPU-accelerated computations, an NVIDIA RTX A4000 GPU with 16 GB of dedicated memory was utilized. In all experiments presented in this section, the parameter α associated with α-divergence is set to 1. Moreover, for Rényi entropy, $\alpha = 1.01$ to approximate Shannon's entropy definition.

Dataset split. As standard in SCA, the leakage is modeled as a function of an intermediate variable, specifically, the AES S-box output, not the secret key itself. For instance, in ASCAD-f, which targets byte 3 of the key, the label for each trace is computed as the output of the SBox cf. [BPS+18], yielding 256 classes corresponding to all possible 8-bit outputs. The network is thus trained to classify the S-box output rather than to infer or "memorize" the key. Moreover, with disjoint trace indices for training and testing, no label leakage or key memorization occurs. Therefore, using a fixed key in label generation does not introduce bias and is consistent with the ID leakage model widely adopted in the literature. This setting is identical to ML ones, where the labels in the training and testing datasets come from the same distribution. In this scenario, uncertainty decomposition remains useful because it provides insight into model confidence, model uncertainty, and data limitations even when there is *no explicit* domain shift. In addition, we complement the fixed-key analysis with experiments on ASCAD-r, where keys vary across traces. We emphasize that uncertainty decomposition is not just about distribution shift as seen in datasets like ASCAD-r. It enhances interpretability and performance diagnostics in any

Table 1: The architecture of the models used in our case studies. Here, C(filters, kernel size, strides) and P(size, stride) show the hyperparameters for a convolutional layer and average pooling. FLAT denotes a flatten layer, whereas FC(#neurons) denotes a fully connected layer with the number of neurons given in parentheses. SM(#classes) shows the number of classes at the softmax layer.

Data set	Leakage model	Architecture
ASCAD-f [WAGP20]	ID	P(350,1),FLAT, FC(10), FC(10), SM(256)
ASCAD-r [WPP22]	ID	C(120,3,1), P(32,2), C(8,1,1), P(32,2), FLAT, FC(30), FC(5), FC(5), SM(256)
CHES CTF [WPP22]	HW	P(1100,1), FLAT, FC(512), FC(256), FC(128), FC(64), FC(9), SM(9)



(a) ASCAD-f (b) ASCAD-r (c) CHES-CTF

Figure 8: Average rank (so-called guessing entropy–GE) plots of the datasets used in this paper.

predictive ML setting. Therefore, the absence of a distribution shift (as in ASCAD-f) does not render the analysis uninteresting; on the contrary, it establishes a controlled baseline to understand uncertainty under conditions without a distribution shift.

Architecture of the models. We aim to study the link between attack performance in terms of guessing entropy and uncertainty. For this, we chose typical NN architectures from the SCA literature to ensure reproducibility and isolate the impact of uncertainty modeling without introducing confounding effects from complex or fine-tuned models. Our goal is not to achieve state-of-the-art performance, but to study how uncertainty metrics relate to attack success (e.g., guessing entropy) in representative settings. While our framework is model-agnostic, simpler models offer clearer interpretability, especially for explainability analysis. The architectures of the models that we used are described in Table 1. In the models considered in our study, negative-log-likelihood (NLL) has been used as the loss function. We emphasize that minimizing the NLL loss function (similarly, cross-entropy) during NN training is asymptotically equivalent to maximizing the perceived information as proved in [MDP20]. Furthermore, none of the models used in our study has gone under hyperparameter tuning by us.

GE curves. Figure 8 plots the average rank (so-called guessing entropy–GE) vs. the number of attack traces for the models that we use in our experiments. The model trained on CHES CTF reaches to the rank 0 with 514 attack traces while in the ASCAD-r this number is 475 and for the ASCAD-f is 153.

4.2 Toy Example: Uncertainty Decomposition for CIFAR10

Before we proceed to the results of the SCA datasets, we show the results obtained for the **CIFAR10** dataset [Kri09] to illustrate the uncertainty decomposition in a well-known dataset cf. [Cha18]. Analysis of the CIFAR10 test images was performed by

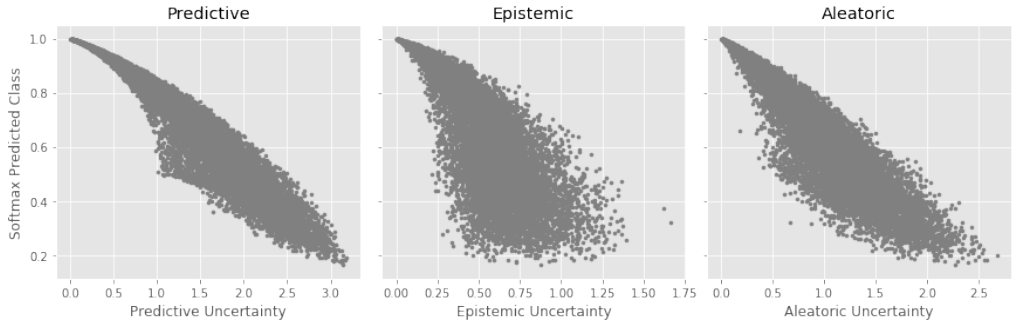


Figure 9: Uncertainty decomposition in **CIFAR** [Kri09, Cha18]. Each point represents a single image in the test dataset. A reduction in the probability delivered by the softmax in the NN results in an increase in model and data uncertainties (downward trend observable in all three sub-figures).

decomposing the predictive uncertainty into its data and model components, as described in Section 3.3. The predictive softmax output $p(y^* = c \mid x^*)$ was computed by averaging weight samples. When the maximum probability delivered by softmax reaches 1, a full probability assignment to a single class is observed. This results in zero uncertainty with regard to both model and data uncertainties. A low softmax output (around 0.2, as shown in Fig. 9) indicates that predictive uncertainty is influenced by model contributions, e.g., differences between the test examples and training examples. The data uncertainty reflects the inherent difficulty in categorizing an image based on the available features.

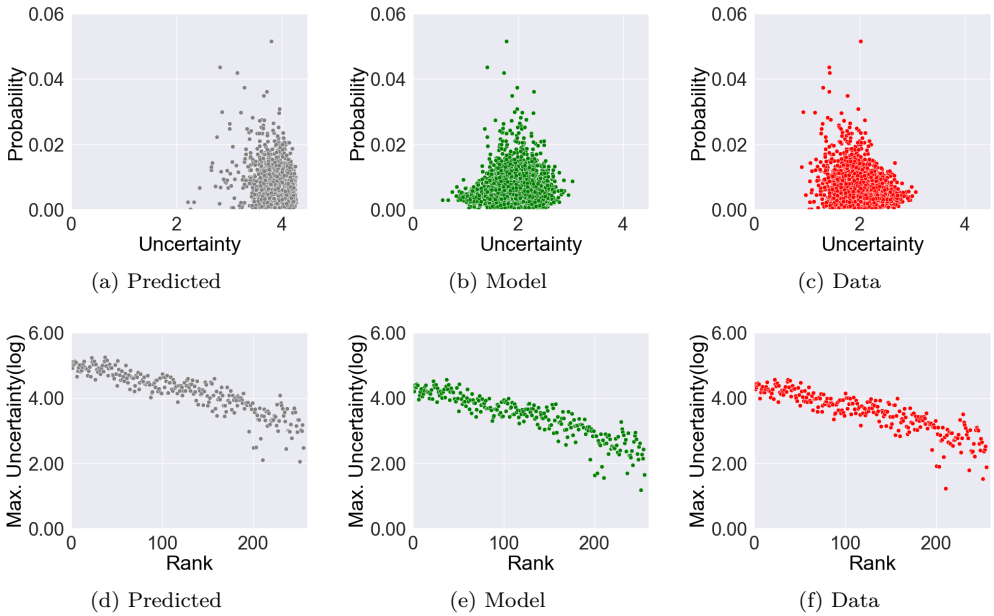
4.3 What Can Be Explained in SCA through Uncertainties?

In this section, we show our results for different case studies. These case studies are designed to explain the behavior of the models trained on SCA datasets. Considering ASCAD-f, as we can see in Figure 8, it takes more than 100 traces for the models to reach rank zero. This can be associated with giving higher probabilities to the wrong keys in the attack phase. Now, we try to find the root cause of this behavior by decomposing the uncertainty based on Equation 13. In our first scenario, we use the model trained on the ASCAD-f dataset with no desynchronization as proposed in [WAGP20][4] We find the variational distribution $q(\boldsymbol{\theta})$ of the model as we discussed in section 3.2 and decompose the uncertainty for this model.

Figure 10 illustrates the results for 5,000 attack traces. We decompose the overall predictive uncertainty into two primary components, data and model uncertainties, which were found to contribute equally in this case. The relatively high value of data uncertainty, i.e., a large fraction of the total predictive uncertainty compared indicates that the available traces do not contain enough information to reliably predict the key. This suggests that intrinsic noise and variability within the traces dominate the model's confidence, revealing that even a well-trained classifier cannot reduce this uncertainty without higher-quality or better-aligned leakage. To reduce this type of uncertainty, we need more informative traces, for example, by employing better probes to minimize noise. On the other hand, the significant amount of model uncertainty suggests that the trained model cannot generalize well to the attack traces. This implies that the model's predictions are less reliable in regions where no data has been seen.

In Figure 10, we also illustrate the link between uncertainty and the attack performance,

[4]We emphasize that none of the models in our study underwent hyperparameter tuning. Our goal is to evaluate how existing models perform and how their attack effectiveness relates to uncertainty.



(a) Predicted (b) Model (c) Data

(d) Predicted (e) Model (f) Data

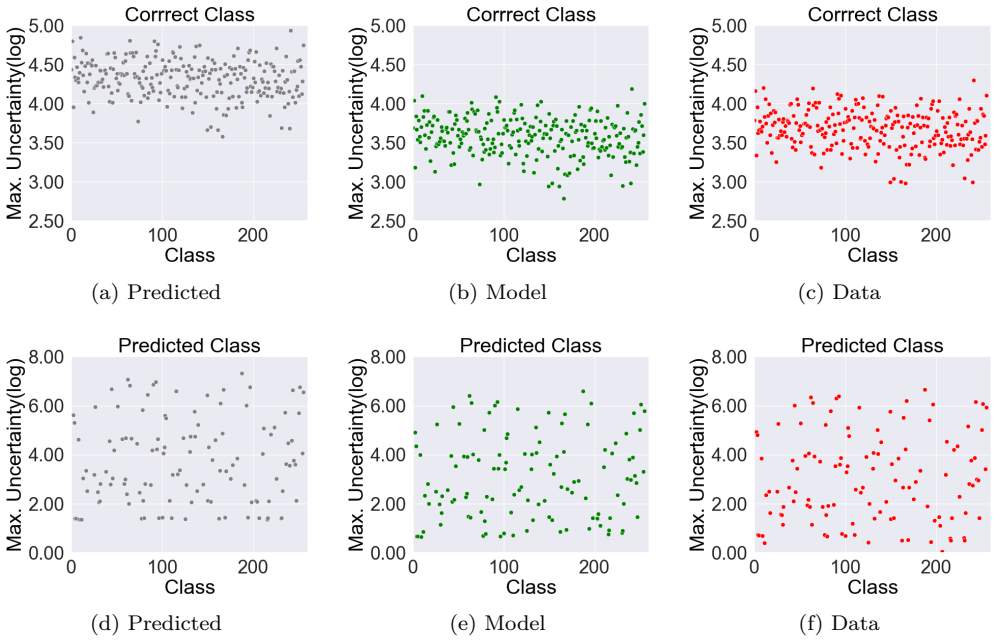
Figure 10: Uncertainty decomposition and rank comparison for the model proposed in [WAGP20] and trained on ASCAD-f dataset with no desynchronization. The first row shows the relationship between the softmax output and uncertainty, where each point corresponds to one trace. The bottom row plots maximum uncertainty against rank, with each point aggregates uncertainty over all traces with that rank.

specifically the rank of the correct key. To achieve this, we calculate the rank of the correct key based on the output probabilities (see Equation 3). Subsequently, we summed up the uncertainty values associated with the traces with the same rank. The results reveal a clear trend: traces with lower ranks exhibit higher uncertainty. This observation shows that the model is unsure about its prediction, although the rank is low.

Is there any link between the labels and uncertainties? Specifically, the objective is to observe whether there is any relationship between the model's uncertainty and the labels of the traces (one-hot encoding of the respective output of the SBOX). The sub-figures in the first row of Figure 11 present the uncertainty of the traces labeled with the correct key. We summed up the uncertainty values of traces sharing the same label, i.e., the correct key. As depicted in Figure 11, the results do not indicate a clear relationship between these two parameters, suggesting that trace labels do not significantly influence the uncertainty associated with model performance.

In Figure 11, we also analyze the relationship between the NN predicted key and the uncertainty for 5000 traces (see the second row). An interesting observation about this is that some of the key candidates never achieve rank 0, resulting in an assigned uncertainty of "1" (or "0" on a log scale) for those classes. Apart from this, the results once again reveal no strong relationship between the predicted key and uncertainty.

What is the impact of randomized keys on the uncertainties? In the ASCAD-f dataset, the same key is used for collecting both profiling and testing traces, while in ASCAD-r, random keys are used during the profiling stage, and a fixed key is used for the attack stage [BPS+17a, BPS+17b]. We employed the model trained on ASCAD-r [WPP22] to evaluate the feasibility of our approach under more challenging conditions.



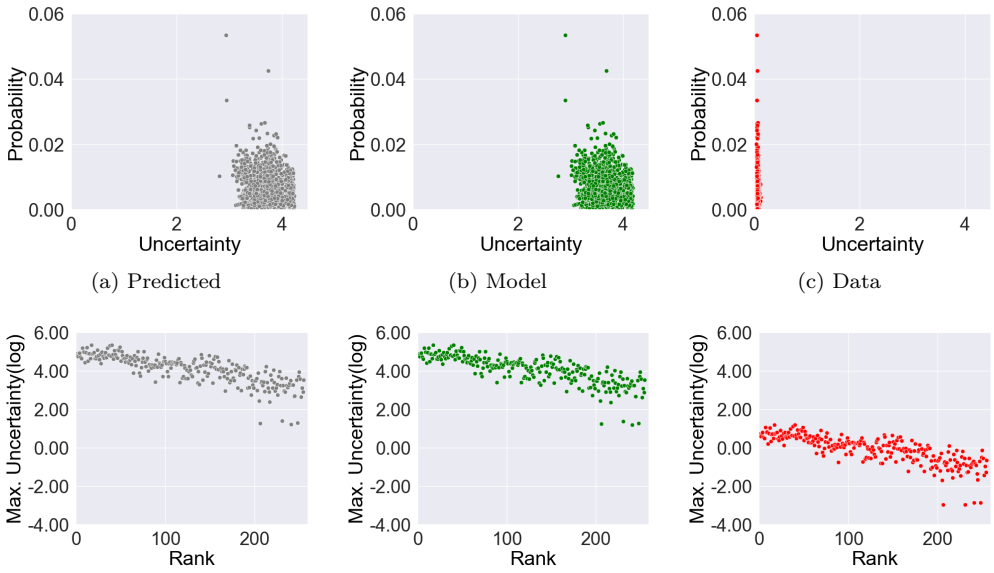
(a) Predicted (b) Model (c) Data

(d) Predicted (e) Model (f) Data

Figure 11: Uncertainty decomposition for the model from [WAGP20], trained on the ASCAD-f dataset without desynchronization. The first row shows the relationship between the softmax output of the correct class, where each point corresponds to the summation of uncertainty values when the given class was the correct class. The sub-figures in the second row depict the summation of uncertainty values when the given class was the predicted class.

Specifically, we use 5000 attack traces from the ASCAD-r dataset.This setup further shifts the distribution of the test set from the profiling set compared to the results presented in this section so far. Consequently, the model faces greater difficulty in generalizing its learned patterns from the profiling phase to the attack phase. Therefore, intuitively, we expected to observe a higher degree of model uncertainty than ASCAD-f. The use of different keys for the profiling and attack phases effectively reduces the model's prior knowledge of the unseen data and, consequently, the model's ability to recover the correct key. Figure 12 gives us a good insight: model uncertainty dominates the overall uncertainty in this scenario. This indicates that the uncertainty stems largely from the model's lack of knowledge about the relationship between the profiling and attack phases. This observation underscores the importance of system-specific knowledge in SCA and suggests that the task becomes increasingly tedious when such knowledge is limited.

Now, if we set Figures 12.c and 10.c side by side, the reduction in the data uncertainty is clearly observable, corresponding to the reduced noise in the dataset. We highlight that the ASCAD-f and ASCAD-r differ in their measurement settings cf. [EST+22]. First, they differ in windowing and filtering conditions. ASCAD-f uses a 700-sample cropped window around the S-box operation, while ASCAD-r uses a 1400-sample window. ASCAD-f traces were captured using an analog low-pass filter during acquisition, while ASCAD-r traces were not filtered. This difference was confirmed by ANSSI in a GitHub discussion, noting that "the difference in the characteristics of the shapes of the two campaigns might come from different oscilloscope probes (e.g., applying some low-pass filtering on the first campaign)," cf. [ASC22]. Filtering changes the waveform smoothness, not the statistical uncertainty of the underlying leakage distribution; therefore, it does not invalidate comparing uncertainty



(a) Predicted (b) Model (c) Data

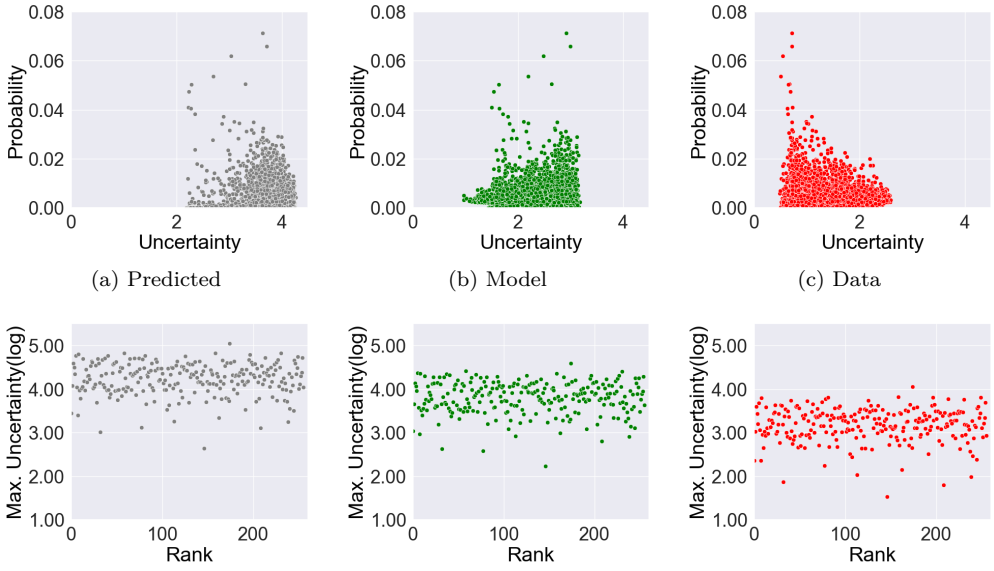
(d) Predicted (e) Model (f) Data

Figure 12: Uncertainty decomposition and rank comparison for the model proposed in [WPP22] and trained on ASCAD-r dataset with no desynchronization. The first row shows the relationship between the softmax output and uncertainty, where each point corresponds to one trace. The bottom row plots maximum uncertainty against rank, with each point aggregates uncertainty over all traces with that rank.

across datasets. On the other hand, the ASCAD-f dataset contains 100,000 time instance recorded at a sampling rate of 200MS/s (not 2GS/s as reported in [BPS+18]), while the ASCAD-r dataset includes 250,000 samples captured at 500MS/s. Both datasets include traces from the first round of an AES software implementation running on an ATmega8515 microcontroller clocked at 4MHz, resulting in 50 and 125 samples per clock cycle, respectively. Increasing the sampling rate can help better capture signal details as one can track rapid transitions in the signal more accurately, potentially separating the signal from some types of noise. Better time resolution can also be achieved, i.e., side-channel leakage can be isolated from other overlapping operations in time. Such differences in the two campaigns can result in reduced data uncertainty for ASCAD-r.

What is the impact of countermeasures like desynchronization on the uncertainty? The objective of this case study is to understand the influence of jitter on the model's uncertainty. To investigate this, we evaluated the model trained on ASCAD-f with no desynchronization as in [WAGP20] when applied against 5000 attack traces from the ASCAD-fixed dataset subjected to 100 desynchronization. This setup introduces temporal shifts, enabling the analysis of their impact on the model's performance and uncertainty levels. Although the model was trained on a dataset without desynchronization, where a small jitter should not significantly affect performance, our results reveal that the model fails to recover the correct key under desynchronized conditions.

Figure 13 illustrates how a substantial shift of the test set distribution introduced by desynchronization causes the increase in model uncertainty. Even minor desynchronization increases the model's uncertainty, disrupting the patterns typically observed in synchronized traces; see Figure 10. Additionally, Figure 13 highlights that although the probability vs. uncertainty plots may seem similar, a small change in their skewness indicates a significant



(a) Predicted (b) Model (c) Data

(d) Predicted (e) Model (f) Data

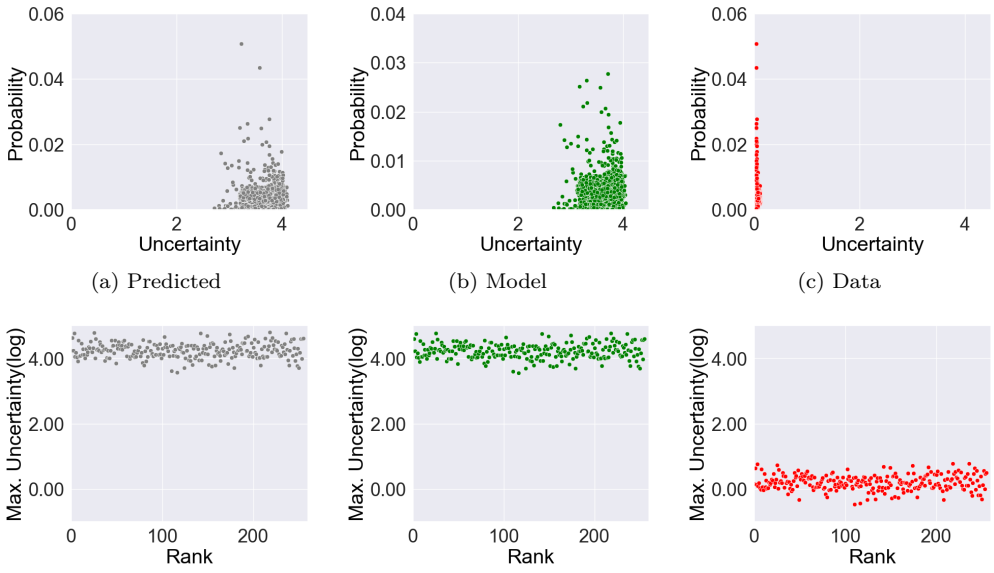
Figure 13: Uncertainty decomposition and rank comparison for the MLP model proposed in [WAGP20] and trained on ASCAD-f dataset with no desynchronization and tested against 5000 attack traces of ASCAD-f dataset with 100 desynchronization. The first row shows the relationship between the softmax output and uncertainty, where each point corresponds to one trace. The bottom row plots maximum uncertainty against rank, with each point aggregates uncertainty over all traces with that rank.

shift in the trend of uncertainty vs. rank plots (see Figure 10). The NN's uncertainty leads to incorrect predictions, and as a result, all guessed keys exhibit high uncertainty regardless of their rank. In a noisy setting like this, such behavior strongly suggests that the model is not functioning effectively.

4.4 What Model Changes Benefit SCA?

This section aims to look at some hyperparameters thought to be useful in the context of SCA through the lens of uncertainty. Given the large space of possible hyperparameters, we focus on some existing methods proposed in SCA-related literature.

Can convolutional filters help the model reduce its uncertainty when dealing with desynchronization? We have already studied how uncertainties can indicate reduced attack effectiveness when an MLP model is trained on a dataset and tested against a dataset containing desynchronized traces. It can be interesting to make a similar observation for convolutional neural networks (CNNs). For this, our model incorporates a convolutional layer and has been proposed in [WPP22]. CNNs are powerful tools for handling sequential or spatial data, as they are designed to identify and capture local patterns through convolutional filters [HTFF09]. Can these filters help the model reduce its uncertainty? The answer is negative. The model uncertainty remains high, although we can observe a clear reduction in the data uncertainty (as explained before, this is due to the ASCAD-r capturing setting). In this case, the model uncertainty values are relatively high, showcasing the inefficacy of the model. As shown in Figure 14 for 5000 attack traces, again, there is no clear relationship between the rank and uncertainty as the model performs poorly, confirmed by the high model uncertainty.



(a) Predicted (b) Model (c) Data

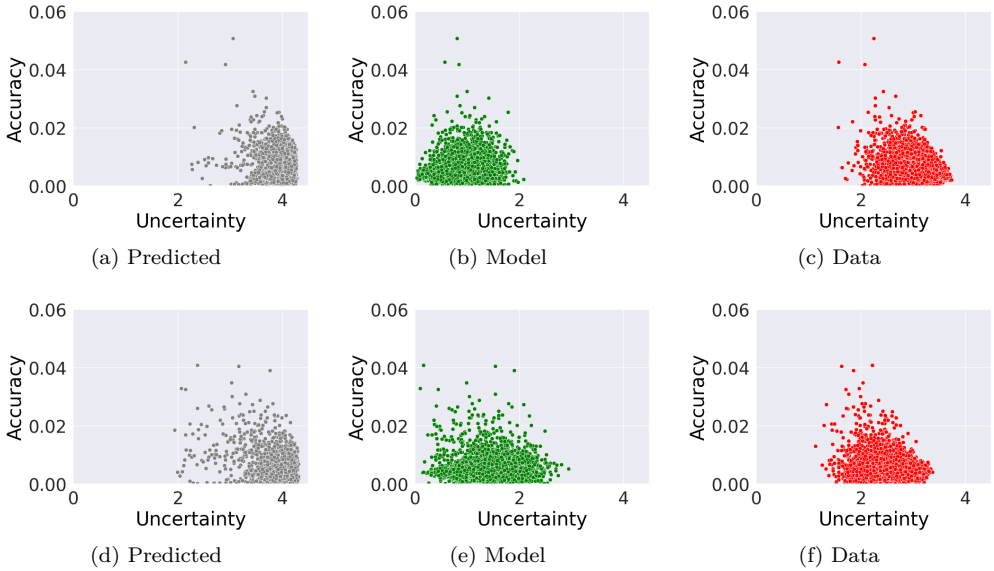
(d) Predicted (e) Model (f) Data

Figure 14: Uncertainty decomposition and rank comparison for the CNN model proposed in [WPP22] and trained on the ASCAD-r dataset with no desynchronization but tested against the ASCAD-r dataset with 100 desynchronization. The first row shows the relationship between the softmax output and uncertainty, where each point corresponds to one trace. The bottom row plots maximum uncertainty against rank, with each point aggregates uncertainty over all traces with that rank.

The impact of the activation function and pooling layers on the uncertainty decomposition. Next, we want to explore how other hyperparameters of the model influence the uncertainty decomposition. We modified the model presented in [WAGP20]. In order to observe the impact of the pooling layer, we first changed the pooling layer of the model from average pooling to max pooling. We can see the results of this experiment in figure 15, first row. Compared to Figure 10, the model uncertainty decreased, and the data uncertainty increased. This aligns with the known selectivity of max pooling, which may overfit to strong features while discarding informative yet uncertain components, thereby concentrating uncertainty into the data uncertainty. This backs why [WAGP20] has employed average pooling for their model. We also conducted the same experiment with a global average pooling (GAP) layer, but since the trained model did not reach $GE = 0$, we did not include the results here.

In the second part of this experiment, we replaced the SeLU activation of the original model with the ReLU activation function; see the second row in Figure 15. Generally, ReLU may lead to overconfident predictions or dead neurons, whereas smoother activations like SeLU better reflect uncertainty under noisy or ambiguous conditions. This can justify why the model in [WAGP20] has been equipped with SeLU. We observe the same trend as for changing the pooling layer. This is in line with the nature of SeLU as a smoother function, which is expected to reduce model uncertainty. However, the change in the activation function had a less pronounced influence on the uncertainty decomposition.

We need to stress that our goal is not to use uncertainty decomposition as a method to tune the hyperparameters, but it could be beneficial for the evaluators to observe the impact of the parameters on the uncertainty decomposition of the model.



(a) Predicted (b) Model (c) Data

(d) Predicted (e) Model (f) Data

Figure 15: Uncertainty decomposition for the model proposed in [WAGP20] with max pooling layer instead of average pooling layer(first row) and ReLU activation function instead of SeLU (second row). Both models are trained on the ASCAD-f dataset with the same setup as in [WAGP20] and Figure 10.

Can an increase in the number of training/validation traces be helpful? Our goal is to showcase the influence of the number of traces in the training process on the uncertainty values. In other words, can uncertainties indicate that more training traces are needed? In doing so, we used different numbers of the training and validation traces to see the impact on the uncertainty decomposition. In this setup, the number of attack traces is 5000 for all models. As shown in Figure 16, changing the number of the training and validation traces does not affect the predictive uncertainty significantly. This does not contradict learning theory, since predictive uncertainty reflects the total expected uncertainty rather than learning efficiency, and its stability simply indicates that reductions in model uncertainty are balanced by data-related uncertainty. This impact is visible for both data and model uncertainties. An increase in the number of validation traces reduces the variance of these uncertainties, suggesting a more stable setting. Comparing the results for Model 1 and Model 3, where the training set is divided into 45K/5K and 49K/1K training/validation traces, the average model uncertainty is much reduced for Model 3 as expected. Nonetheless, the average data uncertainty is increased as the model has not undergone extensive validation in contrast to Model 1. A better balance between the number of traces in the validation and training datasets can be observed in Model 2. However, if there could be an option to reduce the model uncertainty, Model 3 (49K/1K training/validation traces) could be a viable option.

Can we calibrate the softmax outputs to reduce uncertainty? Recently, [NG24] has introduced temperature calibration [GPSW17] in the context of SCA as an effective post-processing method for improving attack in terms of the T_{GE0}. Note that here, the temperature scaling is a representative *example* of regularization techniques in the ML literature. That study has demonstrated that temperature calibration enhances the performance of the attack [NG24]. Noteworthy, temperature scaling [GPSW17] as a practical regularization method differs from how [IUH25] has defined the inverse temperature β to

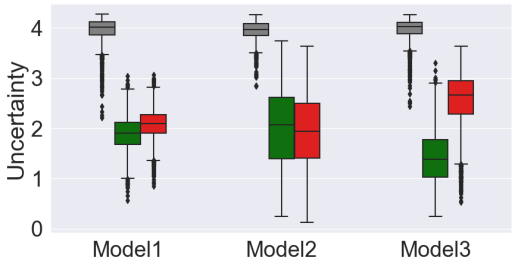


Figure 16: Models trained different validation and training set sizes of ASCAD-f dataset. Model 1 is trained with 45,000 training traces and 5,000 validation traces; for Model 2, we use 47,500 training and 2,500 validation traces, and these numbers were 49,000 and 1,000 for the third model. Here, the gray box is for predicted uncertainty, the green box is for the model uncertainty, and the red box is for data uncertainty.

adjust effective perceived information (EPI). Here, the question that we ask is: how can we see that in terms of uncertainty? In the context of model calibration, let $\hat{q}_i$ denote the calibrated probabilities for class i, and $\mathbf{z}_i$ represent the logits (pre-softmax outputs) of the model. Temperature calibration introduces a scalar parameter $T > 0$ to rescale the logits, modifying the softmax function as follows:

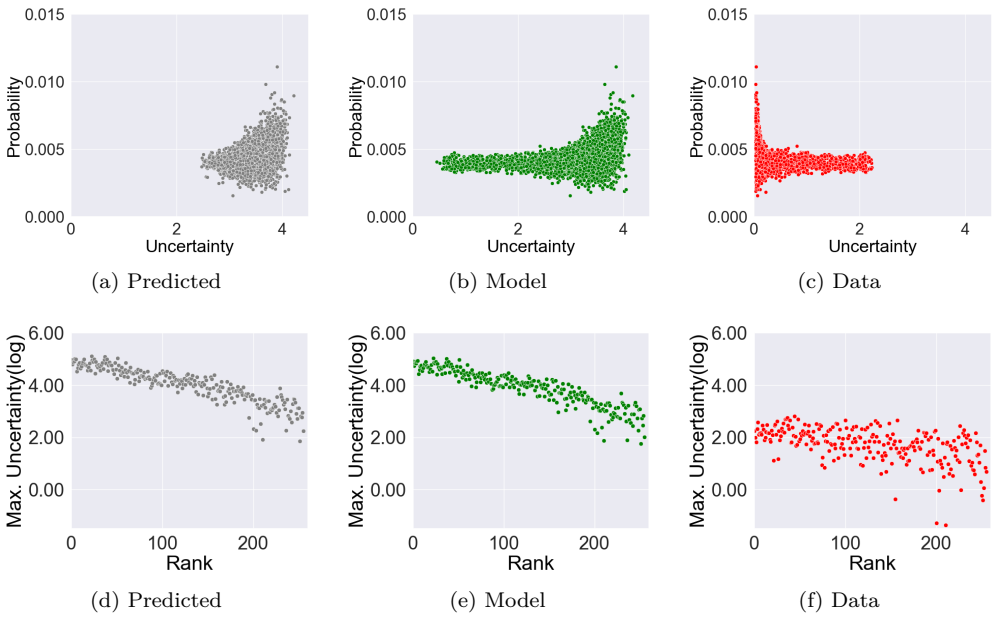
$$\hat{q}_i = \sigma_{SM}(\mathbf{z}_i/T)_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}.$$

When $T > 1$, the output probabilities are distributed more evenly across all classes, effectively increasing the uncertainty of the predictions. This process is commonly referred to as "softening" the softmax [GPSW17]. Conversely, when $T < 1$, the probabilities become skewed, concentrating on the most likely classes. For $T = 1$, no scaling is applied, and the original softmax probabilities are retained. For the MLP model trained on ASCAD-f and used in our paper, namely [WPP22], the temperature has been calculated as 4.11 [NG24].

As depicted in Figure 17, compared to the results in Figure 10, our observation for 5,000 attack traces is the change in the shape of the uncertainty patterns. The softmax probabilities are reduced, and the data uncertainty is much reduced. This shows that the M sampled model could approximate the true posterior $p(\boldsymbol{\theta} \mid \mathcal{D})$ well. Nevertheless, the model uncertainties are increased, more specifically, the shape is now skewed toward right indicating a higher model uncertainty in the prediction. This means that although the output of the model is calibrated, the model lacks the knowledge to be certain about its predictions. The same phenomenon can be observed in Figure 17.f, where regardless of the rank, the data uncertainty remains (more or less) the same. In other words, the main contributor to the predictive uncertainty is the model uncertainty, which is related to the dataset.

Can an increase in the number of epochs be beneficial to reduce the uncertainty? In this case study, we train three different models with varying numbers of epochs, leading to differences in their attack performance. Model 1, trained for 50 epochs, demonstrated the best performance, while Model 2, trained for 35 epochs, ranked second. Model 3, trained for only 20 epochs, performed the worst among the three. As shown in the Figure 19, the guessing entropy of these models varies significantly, reflecting the impact of training duration on the models' ability to recover the correct key.

Increasing the number of training epochs typically enhances model accuracy by enabling it to learn more complex patterns in the data. However, this comes with a trade-off that an excessive number of epochs can lead to overfitting, where the model becomes too tailored



(a) Predicted (b) Model (c) Data

(d) Predicted (e) Model (f) Data

Figure 17: Uncertainty decomposition and rank analysis for the model from [WAGP20] trained on the ASCAD-f dataset without desynchronization, with temperature scaling from [NG24]. The top row shows trace-wise uncertainty versus calibrated softmax outputs, while the bottom row depicts the relationship between rank and maximum uncertainty. Each point represents the total uncertainty across all traces sharing that rank.

to the training data and fails to generalize well to unseen inputs. Before conducting this analysis, we examined whether the model overfit, which was the case for none of the three models. To further explore the relationship between rank and uncertainty, we categorize model predictions into four groups: accurate and certain (AC), accurate and uncertain (AU), inaccurate and certain (IC), and inaccurate and uncertain (IU). Predictions are labeled as "accurate" if the rank is below a predefined threshold, set to 128 in this paper. Being "certain" or "uncertain" depends on whether the uncertainty lies below or above the average uncertainty. This approach allows us to evaluate the trade-offs between rank and uncertainty across different models.

Table 2 presents the results of this analysis for the three models. From the table, we can see that Model 1, the best-performing model, achieved the highest number of accurate and certain predictions (#AC = 59) and accurate and uncertain predictions (#AU = 3566). In comparison, Model 2 had fewer accurate and certain predictions (#AC = 45) and accurate and uncertain predictions (#AU = 2480). Model 3, with the poorest performance, showed only 27 accurate and certain predictions (#AC = 27) and a slightly higher count of accurate and uncertain predictions (#AU = 3003).

Number of the epochs and the problem of overfitting. Finally, we analyze the model's behavior at higher epoch counts to justify our decision to stop training at 50 epochs, as in [WAGP20]. We also need to stress that in the training process, we only used the parameters suggested in [WAGP20].

In the training process, the model's generalization to new, unseen data points is of high importance. This can be diagnosed using learning curves: a model is considered well-trained when the training and validation losses decrease together and the validation loss plateaus or starts increasing, indicating optimal generalization before overfitting. In our experiment, we trained the model for 200 epochs and observed that after 50 epochs,

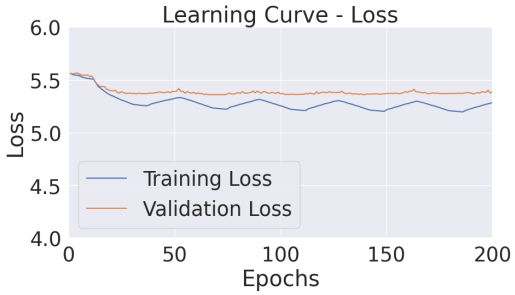


Figure 18: learning curve of the same model proposed in [WAGP20] for ASCAD-f dataset, trained for various numbers of epochs up to 200 epochs.

it showed overfitting, and the validation loss stopped decreasing; see Figure 18. This is the reason why the maximum number of epochs has been set to 50 in our paper and in [WAGP20].

4.5 Interesting Case of CHES-CTF Dataset

Next, we consider the CHES CTF dataset [Ris18], which differs from ASCAD in three important aspects. First, the leakage model is the Hamming weight [GJS19], i.e., the labels correspond to the Hamming weight of intermediate values, resulting in only nine output classes and a reduced maximum uncertainty. Second, the sub-key used during profiling differs from that used in the attack phase, introducing a distribution shift between training and test traces.

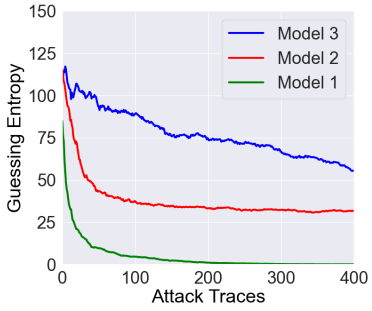


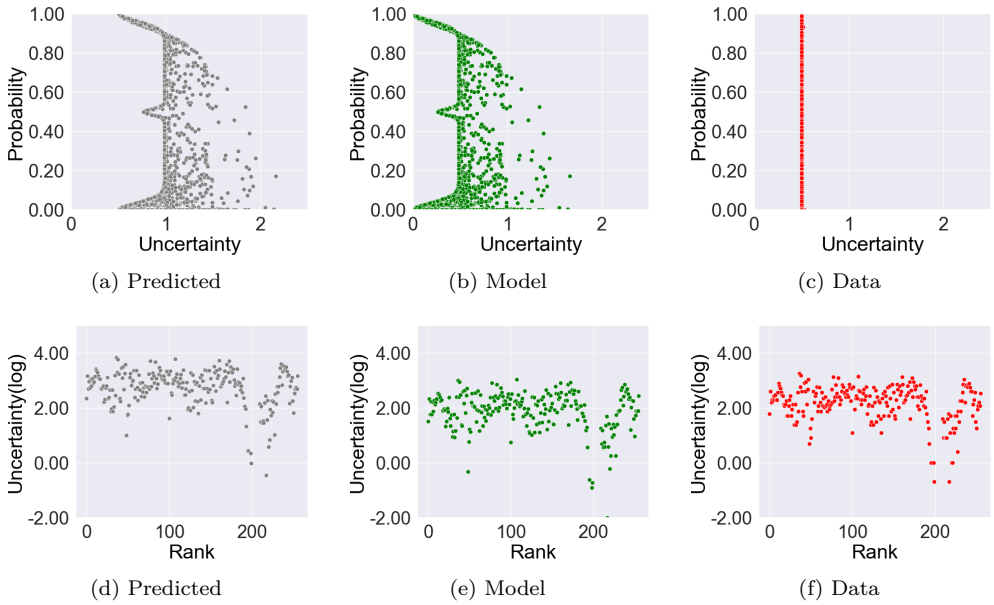
Figure 19: A model trained for different numbers of epochs against ASCAD-f.

Model	#AC	#AU	#IC	#IU
1	59	3566	2	1373
2	45	2480	31	2444
3	27	3003	2	1968

Table 2: Rank vs. uncertainty for the models trained on ASCAD-f dataset [BPS+18]: AC: accurate and certain, AU: accurate and uncertain, IC: inaccurate and certain, IU: inaccurate and uncertain for 5000 attack traces.

Figure 20 illustrates the uncertainty decomposition for 5,000 attack traces of this dataset, where the change in key leads to an increased contribution of model uncertainty. By looking at this figure, we can see a much different pattern in the probabilities delivered by softmax. Compared to other results in this section, the probability varies between 0 and one. More interestingly, model uncertainty is equal to zero for some of the points, indicating that the model successfully approximates the probability density related to $\Pr((P,K)=(p,k)\mid\mathbf{L})$ through $\hat{g}_{\mathbf{L},P,\boldsymbol{\theta}}$.

While $\mathbf{y}=\hat{g}_{\mathbf{L},P,\boldsymbol{\theta}}$ is a discrete probability vector over classes, its empirical distribution across inputs can be well approximated by a mixture of Gaussian-like components. This is motivated by the central limit behavior in leakage aggregation. Regardless of the leakage model (ID and Hamming weight/distance models), the physical leakage is typically modeled as a deterministic function of the sensitive value plus additive Gaussian noise cf. [PR09, PR10, PHJ+17]. When a neural network is trained on such inputs to predict



(a) Predicted (b) Model (c) Data

(d) Predicted (e) Model (f) Data

Figure 20: Uncertainty decomposition and rank comparison for the model proposed in [WPP22] and trained on CHES CTF dataset. The first row shows the relationship between the softmax output and uncertainty, where each point corresponds to one trace. The bottom row plots maximum uncertainty against rank, with each point aggregates uncertainty over all traces with that rank.

class probabilities via a softmax output, its internal representations reflect the underlying structure of the noisy leakage signals [LLLS18]. Due to the aggregation of numerous small, independent noise sources across the trace, especially in long or temporally correlated measurements, the learned feature space exhibits clustering behavior [PN14, LLLS18]. This leads the network's output distributions for a given class to approximate multivariate Gaussian shapes centered on class-specific means. Hence, the predictive uncertainty $H(y^*|x^*, \mathcal{D})$ for a given trace $x^* = \ell^*$ can be approximated by the entropy of t-dimensional Gaussian random variable cf. [PR09, PR10]. This gives $H(y^*|x^*, \mathcal{D}) = 1/2(\log((2\pi e)^t \mid \Sigma \mid))$, with Σ being the covariance matrix of X. To confirm that, we fit an exponential curve to the predictive uncertainty. Curve fitting yields that a single Gaussian distribution can approximate the pdf of $\mathbf{y}$, where the mean and variance are 0.2293 and 0.1218, respectively. This result is in line with what has been observed in [GJS19]. The uncertainty vs. rank curves also show piece-wise upward trends, meaning that the model is more certain about lower-ranked guessed keys compared to previous cases in this section.

4.6 Sensitivity Analysis with SHAP Values

The question that we aim to answer here is that, given the predictive uncertainty $H(y|x,\mathcal{D})$, we can determine which time instances in a trace have an influence on the uncertainty. Figure 22 demonstrates the contribution of time instance to the uncertainty of the model trained on the ASCAD dataset [WAGP20] using SHAP values for 1,000 attack traces. To create SHAP plots, we use a function that maps the time instance to uncertainty, i.e., the trained model that maps the time instance to output distributions. We integrate the Rényi entropy estimation alongside the trained model, using them in series to construct a comprehensive function for uncertainty estimation. This setup enables us to analyze

the contribution of individual time instances to the uncertainty of the model outputs effectively.

Operation-level SHAP perspective. The results reveal that, unlike trace labels, which did not show a significant relationship with uncertainty, operations play a substantial role in determining uncertainty. In Figure 21, each horizontal band in the heatmap (e.g., CM, MO, MOL, OML) shows SHAP values across traces. SHAP values show that high-SNR time instances strongly affect the model's predictive uncertainty, either increasing (red bars) or decreasing (blue bars) it, depending on the underlying operations. This reflects how the model interprets the feature in the context of uncertainty, i.e., class overlap and task difficulty. High-SNR points are typically more informative; therefore, we might expect SHAP values during such periods to be largely negative (i.e., blue), indicating reduced uncertainty. This is apparent for CM and MOL. Compared to CM and MOL with a higher SNR (see Figure 21, MO and OML increase the uncertainty; see the black bar chart on the right side of Figure 21, where CM and MOL are at the negative part of the colorbar, while MO and OML are on the positive side. The reason is that operations MO/OML and CM/MOL exhibit temporal overlap (see Figure 21. This overlap makes class boundaries ambiguous within those time regions. In other words, high-SNR regions not only drive decisions but also introduce greater variance in prediction when the task is intrinsically hard due to class overlap. Consequently, even though these features carry strong signals, they become highly sensitive to small perturbations in input, amplifying predictive uncertainty.

Trace-wise SHAP variability. As illustrated in Figure 22, the high-SNR time samples do not play the same role across all traces (each vertical bar represents a single trace, and the x-axis spans 1,000 such traces). Negative SHAP values indicate that these time instances reduce the uncertainty of the model output (depicted in blue), demonstrating their stabilizing influence. Conversely, the positive SHAP values (depicted in red) indicate that these time instances increase the uncertainty, highlighting their destabilizing impact. For instance, looking at the MO (time samples lying approximately within the 150-200 range, one can observe that its importance varies across the traces; see the horizontal band allocated to MO in Figure 22. Nevertheless, as discussed above, MO's influence does not resolve the uncertainty due to the overlapping operations. Generally, per trace, different groups have varying influences on uncertainty, as indicated by the red- and blue-shaded parts of each bar, respectively. This is in line with the nature of traces collected using different sub-keys. This differentiation underscores the varying contributions of time instances to the model's predictions, emphasizing the importance of feature-specific analysis in understanding model behavior.

Summary. The findings in this section have several practical implications for improving SCA attacks. First, by identifying which time samples contribute most to predictive uncertainty, side-channel evaluators can prioritize denoising, alignment, or enhanced measurement techniques for these critical regions to improve data quality. Second, sensitivity attribution allows for dimensionality reduction by removing time samples that consistently exhibit negligible influence, thereby reducing computational overhead and model complexity. Third, the results offer a way to validate whether the NN's focus aligns with known signal regions (e.g., MO, CM, MOL, OML), supporting explainability and trust in the model. Finally, this analysis provides a granular explanation of poor model performance on certain traces, revealing which time indices contribute to large predictive uncertainty and ultimately degrade key rank—insight that can be used to refine both data and models.

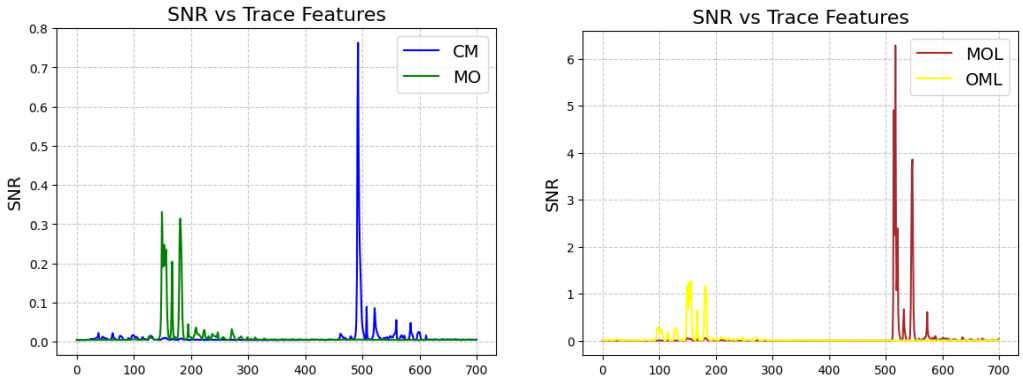


Figure 21: SNR in ASCAD-f dataset: there are 700 time instances in ASCAD-fixed dataset [BPS+20] that based on the SNR we can divide them into four groups: 1. MO (masked SBOX output) 2.CM (common SBOX output mask) 3.MOL (masked SBOX output in linear parts) 4. OML (SBOX output mask in linear parts).

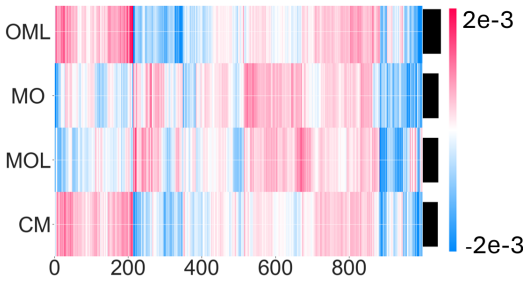


Figure 22: SHAP values of time instances. 1000 attack traces in ASCAD-f are given to the trained model as in [WAGP20]. The x-axis shows the feature indices (time instance), whereas we have 4 groups corresponding to SNR categories in Figure 21. Here, the intensity of the time instances shows more contribution, and the color shows the negative/positive contribution. Positive SHAP values (colored in red) indicate that the time instance increases the uncertainty, while the negative values reduce that.

5 Conclusion and Discussion

In this paper, we provide a framework to estimate and decompose the uncertainties associated with the outputs of the NN models used in SCA. The main goal of this study is to explain why these models tend to assign high probabilities to the wrong key candidates. Our analysis helps find the root cause of this problem. To this end, we use matrix-based Rényi entropy to estimate the entropy as a measure of uncertainty. Through decomposition, it is possible to explain whether uncertainty comes from the lack of information in the traces due to, e.g., the intrinsic noise. This part of the uncertainty can be reduced by employing a better capturing setup, for instance, using better probes while collecting the traces. The other component of the uncertainty is the model uncertainty, which suggests that the model has limits when classifying unseen traces. Multiple case studies have been carried out in this paper to highlight the power of uncertainty as a metric to explain the outcome of an NN-enable SCA. Specifically, the impact of desynchronization and randomized keys have been considered. Furthermore, changing some hyperparameters in

the model has been studied to investigate the impact of convolutional filters, the number of training/validation traces, and the number of training epochs.

Moreover, we utilize SHAP values as an explainable AI tool to find the time sample that contributes the most to the predictive uncertainty. We have demonstrated that the points with high SNR contribute the most to the uncertainty value, which is reasonable since these points have more information regarding the key.

Interpretation of uncertainty results and their practical implications. In general, there is no universal threshold for epistemic (model) or aleatoric (data) uncertainty that definitively indicate whether an attack will succeed or whether more tuning is worthwhile. This is because (1) model and data uncertainty are scale- and task-dependent, especially in SCA where leakage characteristics and labeling assumptions vary; (2) the output uncertainties are relative to the model architecture, dataset size, and leakage model.

In practice, while fixed thresholds for uncertainty are unavailable, relative comparisons and trend-based diagnostics can provide valuable insight in SCA. For instance, observing how uncertainty evolves across training epochs or as the number of traces increases can reveal whether the model is still benefiting from additional data. A decreasing model uncertainty in such cases suggests that uncertainty is reducible and the model is not yet saturated. In contrast, if data uncertainty remains high despite more training, it points to irreducible noise in the traces, such as that caused by desynchronization or countermeasures. Comparing different setups is also informative. For example, training a model on synchronized traces may lead to both low model and data uncertainty along with faster convergence to rank zero. In contrast, using desynchronized traces often maintains high data uncertainty even with extensive training. Such differences can help assess the impact of pre-processing and alignment on model reliability.

Finally, uncertainty metrics can be empirically calibrated against attack performance. One may observe, for example, that model uncertainty below a certain value (e.g., 0.1) tends to correlate with successful key recovery in fewer traces for a particular model–dataset combination. These empirical thresholds, however, must be tuned for each setup rather than being universally applicable. Overall, while absolute thresholds may not exist, analyzing uncertainty trends and comparing across conditions provides practical guidance for attack evaluation and model improvement.

Computational cost and scalability of uncertainty estimation. From a computational perspective, all steps in our framework except the matrix-based Rényi entropy computation rely on standard neural network training and inference procedures commonly used in NN-assisted SCA. The dominant additional cost arises from constructing the Gram matrix and computing its eigenvalues for Rényi entropy estimation, operating on the softmax outputs. Assuming a fixed and constant number of classes (e.g., 256 for AES-128) per task, Rényi entropy estimation has $O(M^2)$ memory and $O(M^3)$ time complexity in the number of Monte Carlo samples M. As M is typically small (e.g., 20–100) to balance estimation quality with computational efficiency, as empirical studies show that after a certain point, increasing M yields very little improvement in the result [LYL+20]. Hence, this step is efficient and scales well to large SCA datasets. The cost is independent of the number of traces and only depends on the number of MC samples.

6 Acknowledgments

This work has been partially supported by NSF under award number 2138420.

References

[ABA+20] Javier Antorán, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. Getting a clue: A method for explaining uncertainty estimates.

arXiv preprint arXiv:2006.06848, 2020.

[ACÖG17] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

[AGF23] Rabin Y Acharya, Fatemeh Ganji, and Domenic Forte. Information theory-based evolution of neural networks for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 401–437, 2023.

[Ama12] Shun-ichi Amari. *Differential-geometrical methods in statistics*, volume 28. Springer Science & Business Media, 2012.

[ASC22] ASCAD Team. Difference of datasets: Sampling frequency / em & power? https://github.com/ANSSI-FR/ASCAD/issues/13[Accessed Oct.9, 2025], 2022.

[BCKW15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

[BCM+23] Sonia Belaïd, Gaëtan Cassiers, Camille Mutschler, Matthieu Rivain, Thomas Roche, François-Xavier Standaert, and Abdul Rahman Taleb. A methodology to achieve provable side-channel security in real-world implementations. *Cryptology ePrint Archive*, 2023.

[BGH+17] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Optimal side-channel attacks for multivariate leakages and multiple models. *Journal of Cryptographic Engineering*, 7:331–341, 2017.

[BGP+11] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *Journal of Cryptology*, 24(2):269–291, 2011.

[Bha06] Rajendra Bhatia. Infinitely divisible matrices. *The American Mathematical Monthly*, 113(3):221–235, 2006.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[BPF20] Samyadeep Basu, Philip Pope, and Soheil Feizi. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020.

[BPS+17a] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. ASCAD: the ATMega8515 SCA traces databases (fixed key). [Online]https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_fixed_key [Accessed: Jan.8, 2024], 2017.

[BPS+17b] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. ASCAD: the ATMega8515 SCA traces databases (variable key). [Online]https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_variable_key [Accessed: Jan.8, 2024], 2017.

[BPS+18] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ascad database. 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(2):163–188, 2020.

[Cha18] Lucy R Chai. Uncertainty estimation in bayesian neural networks and links to interpretability. *Master's Thesis, Massachusetts Institute of Technology*, 2018.

[CHE18] CHES Capture the Flag Organizers. Ches ctf 2018 side-channel analysis dataset (ches_ctf.h5). https://www.dropbox.com/s/lpw1k3so99krmmq/ches_ctf.h5?dl=0, 2018. Accessed: 2025-10-10.

[CJRR99] Suresh Chari, Charanjit S Jutla, Josyula R Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa*

Barbara, California, USA, August 15–19, 1999 Proceedings 19, pages 398–412. Springer, 1999.

[CRR02] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *Intrl. Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002.

[DCEM18] Thomas De Cnudde, Maik Ender, and Amir Moradi. Hardware masking, revisited. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 123–148, 2018.

[DCGRP19] Eloi De Cherisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. An information-theoretic model for side-channel attacks in embedded hardware. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 310–315. IEEE, 2019.

[DHLDVU16] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *arXiv preprint arXiv:1605.07127*, 2016.

[DHLDVU18] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 1184–1193. PMLR, 2018. URL: https://proceedings.mlr.press/v80/depeweg18a.html.

[DHLUR17] Stefan Depeweg, José Miguel Hernández-Lobato, Steffen Udluft, and Thomas Runkler. Sensitivity analysis for predictive uncertainty in bayesian neural networks. *arXiv preprint arXiv:1712.03605*, 2017.

[DKD09] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

[EST+22] Maximilian Egger, Thomas Schamberger, Lars Tebelmann, Florian Lippert, and Georg Sigl. A second look at the ascad databases. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 75–99. Springer, 2022.

[G+16] Yarin Gal et al. Uncertainty in deep learning. 2016.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[GFA+21] Felipe L Gewers, Gustavo R Ferreira, Henrique F De Arruda, Filipi N Silva, Cesar H Comin, Diego R Amancio, and Luciano da F Costa. Principal component analysis: A natural approach to data exploration. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

[GG16] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[GIG17] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International conference on machine learning*, pages 1183–1192. PMLR, 2017.

[GJS19] Aron Gohr, Sven Jacob, and Werner Schindler. Ches 2018 side channel contest ctf-solution of the aes challenges. *Cryptology ePrint Archive*, 2019.

[GLRP06] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. stochastic methods: A performance analysis for side channel cryptanalysis. In *Cryptographic Hardware and Embedded Systems-CHES 2006: 8th International Workshop, Yokohama, Japan, October 10-13, 2006. Proceedings 8*, pages 15–29. Springer, 2006.

[GOKT16] Dennis RE Gnad, Fabian Oboril, Saman Kiamehr, and Mehdi B Tahoori. Analysis of transient voltage fluctuations in fpgas. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 12–19. IEEE, 2016.

[GP13] Luis G Sanchez Giraldo and Jose C Principe. Rate-distortion auto-encoders. *arXiv preprint arXiv:1312.7381*, 2013.

[GPSW17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[Gra11] Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

[GRP14] Luis Gonzalo Sanchez Giraldo, Murali Rao, and Jose C Principe. Measures of entropy from data using infinitely divisible kernels. *Trans. on Information Theory*, 61(1):535–548, 2014.

[GSB16] Raja Giryes, Guillermo Sapiro, and Alex M Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing*, 64(13):3444–3457, 2016.

[GTA+23] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56(Suppl 1):1513–1589, 2023.

[HB10] Jip Hogenboom and Lejla Batina. Principal component analysis and side-channel attacks-master thesis. *Principal component analysis and side-channel attacks-master thesis*, pages 536–539, 2010.

[HGG19] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. Deep neural network attribution methods for leakage analysis and symmetric key recovery. In *International conference on selected areas in cryptography*, pages 645–666. Springer, 2019.

[HHGL11] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

[HLLR+16] Jose Hernandez-Lobato, Yingzhen Li, Mark Rowland, Thang Bui, Daniel Hernández-Lobato, and Richard Turner. Black-box alpha divergence minimization. In *International conference on machine learning*, pages 1511–1520. PMLR, 2016.

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough: Deriving optimal distinguishers from communication theory. In *Cryptographic Hardware and Embedded Systems–CHES 2014: 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings 16*, pages 55–74. Springer, 2014.

[HTFF09] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[HVC93] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[IUH25] Akira Ito, Rei Ueno, and Naofumi Homma. Perceived information revisited ii: Information-theoretical analysis of deep-learning based side-channel attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(1):450–474, 2025.

[JLB+22] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

[KG17] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*, pages 388–397. Springer, 1999.

[Koc96] Paul C Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

[Koz87] Leonenko Kozachenko. Sample estimate of the entropy of a random vector. *Probl. Pered. Inform.*, 23:9, 1987.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/cifar.html, 2009. Technical report, University of Toronto.

[Kum17] Siddharth Krishna Kumar. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*, 2017.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[LC23] Scott Lundberg and SHAP Contributors. An introduction to explainable ai with shapley values. https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html[Accessed Apr.6, 2025], 2023.

[LG17] Yingzhen Li and Yarin Gal. Dropout inference in bayesian neural networks with alpha-divergences. In *International conference on machine learning*, pages 2052–2061. PMLR, 2017.

[LL17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[LLLS18] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

[LM18] Liran Lerman and Olivier Markowitch. Efficient profiled attacks on masking schemes. *IEEE Transactions on Information Forensics and Security*, 14(6):1445–1454, 2018.

[LPMS18] Liran Lerman, Romain Poussier, Olivier Markowitch, and François-Xavier Standaert. Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: extended version. *Journal of Cryptographic Engineering*, 8(4):301–313, 2018.

[LRP07] Kerstin Lemke-Rust and Christof Paar. Gaussian mixture models for higher-order side channel analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 14–27. Springer, 2007.

[LTR17] Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168:1223–1247, 2017.

[LYK24] Jipeng Li, Xueqiong Yuan, and Ercan Engin Kuruoglu. Exploring weight distributions and dependence in neural networks with α-stable distributions. *IEEE Transactions on Artificial Intelligence*, 5(11):5519–5529, 2024.

[LYL+20] Gaoyang Li, Li Yang, Chi-Guhn Lee, Xiaohua Wang, and Mingzhe Rong. A bayesian deep learning rul framework integrating epistemic and aleatoric uncertainties. *IEEE Transactions on Industrial Electronics*, 68(9):8829–8841, 2020.

[M+05] Tom Minka et al. Divergence measures and message passing. 2005.

[MCHS23] Loïc Masure, Gaëtan Cassiers, Julien Hendrickx, and François-Xavier Standaert. Information bounds and convergence rates for side-channel security evaluators. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):522–569, 2023.

[MDP19] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Gradient visualization for general characterization in profiling attacks. In *International Workshop on constructive side-channel analysis and secure design*, pages 145–167. Springer, 2019.

[MDP20] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Trans. on Cryptographic Hardware and Embedded Systems*, pages 348–375, 2020.

[MHB17] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.

[MHK+24] Dev M Mehta, Mohammad Hashemi, David S Koblah, Domenic Forte, and Fatemeh Ganji. Bake it till you make it: Heat-induced power leakage from masked neural networks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(4):569–609, 2024.

[MM13] Amir Moradi and Oliver Mischke. On the simplicity of converting leakages from multivariate to univariate: (case study of a glitch-resistant masking scheme). In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 1–20. Springer, 2013.

[MMOS16] Daniel P Martin, Luke Mather, Elisabeth Oswald, and Martijn Stam. Characterisation and estimation of the key rank distribution in the context of side channel evaluations. In *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22*, pages 548–572. Springer, 2016.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *Intrl. Conf. on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016.

[NDR+22] Ian E Nielsen, Dimah Dera, Ghulam Rasool, Ravi P Ramachandran, and Nidhal Carla Bouaynaya. Robust explainability: A tutorial on gradient-based attribution methods for deep neural networks. *IEEE Signal Processing Magazine*, 39(4):73–84, 2022.

[NG24] Seyedmohammad Nouraniboosjin and Fatemeh Ganji. Too hot to be true: Temperature calibration for higher confidence in NN-assisted side-channel analysis. Cryptology ePrint Archive, Paper 2024/071, 2024. URL: https://eprint.iacr.org/2024/071.

[Pan03] Liam Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003.

[PHJ+17] Stjepan Picek, Annelie Heuser, Alan Jovic, Simone A Ludwig, Sylvain Guilley, Domagoj Jakobovic, and Nele Mentens. Side-channel analysis and machine learning: A practical perspective. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4095–4102. IEEE, 2017.

[PHJ+19] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Trans. on Cryptographic Hardware and Embedded Systems*, 2019(1):1–29, 2019.

[PN14] Hossein Pishro-Nik. *Introduction to probability, statistics, and random processes*. Kappa Research, LLC Blue Bell, PA, USA, 2014.

[PR09] Emmanuel Prouff and Matthieu Rivain. Theoretical and practical aspects of mutual information based side channel analysis. In *Applied Cryptography and Network Security: 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings 7*, pages 499–518. Springer, 2009.

[PR10] Emmanuel Prouff and Matthieu Rivain. Theoretical and practical aspects of mutual information-based side channel analysis. *International Journal of Applied Cryptography*, 2(2):121–138, 2010.

[PYW+17] Sihang Pu, Yu Yu, Weijia Wang, Zheng Guo, Junrong Liu, Dawu Gu, Lingyun Wang, and Jie Gan. Trace augmentation: What can be done even before preprocessing in a profiled sca? In *International Conference on Smart Card Research and Advanced Applications*, pages 232–247. Springer, 2017.

[RBAF24] Unai Rioja, Lejla Batina, Igor Armendariz, and Jose Luis Flores. Keep it unbiased: a comparison between estimation of distribution algorithms and deep learning for human interaction-free side-channel analysis. *Journal of Cryptographic Engineering*, 14(3):499–511, 2024.

[RBFA21] Unai Rioja, Lejla Batina, Jose Luis Flores, and Igor Armendariz. Auto-tune pois: Estimation of distribution algorithms for efficient side-channel analysis. *Computer Networks*, 198:108405, 2021.

[Ris18] Riscure. CHES_CTF: trace database. [Online]http://aisylabdatasets.ewi.tudelft.nl [Accessed: Jan.8, 2024], 2018.

[Riv08] Matthieu Rivain. On the exact success rate of side channel analysis in the gaussian model. In *International Workshop on Selected Areas in Cryptography*, pages 165–183. Springer, 2008.

[Ros14] Sheldon M Ross. *Introduction to probability models*. Academic press, 2014.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[SF24] Gesina Schwalbe and Bettina Finzel. A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. *Data Mining and Knowledge Discovery*, 38(5):3043–3101, 2024.

[SG05] Edward Snelson and Zoubin Ghahramani. Compact approximations to bayesian predictive distributions. In *Proceedings of the 22nd international conference on Machine learning*, pages 840–847, 2005.

[SGK17] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMlR, 2017.

[SHJ+20] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[Sil18] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[SMY09] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Annual Intrl. Conf. on the Theory and Applications of Cryptographic Techniques*, pages 443–461. Springer, 2009.

[SPAQ06] F-X Standaert, Eric Peeters, Cédric Archambeau, and J-J Quisquater. Towards security limits in side-channel attacks: With an application to block ciphers. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 30–45. Springer, 2006.

[SREG+25] Ahmed M Salih, Zahra Raisi-Estabragh, Ilaria Boscolo Galazzo, Petia Radeva, Steffen E Petersen, Karim Lekadir, and Gloria Menegaz. A perspective on explainable artificial intelligence methods: Shap and lime. *Advanced Intelligent Systems*, 7(1):2400304, 2025.

[TB24] Arthur Thuy and Dries F Benoit. Explainability through uncertainty: Trustworthy decision-making with neural networks. *European Journal of Operational Research*, 317(2):330–340, 2024.

[UKM+17] Thomas Unterluggauer, Thomas Korak, Stefan Mangard, Robert Schilling, Luca Benini, Frank K Gürkaynak, and Michael Muehlberghuber. Leakage bounds for gaussian side channels. In *International Conference on Smart Card Research and Advanced Applications*, pages 88–104. Springer, 2017.

[VdSDK+21] Rens Van de Schoot, Sarah Depaoli, Ruth King, Bianca Kramer, Kaspar Märtens, Mahlet G Tadesse, Marina Vannucci, Andrew Gelman, Duco Veen, Joukje Willemsen, et al. Bayesian statistics and modelling. *Nature Reviews Methods Primers*, 1(1):1, 2021.

[WAGP20] Lennert Wouters, Victor Arribas, Benedikt Gierlichs, and Bart Preneel. Revisiting a methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 147–168, 2020.

[WG68] Martin B Wilk and Ram Gnanadesikan. Probability plotting methods for the analysis for the analysis of data. *Biometrika*, 55(1):1–17, 1968.

[WOS14] Carolyn Whitnall, Elisabeth Oswald, and François-Xavier Standaert. The myth of generic dpa... and the magic of learning. In *Topics in Cryptology–CT-RSA 2014: The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, pages 183–205. Springer, 2014.

[WPP22] Lichao Wu, Guilherme Perin, and Stjepan Picek. I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. *IEEE Transactions on Emerging Topics in Computing*, 2022.

[YBBP23] Trevor Yap, Adrien Benamira, Shivam Bhasin, and Thomas Peyrin. Peek into the black-box: Interpretable neural network using sat equations in side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 24–53, 2023.

[YP19] Shujian Yu and José C Príncipe. Simple stopping criteria for information theoretic feature selection. *Entropy*, 21(1):99, 2019.

[YPB24] Trevor Yap, Stjepan Picek, and Shivam Bhasin. Occpois: Points of interest based on neural network's key recovery in side-channel analysis through occlusion. In *International Conference on Cryptology in India*, pages 3–28. Springer, 2024.

[YWJP20] Shujian Yu, Kristoffer Wickstrøm, Robert Jenssen, and José C Príncipe. Understanding convolutional neural networks with information theory: An initial exploration. *Trans. on Neural Networks and Learning Systems*, 2020.

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

[ZR95] Huaiyu Zhu and Richard Rohwer. Information geometric measurements of generalisation. 1995.