Deep Learning-based Side-channel Analysis: How and Why

Stjepan Picek

PROACT Training School, June 4, 2023

Outline

Code and Datasets

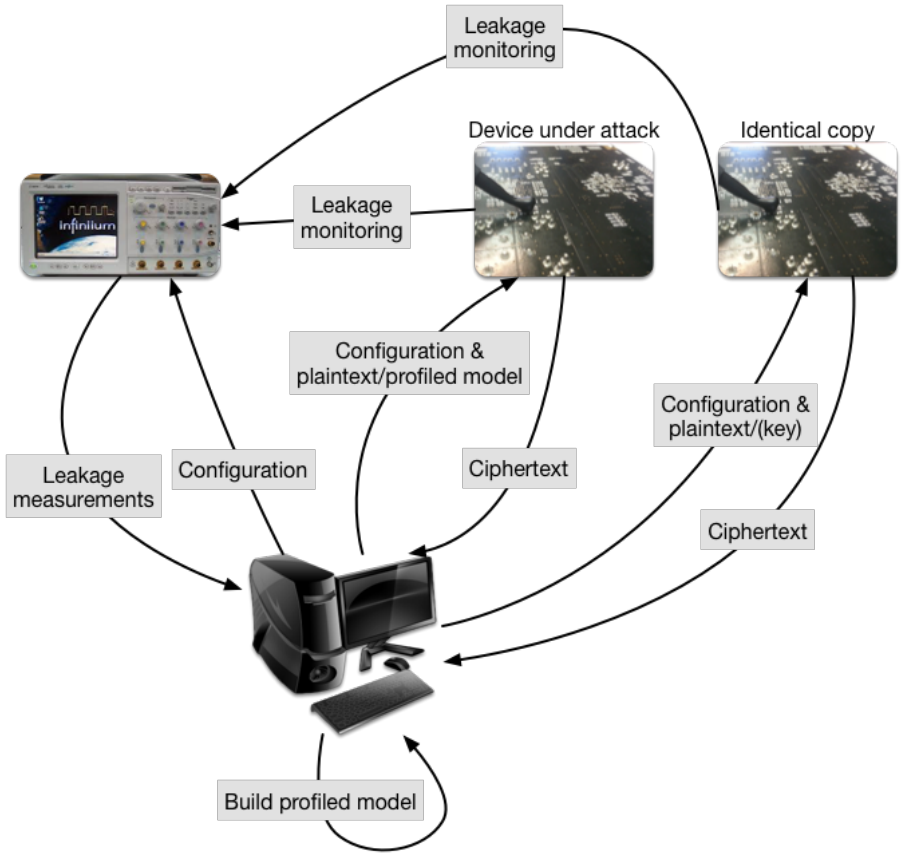
https://tinyurl.com/2s5x9n8x

Outline

Profiling Attacks

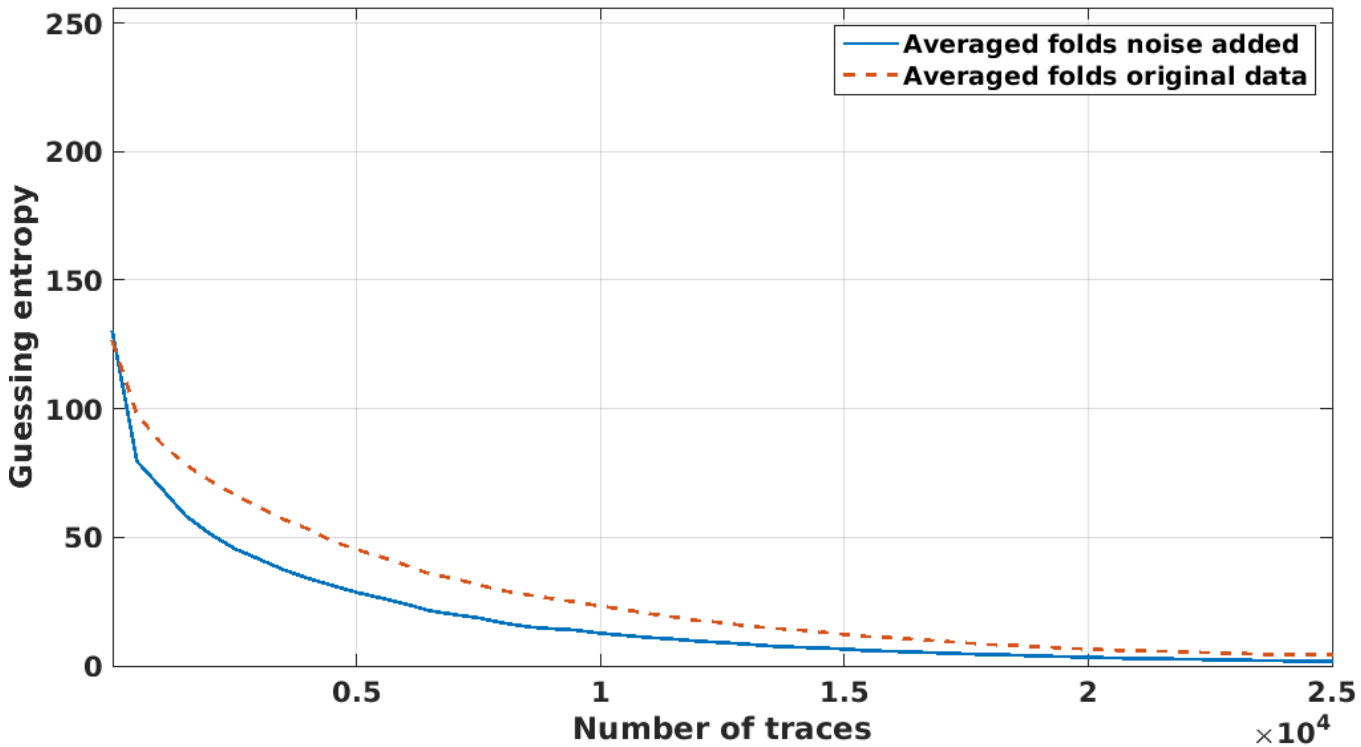
- Profiling attacks have a prominent place as the most powerful among side-channel attacks.
- Within profiling phase the adversary estimates leakage models for targeted intermediate computations, which are then exploited to extract secret information in the actual attack phase.
- Template Attack (TA) is the most powerful attack from the information theoretic point of view.
- Some **machine learning** (ML) techniques also belong to the profiling attacks.

Profiling Attacks



- Profiling attacks are more complicated than the direct attacks.
- The attacker must have a copy of the device to be attacked.

Example of Guessing Entropy



(a) ASCAD network averaged

Outline

Machine Learning

Machine Learning

Machine Learning (ML) is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence.

Machine Learning

Field of study that gives computers the ability to learn without being explicitly programmed

Machine Learning

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured with P, improves with experience E.

Types of Machine Learning

- Supervised learning.
- Unsupervised learning.
- Semi-supervised learning.
- Reinforcement learning.

Supervised Learning

- Supervised learning - available data include information how to correctly classify at least a part of data.
- Common tasks are **classification** and regression.

Notions

- Overfitting and underfitting.
- Regularization.
- No Free Lunch, Curse of Dimensionality, Universal Approximation Theorem, ...
- Parameters vs hyperparameters.

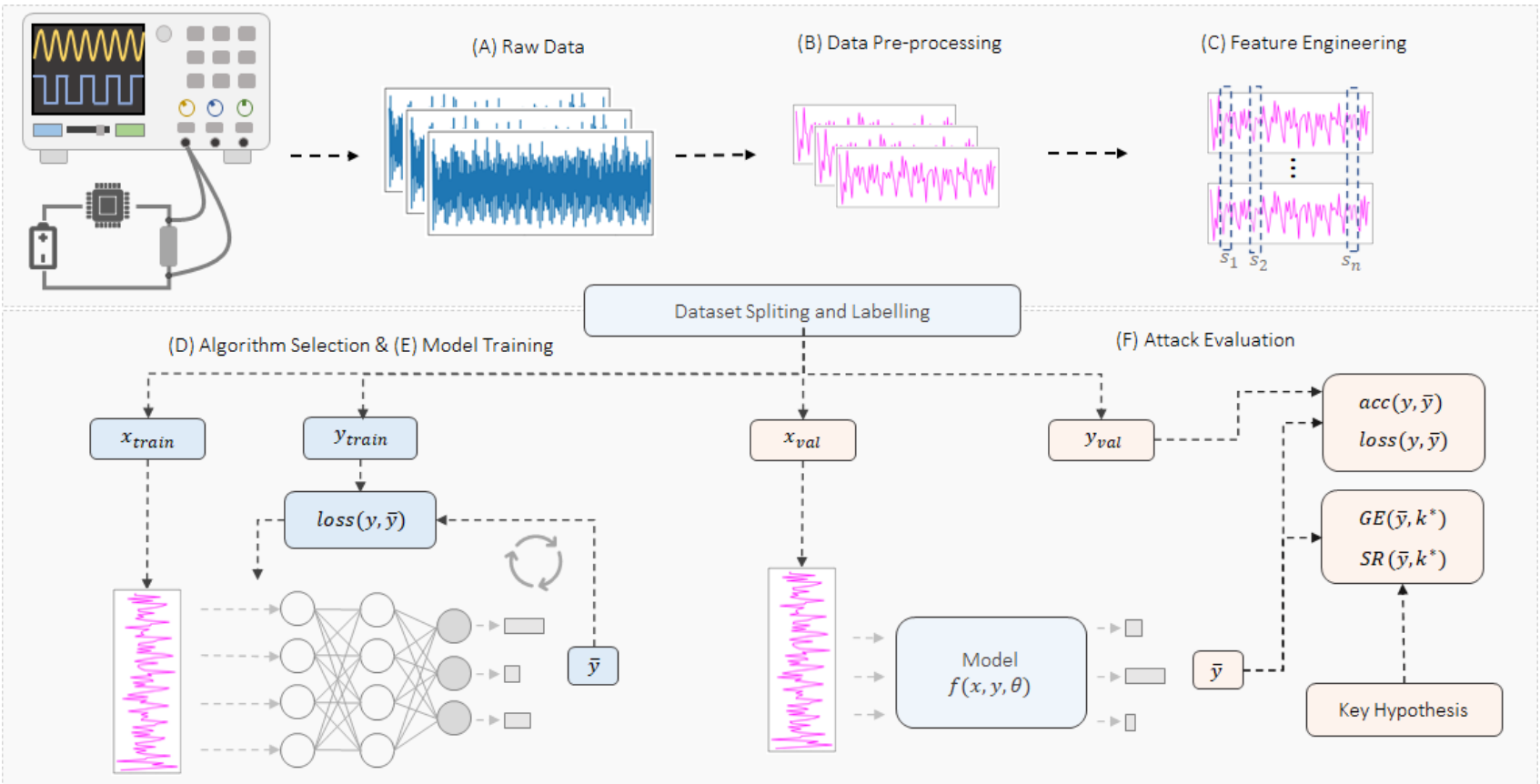
Machine Learning Basic Components

- Model.
- Loss function.
- Optimization procedure to minimize the empirical error.

Common "Traditional" Approaches in Profiling SCA

- Multilayer Perceptron.
- Naive Bayes.
- Support Vector Machines.
- Random Forest.

Machine Learning Process Flow



Deep Learning

- Stacked neural networks, i.e., networks consisting of multiple layers.
- Layers are made of nodes.

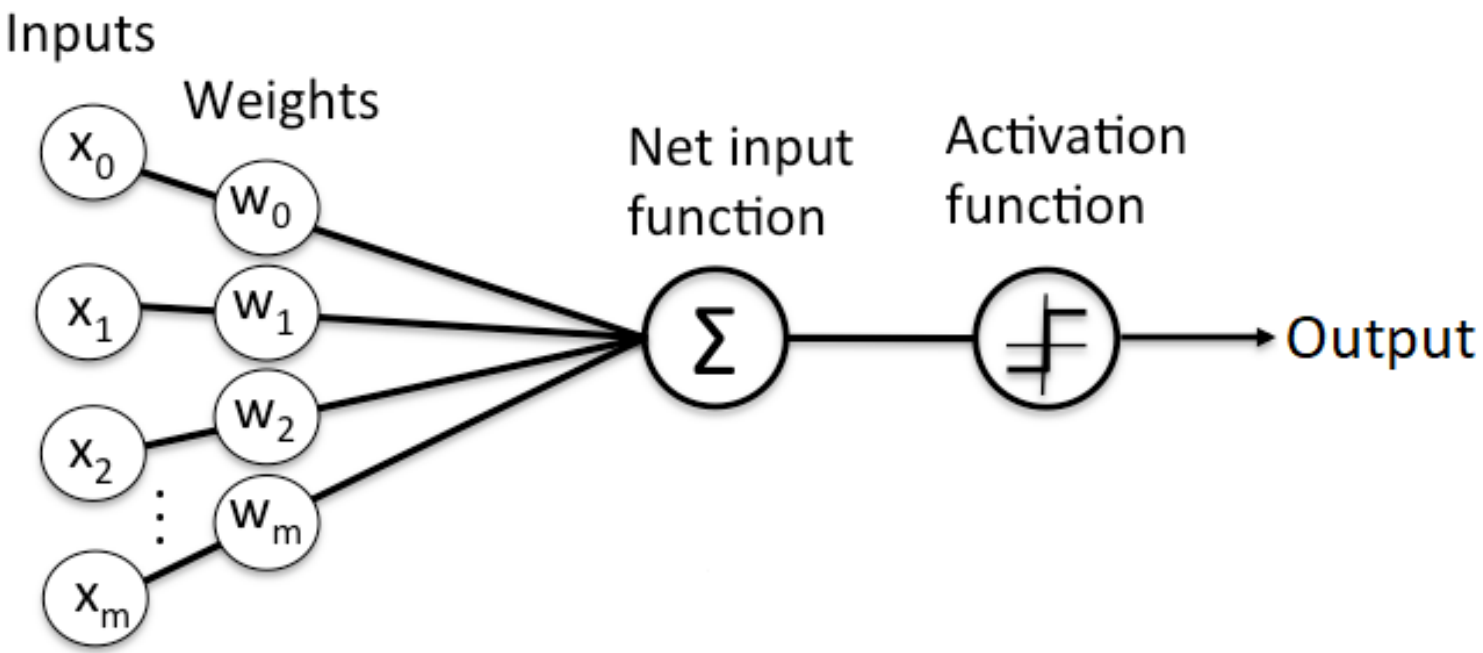


Figure: Perceptron.

Multilayer Perceptron

- One input layer, one output layer, at least one hidden layer.

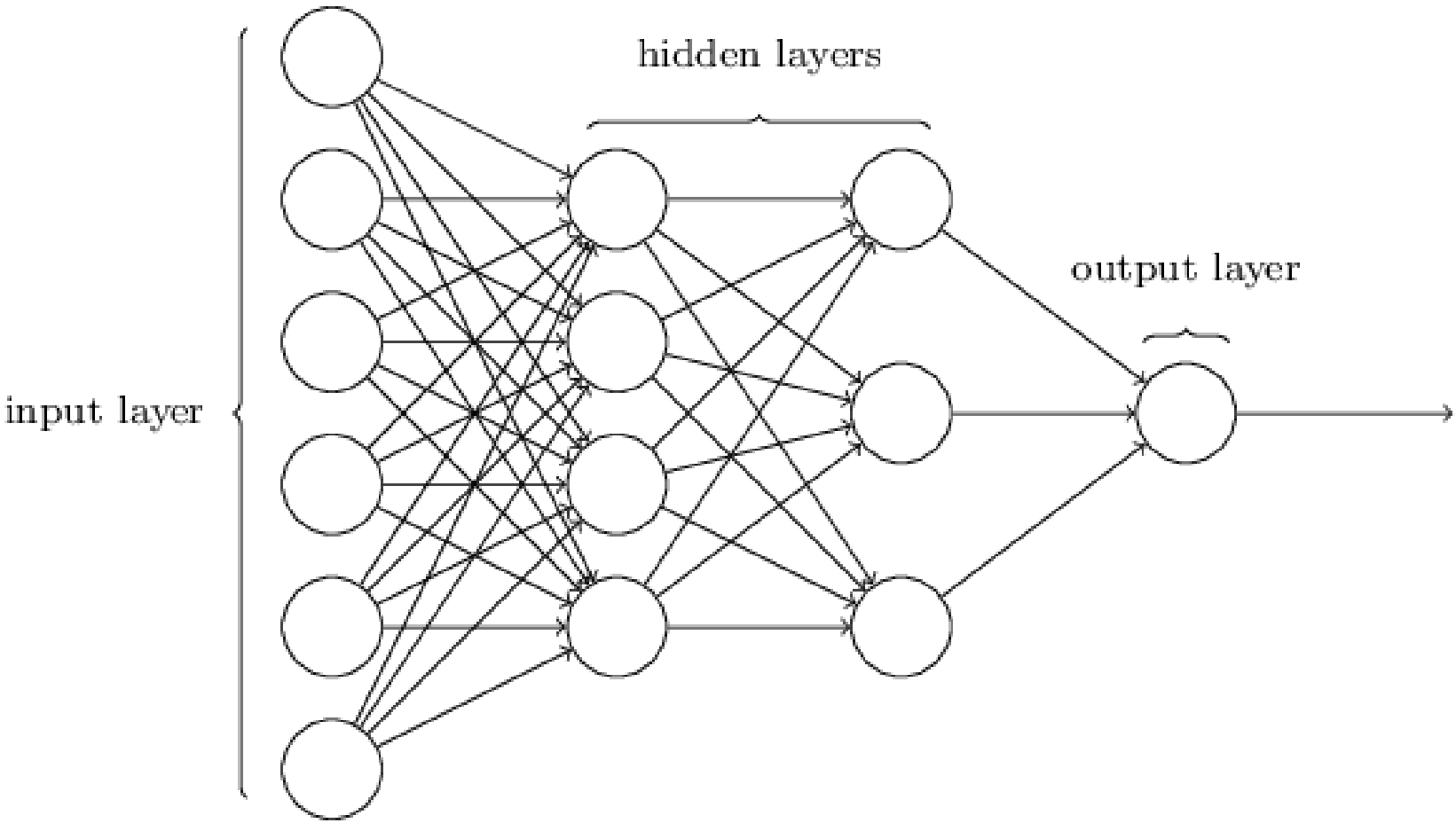


Figure: Multilayer perceptron.

Deep Learning

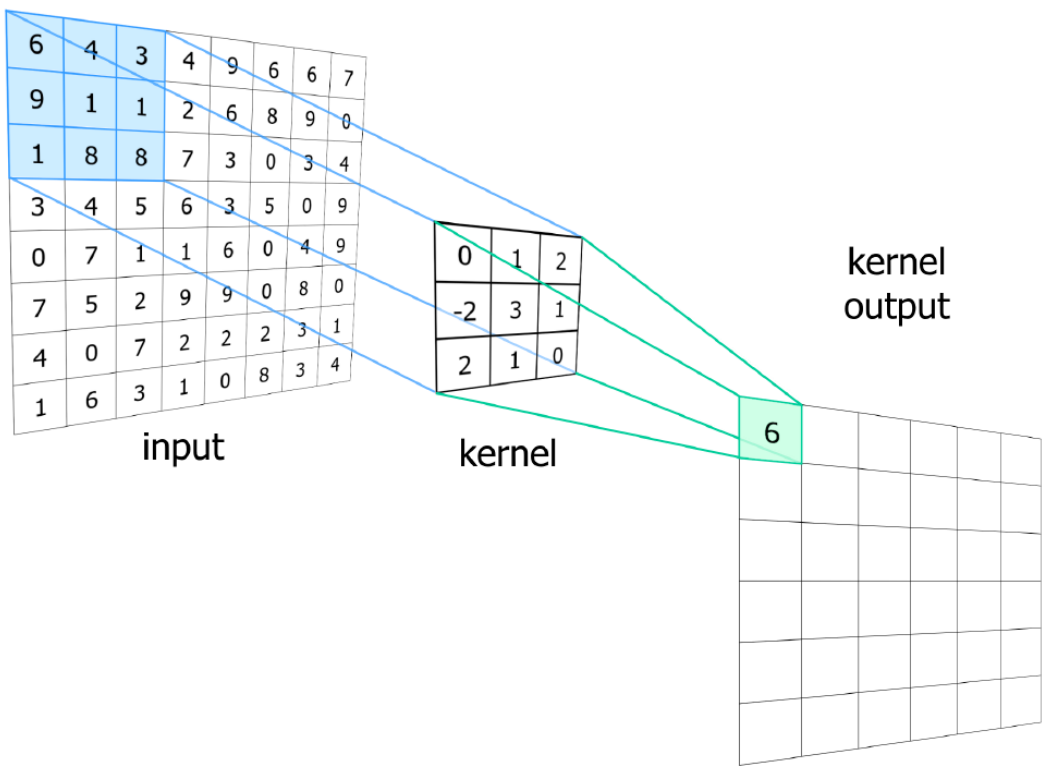
- By adding more hidden layers, we arrive at deep learning.
- Some definitions say everything more than one hidden layer is deep learning.
- A field existing for a number of years but one that gained much attention in the last decade.
- Sets of algorithms that attempt to model high-level abstractions in data by using model architectures with multiple processing layers, composed of a sequence of scalar products and non-linear transformations.
- In many tasks, deep learning is not necessary since machine learning performs well.

Convolutional Neural Networks

- CNNs represent a type of neural network first designed for 2-dimensional convolutions.
- They are primarily used for image classification, but lately, they have proven to be powerful classifiers in other domains.
- From the operational perspective, CNNs are similar to ordinary neural networks: they consist of a number of layers where each layer is made up of neurons.
- CNNs use three main types of layers: convolutional layers, pooling layers, and fully-connected layers.

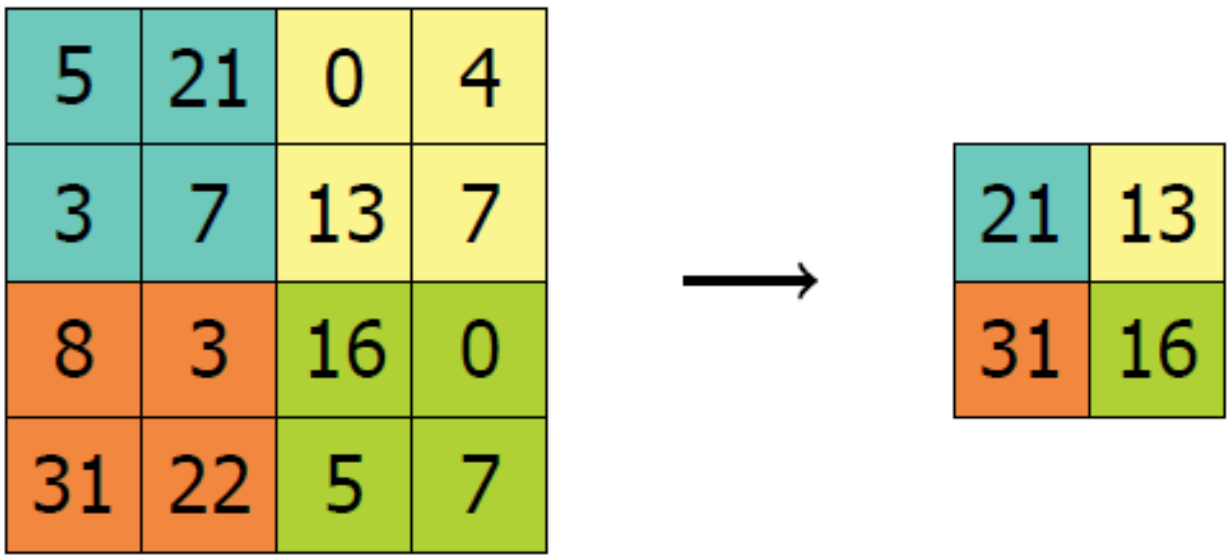
Convolutional Neural Networks - Convolution Layer

Convolutional layer: on this layer, during the forward computation phase, the input data are convoluted with some filters. The output of the convolution is commonly called a feature map. It shows where the features detected by the filter can be found on the input data.



CNN - Pooling

Max (average) pooling layer: sub-sampling layer. The feature map is divided into regions and the output of this layer is the concatenation of the maximum (average) values of all these regions.



Activation Functions

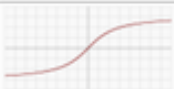
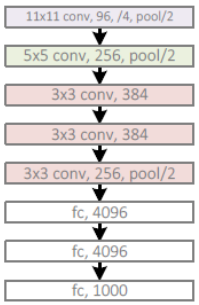
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

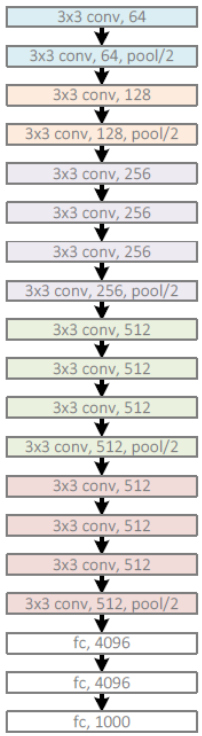
Figure: Activation functions.

More Complex Architectures

AlexNet, 8 layers
(ILSVRC 2012)

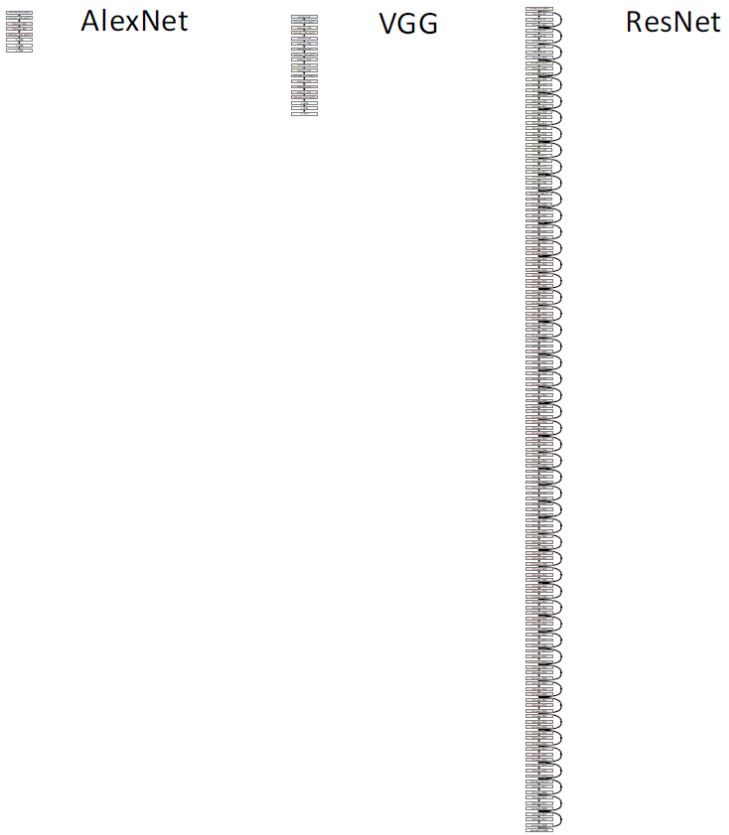


VGG, 19 layers
(ILSVRC 2014)



GoogleNet, 22 layers
(ILSVRC 2014)

More Complex Architectures

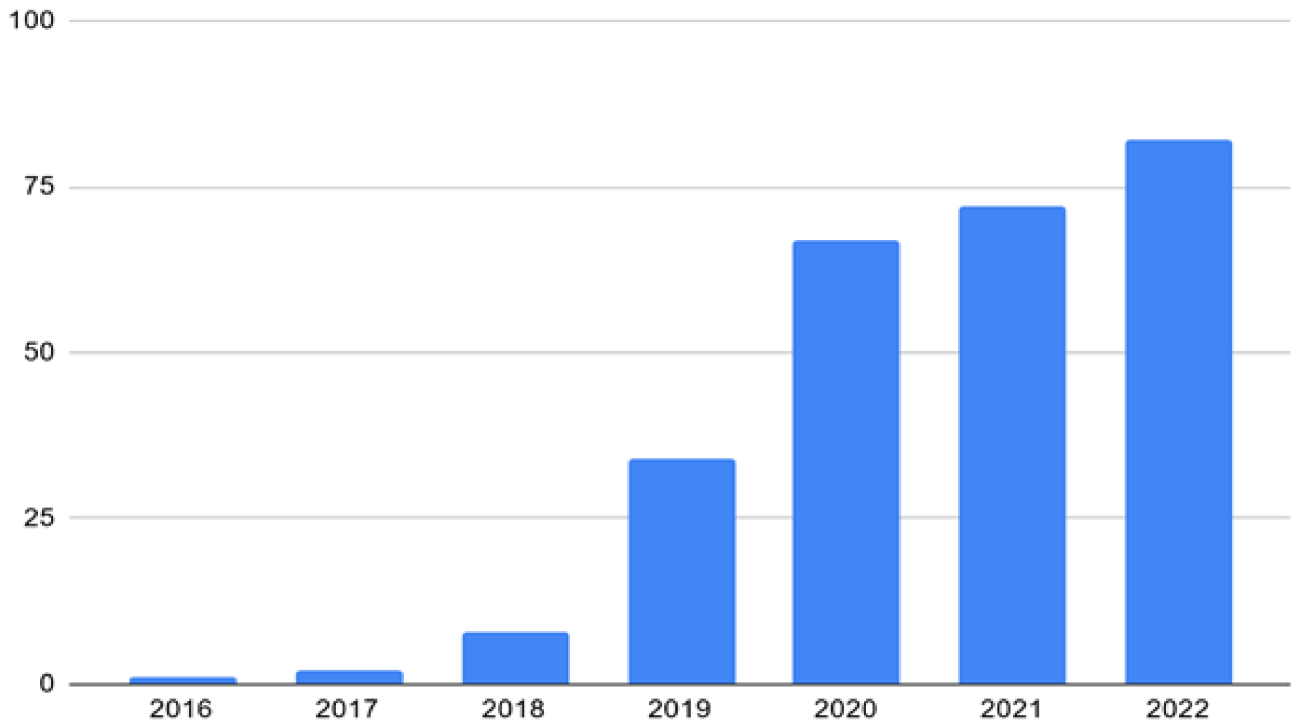


- More complex architectures means more complex hyperparameter tuning.

Different Neural Network Types in SCA

- Autoencoder.
- Recurrent neural network.
- Residual neural network.
- Generative Adversarial Network.

Deep Learning Publications

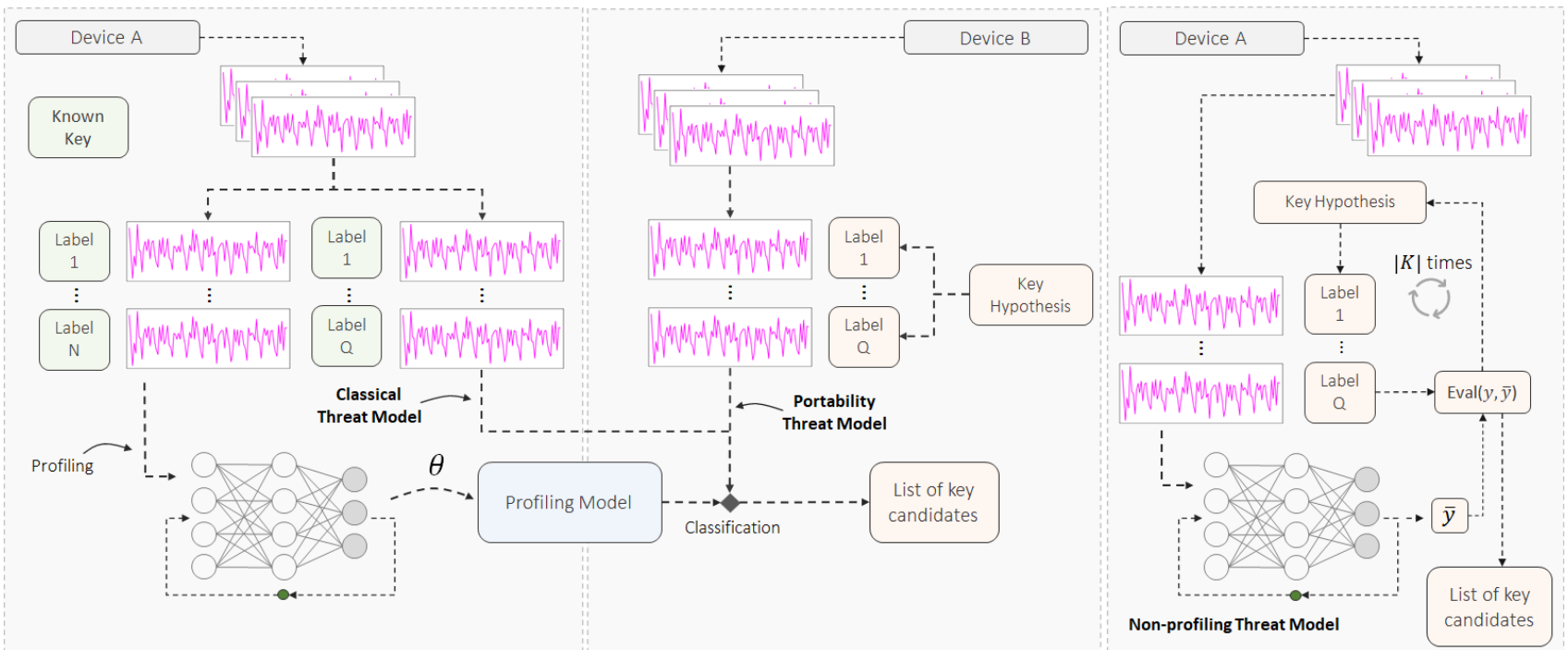


Outline

Datasets

Dataset	Platform	Traces (Features)	Keys	Implementation	Countermeasures
DPAv2 [100] (2010)	SASEBO GII (FPGA)	100 000 (3 253)	1 fixed key	AES128	None
DPAv4 [101] (2013)	Atmega (Software)	100 000 (435 000)	1 fixed key	AES256	First-order masking (RSM)
DPAv4.2 [102] (2014)	Atmega (Software)	80 000 (1 704 400)	16 different keys	AES128	First-order masking (RSM)
AES_HD [95] (2018)	SASEBO GII (FPGA)	50 000 (1 250)	1 fixed key	AES128	None
AES_HD_MM [30] (2014)	SASEBO GII (FPGA)	5 600 000 (3 125)	1 fixed key	AES128	BM + Affine Masking
AES_RD [25] (2009)	Atmel AVR (Software)	50 000 (3 500)	1 fixed key	AES128	Hiding (Random Delay Interrupt)
ASCADf [84] (2018)	Atmega (Software)	60 000 (100 000)	1 fixed key	AES128	First-order BM (XOR)
ASCADv1 [84] (2018)	Atmega (Software)	300 000 (250 000)	Random keys + 1 fixed key	AES128	First-order BM (XOR)
ASCADv2 [2] (2021)	STM32 (Software)	810 000 (1 000 000)	Random + 1 fixed key	AES128	Second-order BM + Hiding (Shuffling)
CHES_CTF 2018 [92] (2018)	STM32 (Software)	42 000 (650 000)	Random Keys + 3 fixed keys	AES128	First-order BM (XOR)
CHES_CTF 2020 [7] (2020)	Software/Hardware	†	Fixed/random	Clyde128	ISW masking ⋆
Portability [10] (2020)	Atmega (Software)	50 000 (600)	4 fixed keys	AES128	None
Ed25519 (WolfSSL) [113] (2019)	STM32 (Software)	6 400 (1 000)	Random Ephemeral keys	EdDSA	None
Curve25519 (μNaCl) [20] (2020)	STM32 (Software)	5 997 (5,500)	Random Ephemeral keys	EdDSA	CSWAP, Coord./Scalar Rand.
Curve25519 [111]	STM32 (Software)	300 (8 000)	Random keys	EdDSA	CSWAP, Coord./Scalar Rand.
Curve25519 [111]	STM32 (Software)	300 (1 000)	Random keys	EdDSA	CSPOINTER, Coord./Scalar Rand.

Threat Models



Tools

- Python.
- scikit-learn.
- TensorFlow/PyTorch.
- Keras.

Publicly Available Custom SCA Tools

- Brisfors and Forsmark developed a python-based tool called *DLSCA* that allows deep learning-based SCA `https://github.com/brisfors/DLSCA`.
- The tool allows running multilayer perceptron architecture for attacks on AES128 and plotting the results (key rank, guessing entropy).
- While the authors mention it is not difficult to add new functionalities, there has not been any development in the last few years.

SCAAML

- Google recently published their python-based deep learning framework for SCA called *SCAAML* `https://github.com/google/scaaml`.
- The framework is actively developed but offers (at the moment) limited functionality.
- The framework provides one CNN architecture designed to attack TinyAES (the architecture is tuned and the best-performing one over more than 1 000 tested ones).
- To evaluate the attack performance, it is possible to use the (average) key rank.

AISY Framework

- The AISY framework is intended for the deep learning-based SCA.
- **Easy to use**. AISY framework allows very easy execution of deep learning in profiling side-channel attacks. The framework is built on top of *Keras* library (integrated in *TensorFlow* library) and users familiar to basic *Keras*'s functionalities can easily extent the framework.
- **Integrated Database**. AISY framework comes with the option to store all analysis results in an SQLite database. Standard libraries are implemented in the framework, and users can easily add custom tables to the framework.

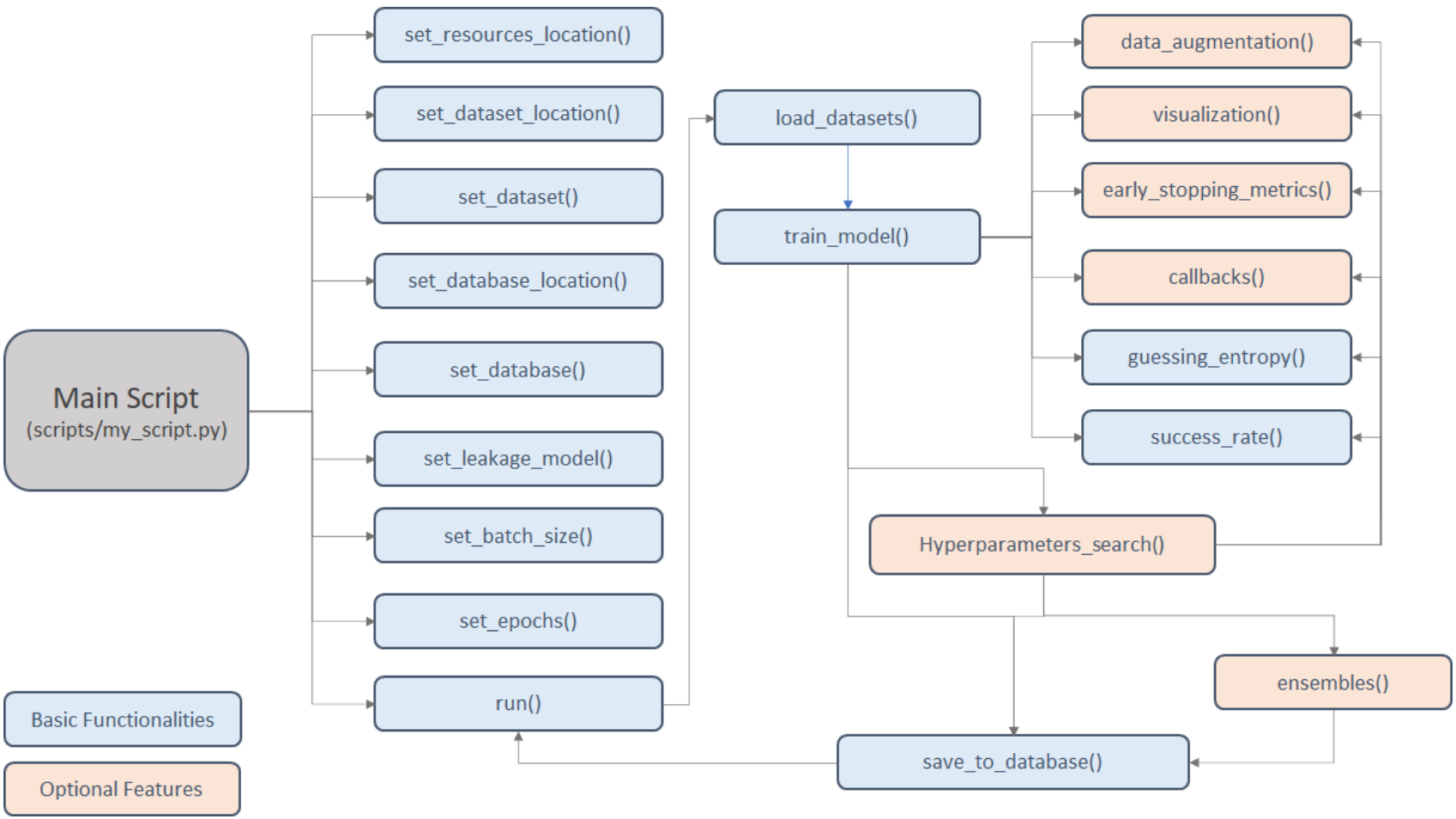
AISY Framework

- **Web application**. AISY framework is also integrated with *Flask* python-based web framework. A web application is integrated with a web-based user interface. The web application provides a user-friendly way to visualize analysis, plots, results, and tables.
- **One-click Script Generation**. A user can generate the full script used to produce results stored in the web application database.

AISY Framework

- **State-of-the-art side-channel analysis**. AISY framework brings state-of-the-art deep learning-based SCA. Version 0.1 already provides state-of-the-art features like ensembles, hyperparameter search, visualization, data augmentation, and special SCA metrics.
- **Teamwork**. AISY framework allows easy and efficient teamwork. The structure is done so that multiple users can easily share results and contribute together to analysis.
- `https://github.com/AISyLab/AISY_Framework`

AISY Framework



ML vs. SCA Metrics

- Training process is assessed based on ML metrics while in the attack phase we care about SCA metrics.
- Does good ML performance mean good SCA performance?
- How about poor ML performance?
- Can we use SCA metrics in the training phase?

Hyperparameters

- MLP: activation functions, number of dense layers, number of neurons in dense layers, regularization, learning rate, optimizer, loss function, number of epochs, batch size, initialization.
- CNN: number of convolutional layers, number and size of kernels, activation functions, pooling type and size, stride, number of dense layers, number of neurons in dense layers, regularization, learning rate, optimizer, loss function, number of epochs, batch size, initialization.

Hyperparameter Tuning

- Hyperparameter tuning is extremely important.
- Different algorithms have different hyperparameters.

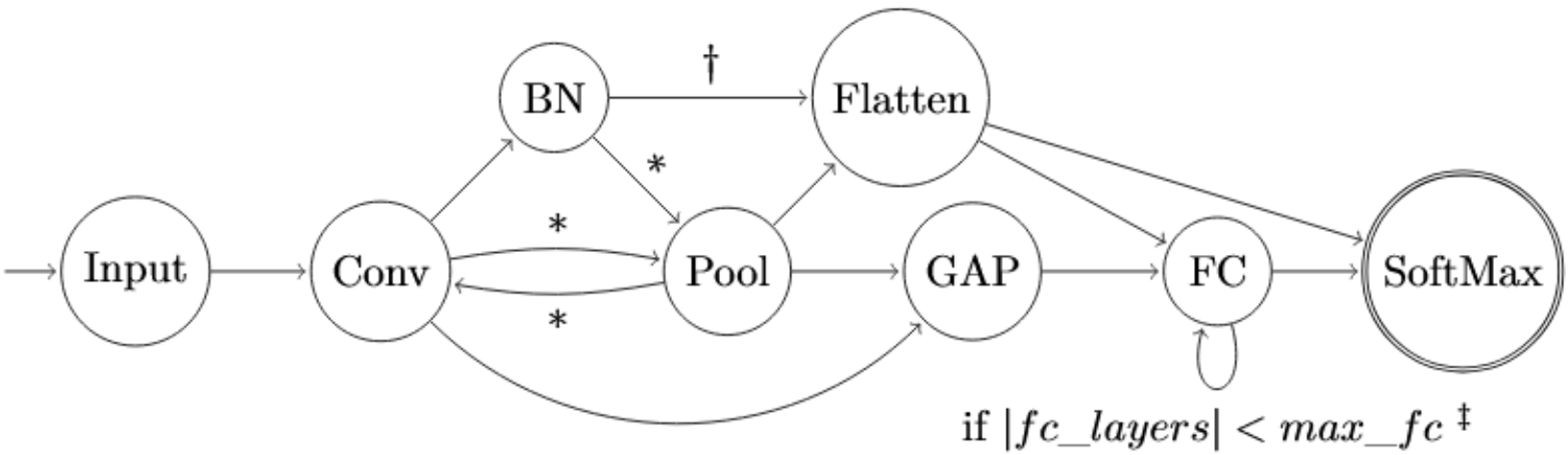
- General design principles.
- Methodologies.
- Random search.
- Grid search.
- Advanced techniques.

Methodologies

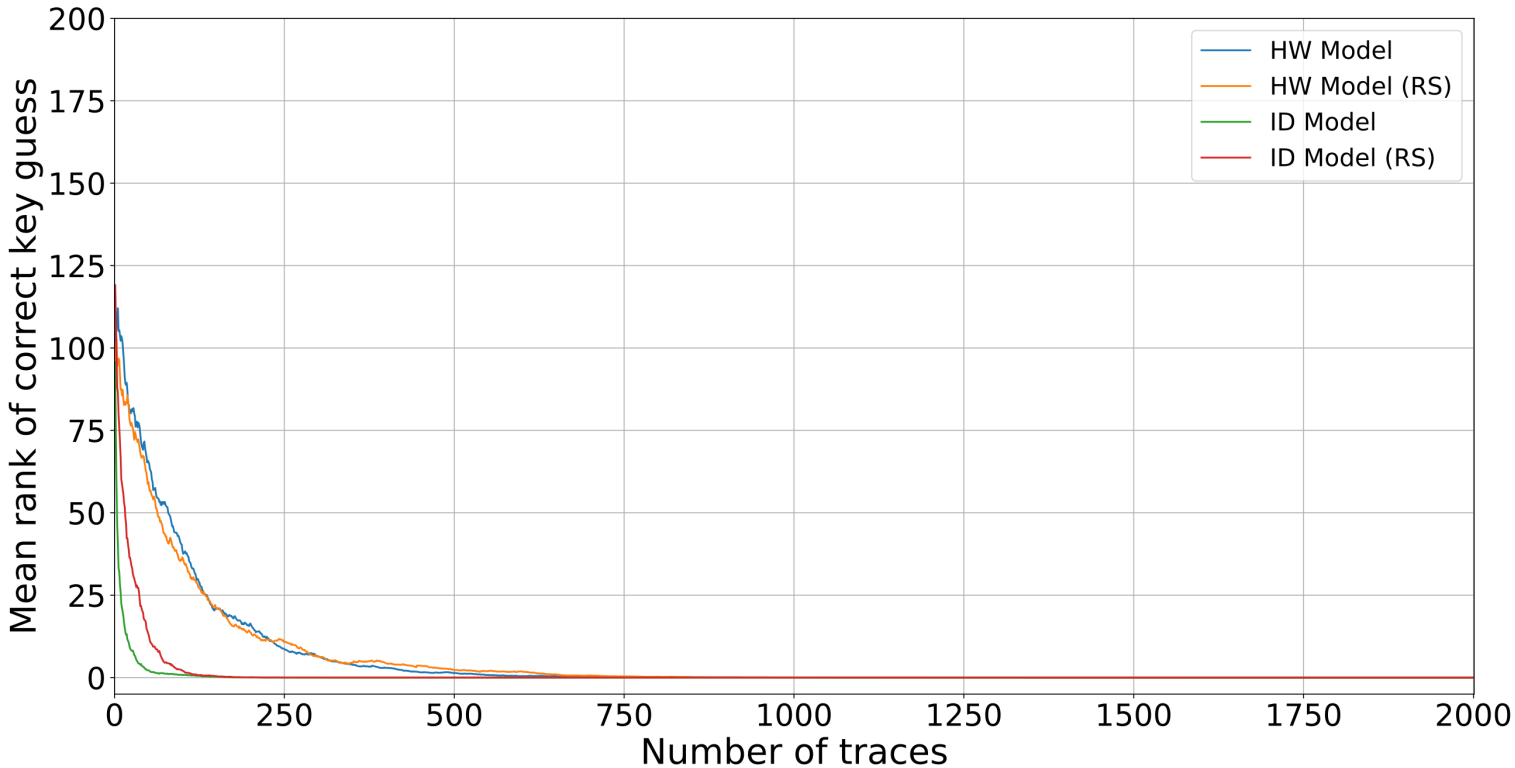
- Zaid, G., Bossuet, L., Habrard, A., & Venelli, A. (2019). Methodology for Efficient CNN Architectures in Profiling Attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(1), 1-36. https://doi.org/10.13154/tches.v2020.i1.1-36.
- Wouters, L., Arribas, V., Gierlichs, B., & Preneel, B. (2020). Revisiting a Methodology for Efficient CNN Architectures in Profiling Attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(3), 147-168. https://doi.org/10.13154/tches.v2020.i3.147-168.

Reinforcement Learning

- Reinforcement learning attempts to teach an agent how to perform a task by letting the agent experiment and experience the environment, maximizing some reward signal.
- `https://github.com/AISyLab/Reinforcement-Learning-for-SCA`



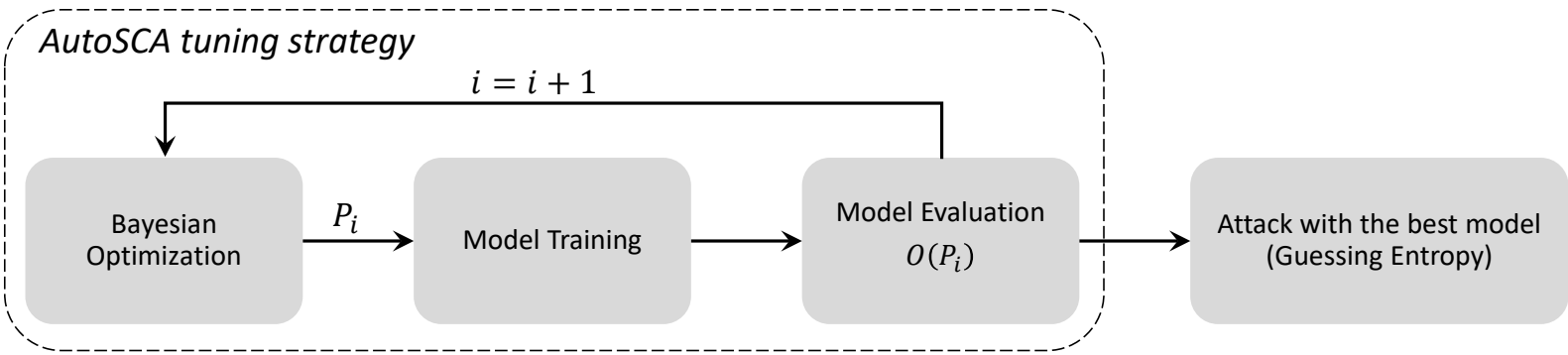
Reinforcement Learning



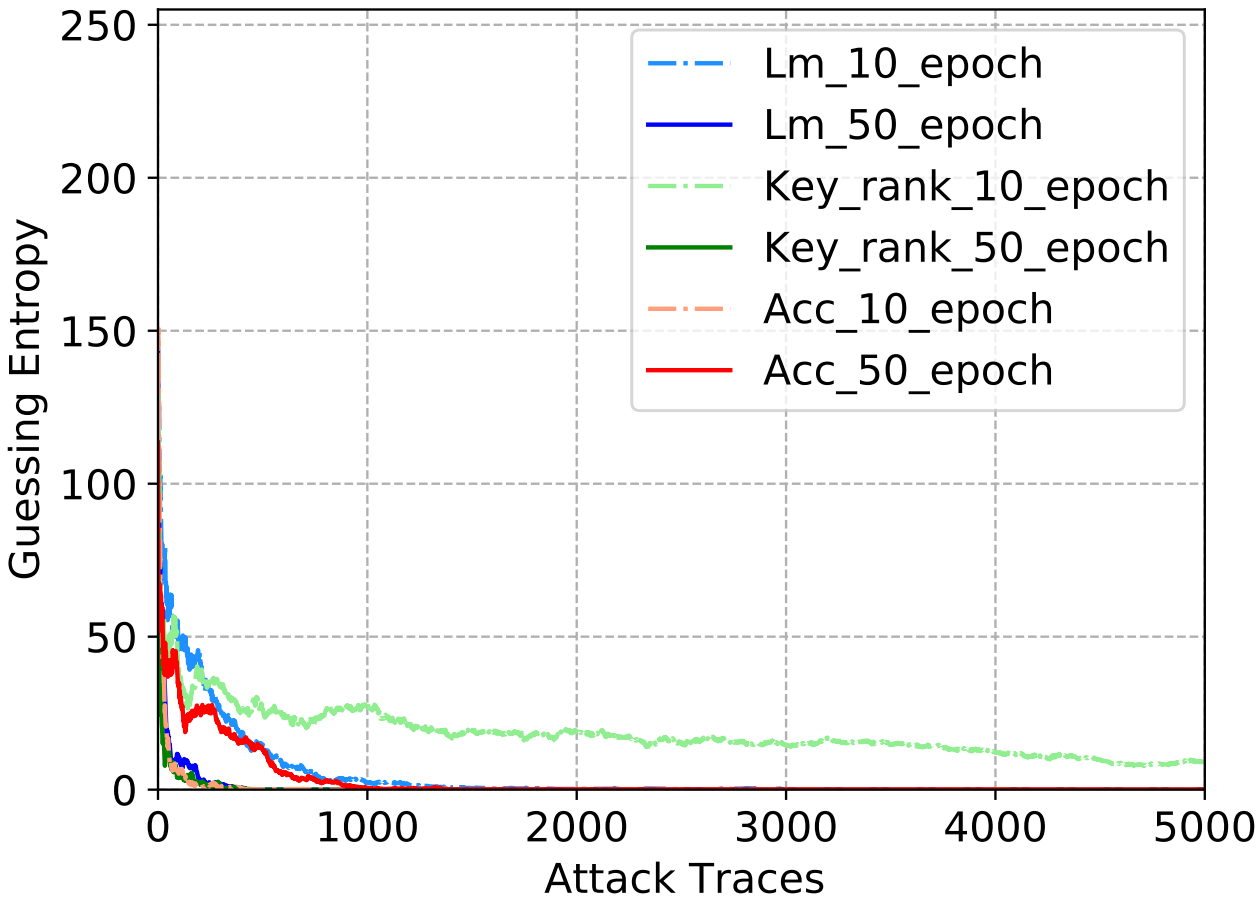
Bayesian Optimization

- In Bayesian optimization, the aim is to build a probabilistic model of the underlying function.
- We first require a probabilistic model of a function (often referred to as the surrogate model), where there are several ways to model it.
- Second, we require an acquisition function for Bayesian optimization to generate the next neural network architecture to observe, i.e., to select what point to sample next.

Bayesian Optimization



Bayesian Optimization



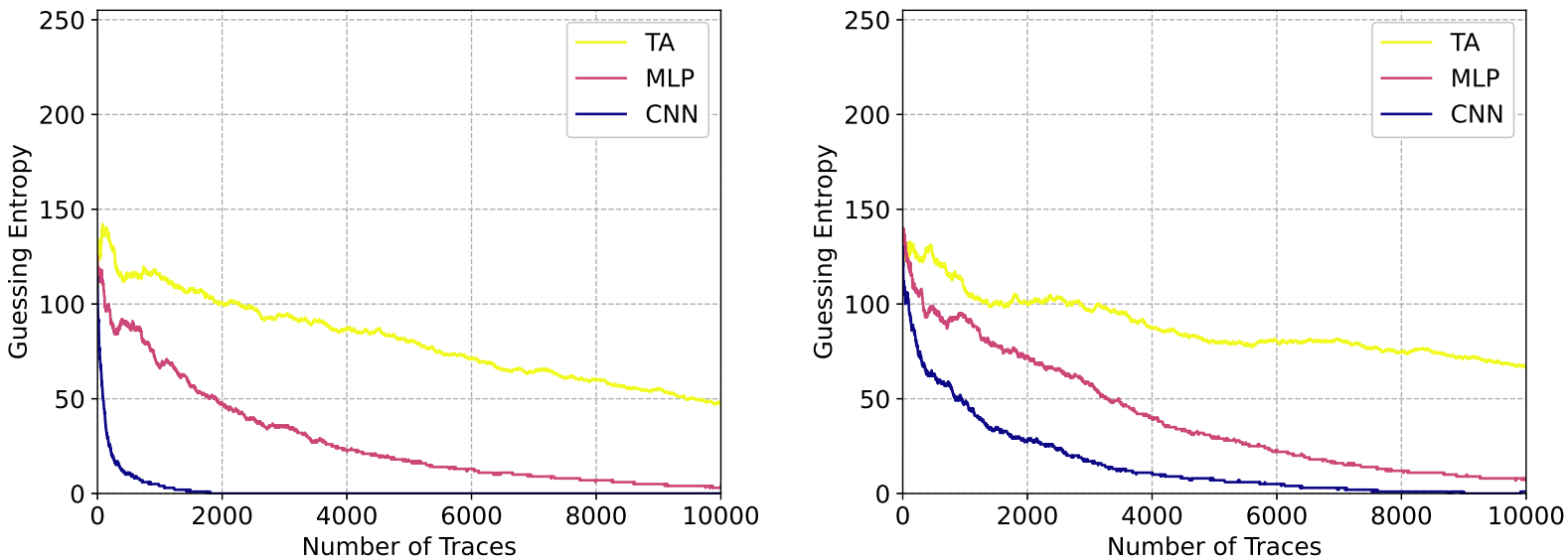
Countermeasures as Noise

- Various countermeasures make SCAs significantly more complex, and such countermeasures can be further combined to make the attacks even more challenging.
- Could we consider those as noise and use some techniques to remove the noise?
- We propose a new approach to remove several common hiding countermeasures with a denoising autoencoder.
- Gaussian noise, random delay interrupts, desynchronization, jitter, shuffling.

Denoising Autoencoder

- An autoencoder consists of two parts: encoder (ϕ) and decoder (ψ).
- The goal of the encoder is to transfer the input to its latent space $\mathcal{F}$, i.e., $\phi : \mathcal{X} \rightarrow \mathcal{F}$.
- The decoder, on the other hand, reconstructs the input from the latent space, which is equivalent to $\psi : \mathcal{F} \rightarrow \mathcal{X}$.
- When applying the autoencoder for the denoising purpose, the input and output are not identical but represented by noisy-clean data pairs.

Denoising Autoencoder

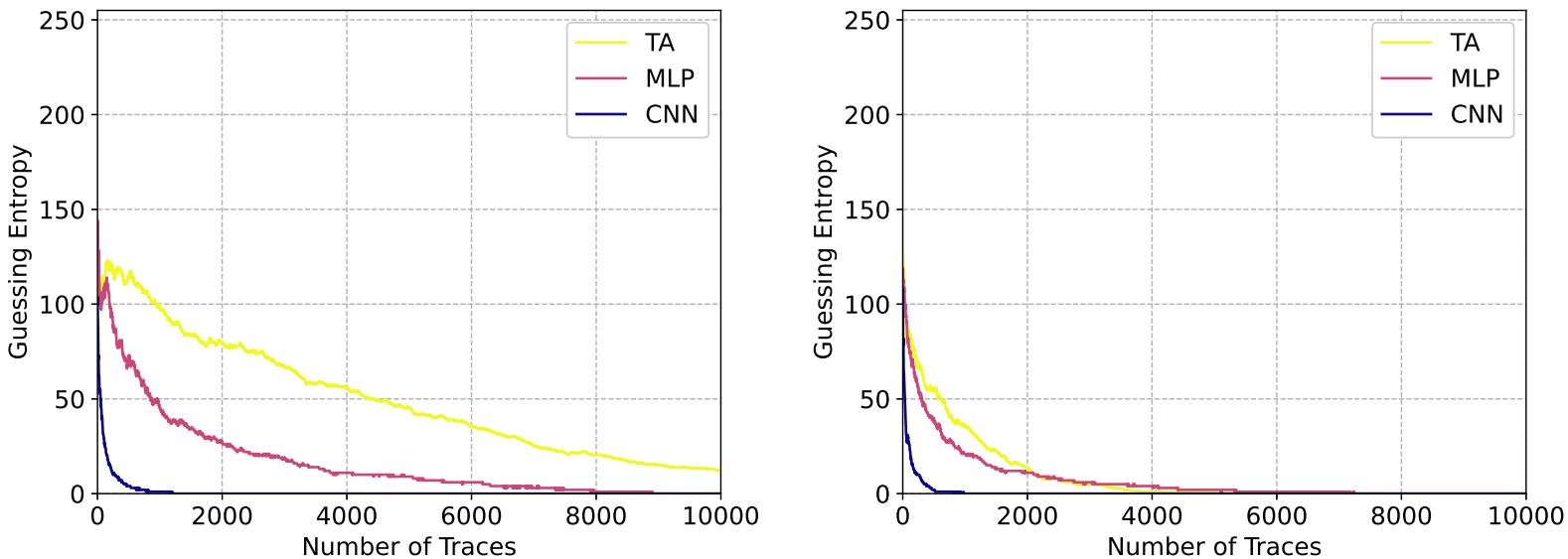


(a) GE: denoise with averaging.

(b) GE: denoise with CAE.

Figure: Guessing entropy: denoising Gaussian noise with averaging (a) and CAE (b).

Denoising Autoencoder



(a) GE: denoise with static alignment.

(b) GE: denoise with CAE.

Figure: Guessing entropy: denoising desynchronization with static alignment (a) and CAE (b).

Generalization of Function Approximation

- While we use machine learning metrics to drive the training, we are interested in results as observed through SCA metrics.
- Ideally, we should always train a neural network until it achieves the maximum quality in generalization to the validation set.
- Underfitting, generalization, and overfitting phases.

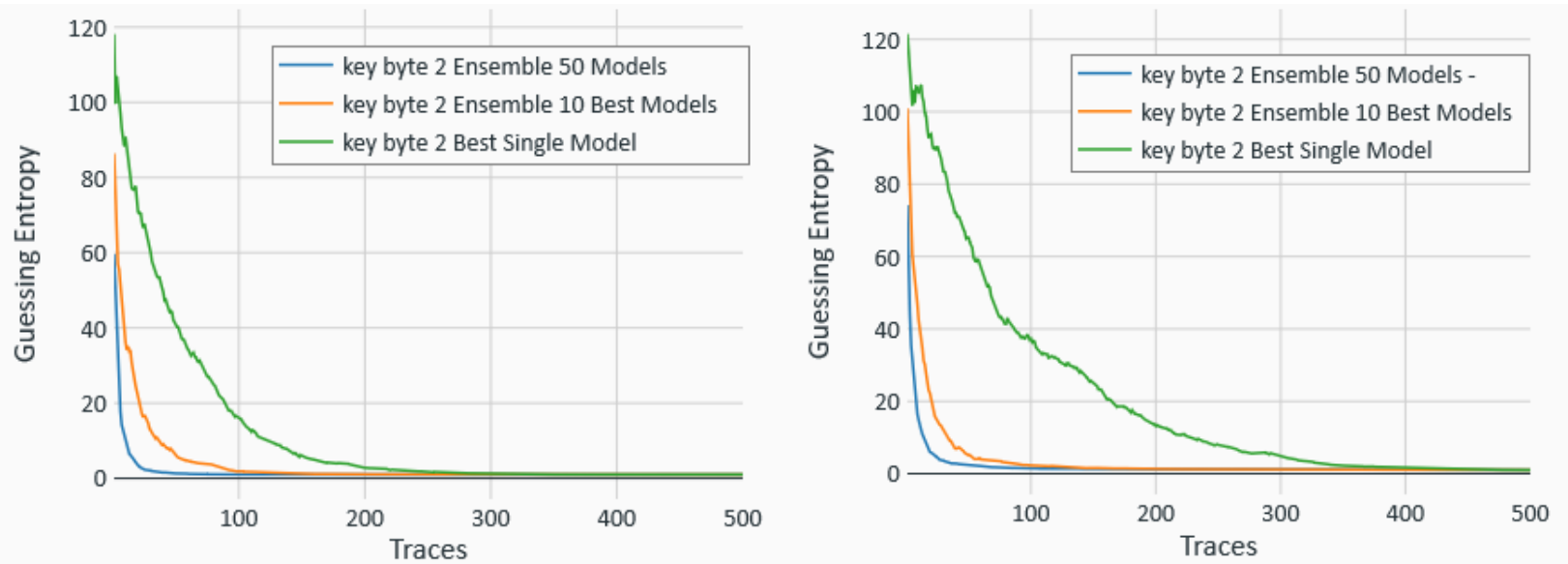
Generalization of Function Approximation

- In SCA, the generalization phase is directly related to the key recovery, and it may start very soon after the training starts because a low accuracy can already represent the turning point from underfitting to generalization.
- Can a low accuracy (sometimes close to random guessing) still be associated with this *good enough* generalization phase?

How to Improve Generalization?

- There are many ways to improve generalization (more powerful classification methods, better hyperparameter tuning, regularization, etc.).
- We can also do something simpler!
- Commonly, in the experimental phase, one runs a number of evaluations to find the best hyperparameters.
- Can we somehow use multiple results?
- It sounds reasonable to take the most out of the hyperparameter tuning phase and explore whether one can use more than a single machine learning model obtained during the tuning phase.

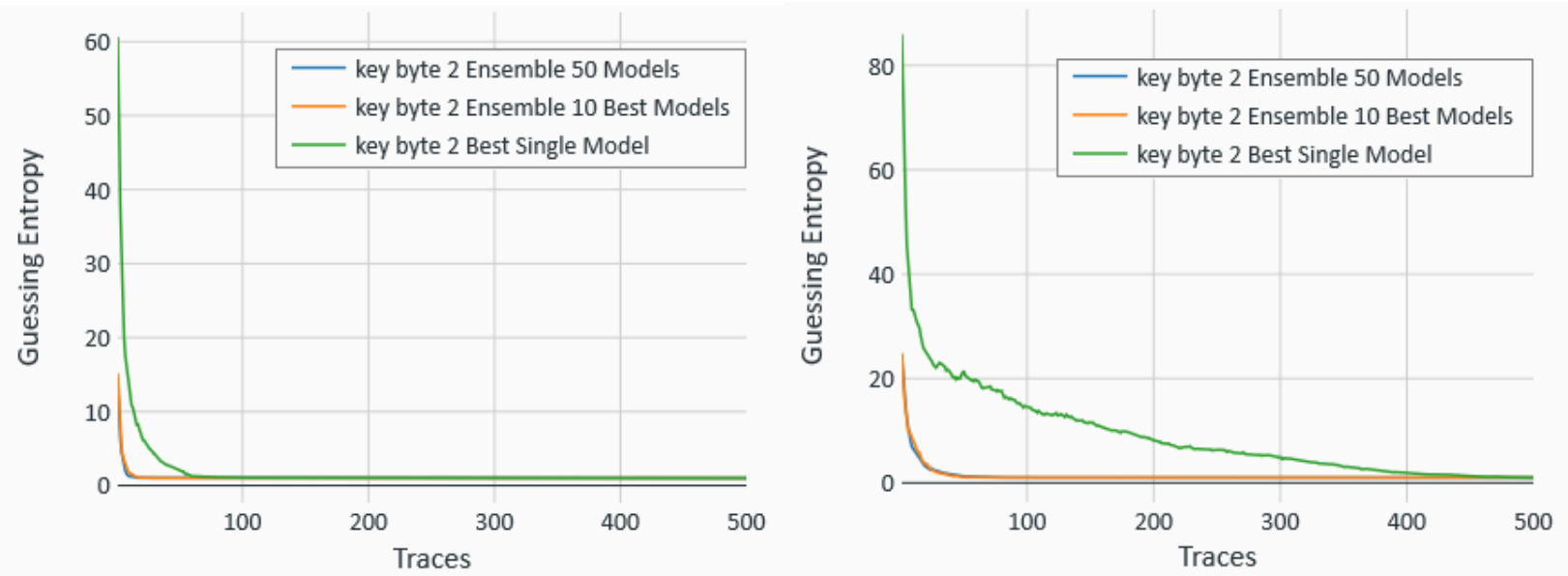
Deep Learning Ensembles



(a) MLP results. (b) CNN results.

Figure: Guessing entropy for ASCAD for the Hamming weight leakage model.

Deep Learning Ensembles



(a) MLP results. (b) CNN results.

Figure: Guessing entropy for ASCAD for the identity leakage model.

Outline

Important Hyperparameters

- MLP: activation functions, number of dense layers, number of neurons in dense layers, regularization, learning rate, optimizer, loss function, number of epochs, batch size, initialization.
- CNN: Number of convolutional layers, number and size of kernels, activation functions, pooling type and size, stride, number of dense layers, number of neurons in dense layers, regularization, learning rate, optimizer, loss function, number of epochs, batch size, initialization.

Many Things Actually Work

- Simple hyperparameter search.
- Data Augmentation.
- Various types of architectures.
- Small architectures.
- Custom metrics and neural network elements.
- ...

Feature Selection for Deep Learning SCA

Scenario	Knowledge of r mask share	POI selection and pre-processing	Noisy/non-leaking samples
RPOI	Yes	Main SNR peaks of r and s_r. No pre-processing required.	No
OPOI	Yes	Minimum trace interval including SNR peaks of r and s_r. No pre-processing required.	Reduced
NOPOI	No	No POI selection and pre-processing is required.	All available

Table: Possible feature selection scenarios for deep learning-based SCA with the synchronized measurements.

Feature Selection for Deep Learning SCA

Dataset	RPOI	OPOI	NOPOI	Total
ASCADf	up to 1 000 SNR peaks from NOPOI interval	[45 400, 46 100]	[0, 100 000]	100 000
ASCADr	up to 1 000 SNR peaks from NOPOI interval	[80 945, 82 345]	[0, 250 000]	250 000
DPAv4.2	up to 1 000 SNR peaks from NOPOI interval	[170 000, 174 000] + [206 000, 210 000]	[250 000, 400 000]	1 700 000
CHES CTF	-	[0, 10 000] + [120 000, 150 000]	[0, 150 000]	650 000

Table: Selected intervals for each feature selection scenario. '-' denotes that we did not explore that specific setting.

Feature Selection for Deep Learning SCA

Table: Points of interest, minimum number of attack traces to get guessing entropy equal to 1, model search success (when GE=1), and number of trainable parameters for all datasets and feature selection scenarios.

Dataset	Neural Network Model	Feature Selection Scenario	Amount of POIs (HW/ID)	Attack Traces (HW/ID)	Search Success (%) (HW/ID)	Trainable Parameters (HW/ID)
ASCADf	MLP	RPOI	200/100	5/**1**	99.22%/96.86%	82 209/429 256
ASCADf	CNN	RPOI	400/200	5/**1**	99.23%/99.08%	499 533/158 108
ASCADf	MLP	OPOI	700/700	480/104	82.80%/68.80%	16 309/10 266
ASCADf	CNN	OPOI	700/700	744/87	55.53%/35.33%	594 305/62 396
ASCADf	MLP	NOPOI	2 500/2 500	7/**1**	74.50%/39.00%	2 203 009/5 379 256
ASCADf	CNN	NOPOI	10 000/10 000	7/ **1**	15.40%/2.45%	545 693/439 348
ASCADf	CNN	NOPOI desync	10 000/10 000	532/36	2.44%/2.64%	268 433/64 002
ASCADr	MLP	RPOI	200/20	3/**1**	99.23%/100%	565 209/639 756
ASCADr	CNN	RPOI	400/30	5/**1**	100%/100%	575 369/636 224
ASCADr	MLP	OPOI	1 400/1 400	328/129	71.40%/37.25%	31 149/34 236
ASCADr	CNN	OPOI	1 400/1 400	538/78	47.92%/23.95%	270 953/87 632
ASCADr	MLP	NOPOI	25 000/25 000	6/ **1**	44.39%/7.02%	5 243 209/12 628 756
ASCADr	CNN	NOPOI	25 000/25 000	7/ **1**	19.17%/4.35%	369 109/721 012
ASCADr	CNN	NOPOI desync	25 000/25 000	305/73	0.71%/1.04%	22 889/90 368

Introduction

- There are already few settings that consider unsupervised deep learning-based SCA.
- First approach (DDLA) is by B. Timon.
- While it works, the attack performance is not great and the computational complexity is high.

Plaintext/Ciphertext-based Non-profiling SCA

- Supervised deep learning-based SCA learns a mapping based on known plaintexts and keys.
- Then, the adversary estimates the conditional probability given a leakage trace with the unknown key.
- In unsupervised setting, we do not know the key.
- But, the key is commonly fixed for all traces.

Plaintext/Ciphertext-based Non-profiling SCA

- Supervised deep learning-based SCA learns a mapping based on known plaintexts and keys.
- Then, the adversary estimates the conditional probability given a leakage trace with the unknown key.
- In unsupervised setting, we do not know the key.
- But, the key is commonly fixed for all traces.
- The label $\mathsf{I}(k, d_i)$ and d_i would satisfy:

$$d_i \longmapsto \mathsf{I}(k, d_i). \tag{1}$$

Plaintext/Ciphertext-based Non-profiling SCA

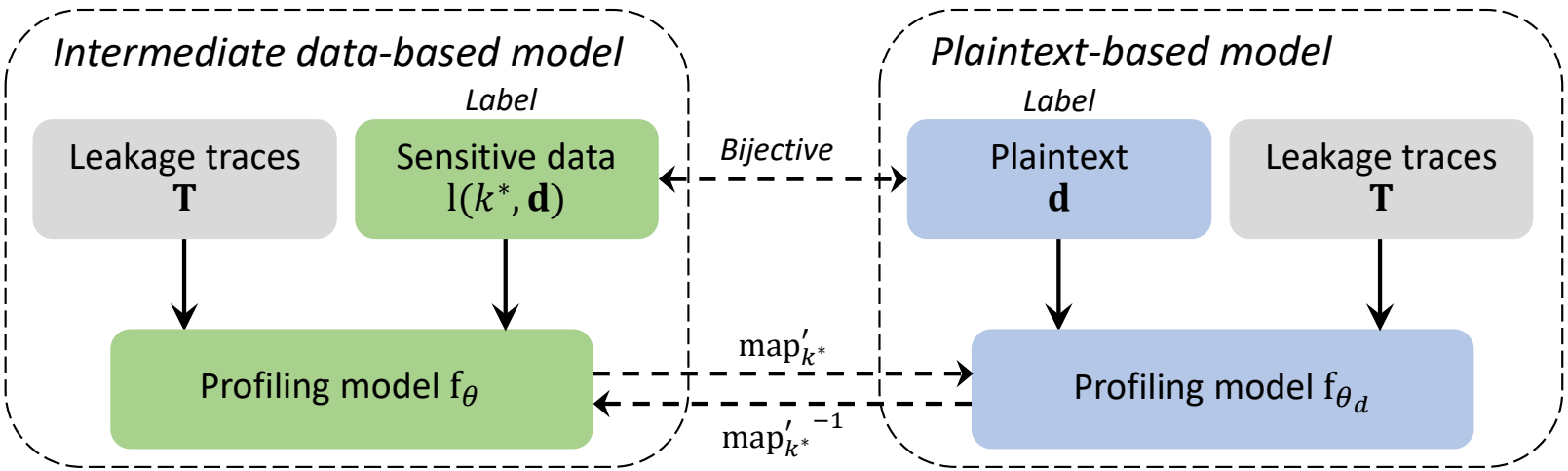


Figure: The relationship between intermediate data-based model and plaintext-based model.

Plaintext/Ciphertext-based Non-profiling SCA

- For supervised DLSCA, if a profiling model is generalized well on the leakage traces, the probability of the incorrect value is closely correlated with the correct label.
- In unsupervised setting, we can still estimate the label distance, providing us with plaintext/ciphertext distribution.

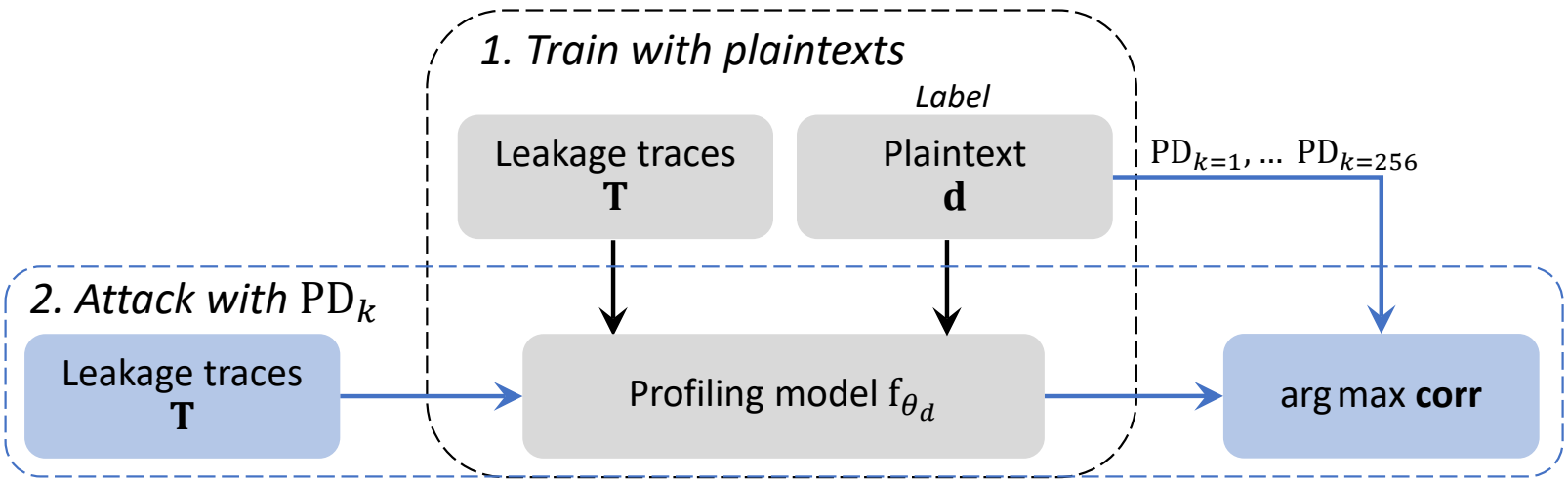


Figure: Attack scheme of the Plaintext Labeling Deep Learning (PLDL).

Plaintext/Ciphertext-based Non-profiling SCA

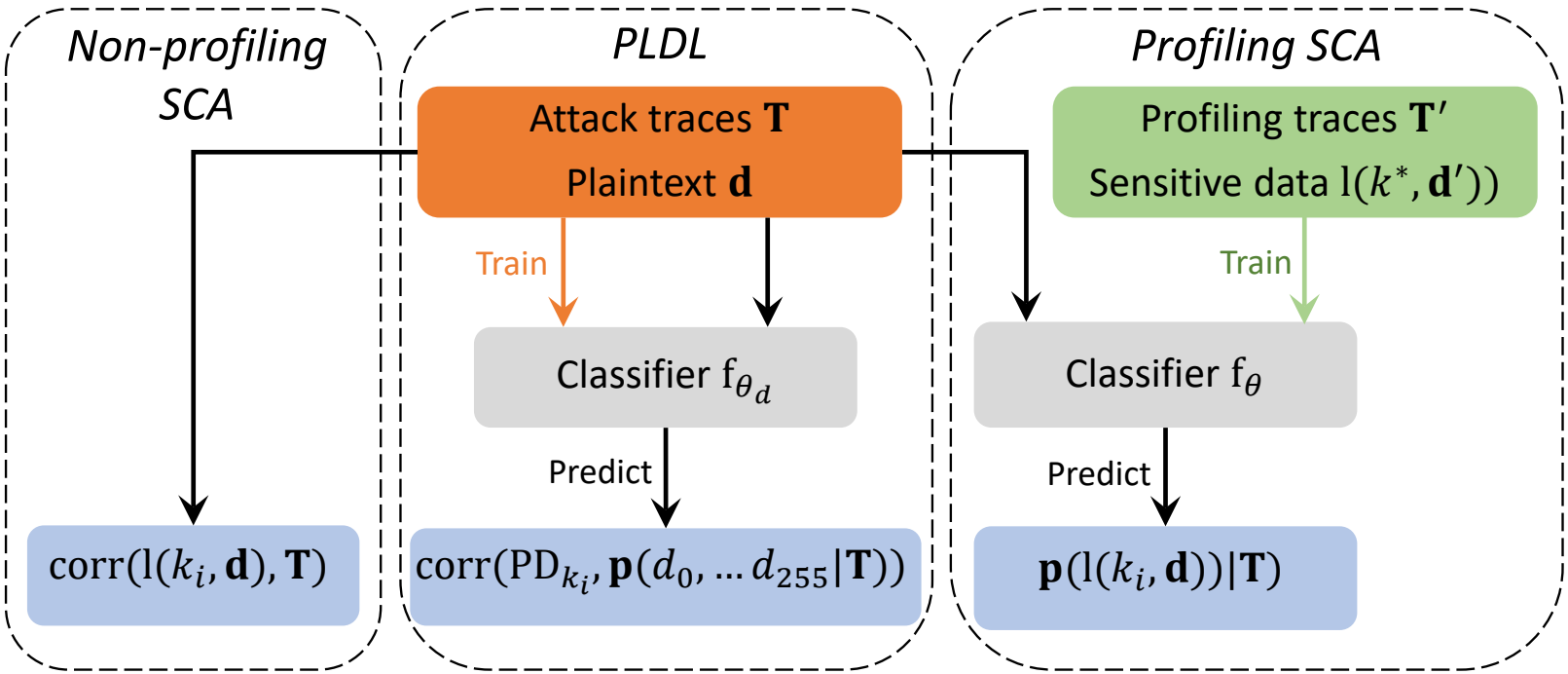


Figure: A demonstration of non-profiling SCA, PLDL, and profiling SCA.

Plaintext/Ciphertext-based Non-profiling SCA

Table: Performance benchmark with non-profiling attacks.

Dataset	CPA	MOR	DDLA	PLDL
ASCAD_F	KR161/KR47	1 957/638	KR7/309	**8**/**111**
ASCAD_R	KR64/KR8	KR28/KR9	27 266/KR48	**20**/**19**
CHES_CTF	KR139/KR220	KR6/KR31	KR54/KR85	**6 121**/**KR2**
AES_RD	KR2/KR31	KR33/3 112	2 541/KR2	**1**/**57**
AES_HD	KR19/KR145	5 593/KR10	KR26/KR20	**60**/**KR6**

Some More Things

- Explanaibility.
- Custom loss functions.
- Portability.

Outline

Raw Data

- Most of the targets are software implementations offering limited countermeasures (e.g., first-order masking and simulation of misalignment).
- Break higher-order masking dataset without knowing the shares.

Data Pre-processing

- We are missing a clear set of guidelines on what pre-processing techniques to use and in what settings.
- Considering data augmentation techniques, we can recognize two directions: either use standard machine learning techniques or customize data augmentation for SCA. There is a need for a systematic comparison of those approaches.
- If we know something about the traces, why not to use this knowledge?

Feature Engineering

- Feature engineering is not needed for deep learning-based SCA, or we need only very basic techniques. As using side-channel traces with thousands (or tens of thousands) features is common, we must investigate the possible drawbacks of using extremely lengthy traces.
- If feature engineering is required, we must understand what techniques to use.

Algorithm Selection

- As research papers consider relatively small datasets, there are no efficient guidelines to determine the hyperparameters on more realistic settings containing millions of noisy and protected side-channel traces.

Model Training

- Recognize the most important hyperparameters for deep learning-based SCA.
- Evaluate how custom neural networks can enhance the attack performance and generalize for different settings.
- As one of the dominant problems in DL-SCA is overfitting, it is necessary to investigate how to prevent or, at least reduce it.

Attack Evaluation

- Little is understood about the relationship between commonly used SCA metrics (GE, SR) and model-learned parameters (i.e., the neural network weights).
- It is unlikely to find a universal profiling model that could defeat all types of available countermeasures and that could be used in a wide variety of targets. We should measure and understand how the selected hyperparameters make the model succeed (or fail) in fitting existing leakages.

AI Explainability and SCA

- Understand how neural networks process masking countermeasures.
- Propose efficient countermeasures based on the AI explainability that are tuned to fight against deep learning-based SCA.

Outline

Conclusions

- Deep learning is efficient and powerful option for SCA.
- Current results are very promising as we can break protected targets even with very small architectures.
- What is less clear is how would the approach scale for more realistic targets.
- It is not easy to select the most promising approaches from all that is available.
- The big challenges are unsupervised deep learning-based SCA and explainable AI for SCA.