Train Wisely: Multifidelity Bayesian Optimization Hyperparameter Tuning in Deep Learning-based Side-Channel Analysis

Trevor Yap, Shivam Bhasin, Léo Weissbart

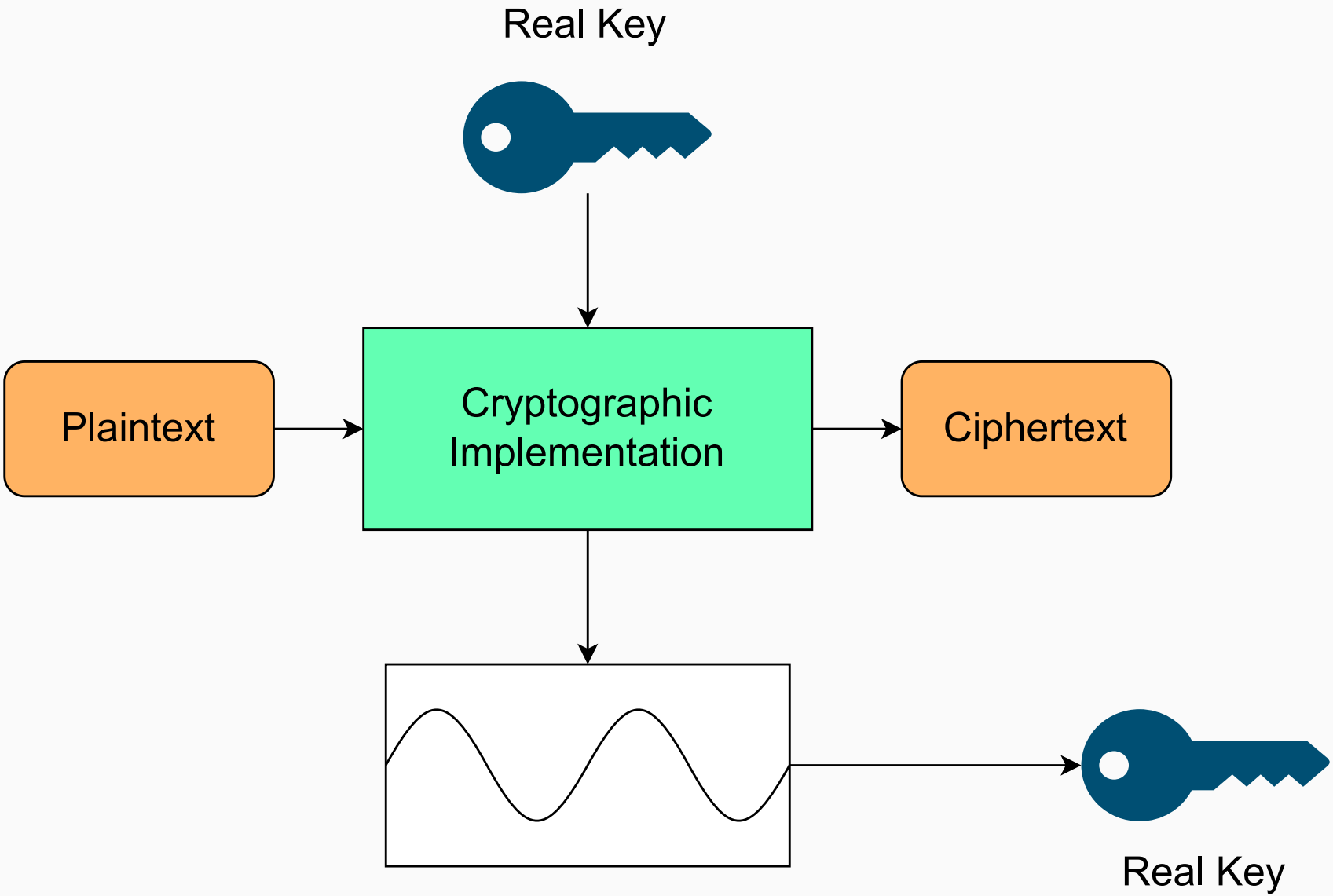
Radboud Universiteit

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

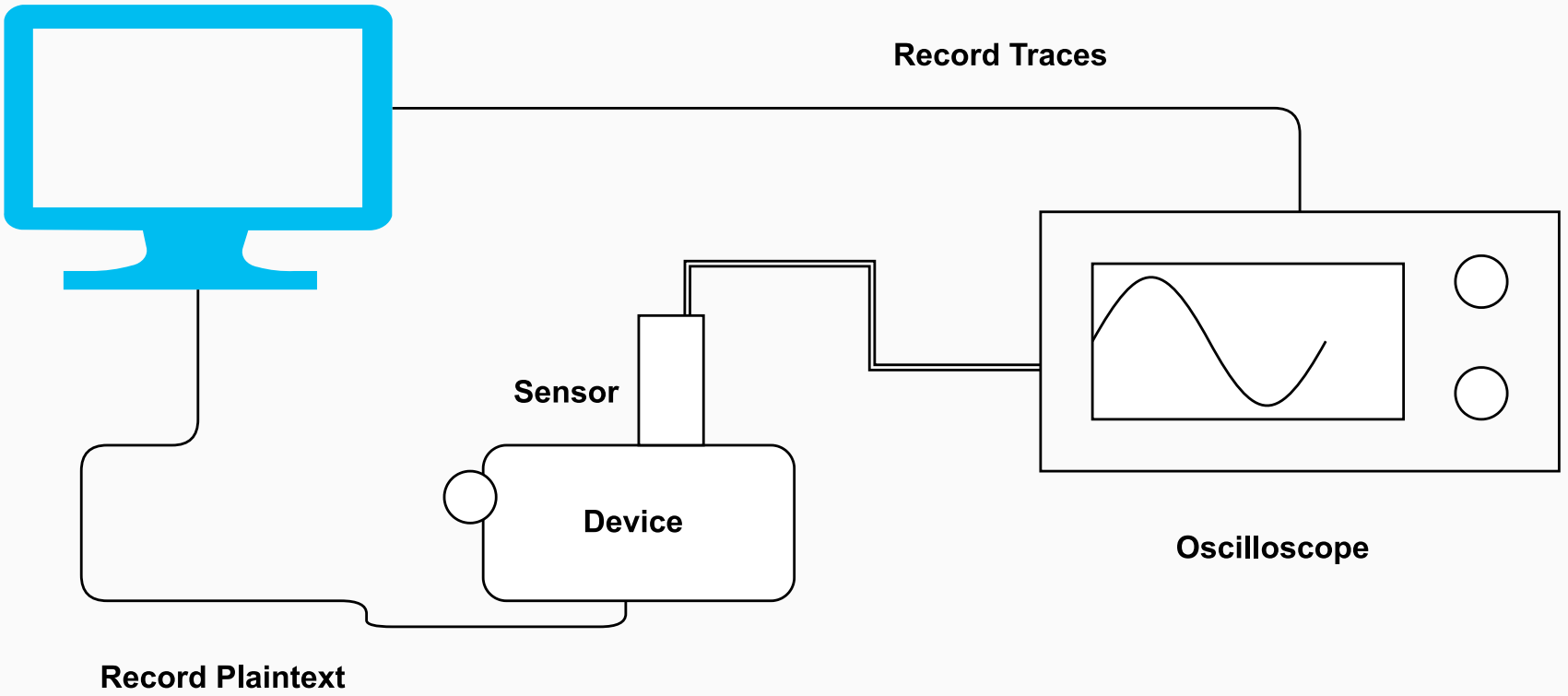
Table of contents

Table of contents

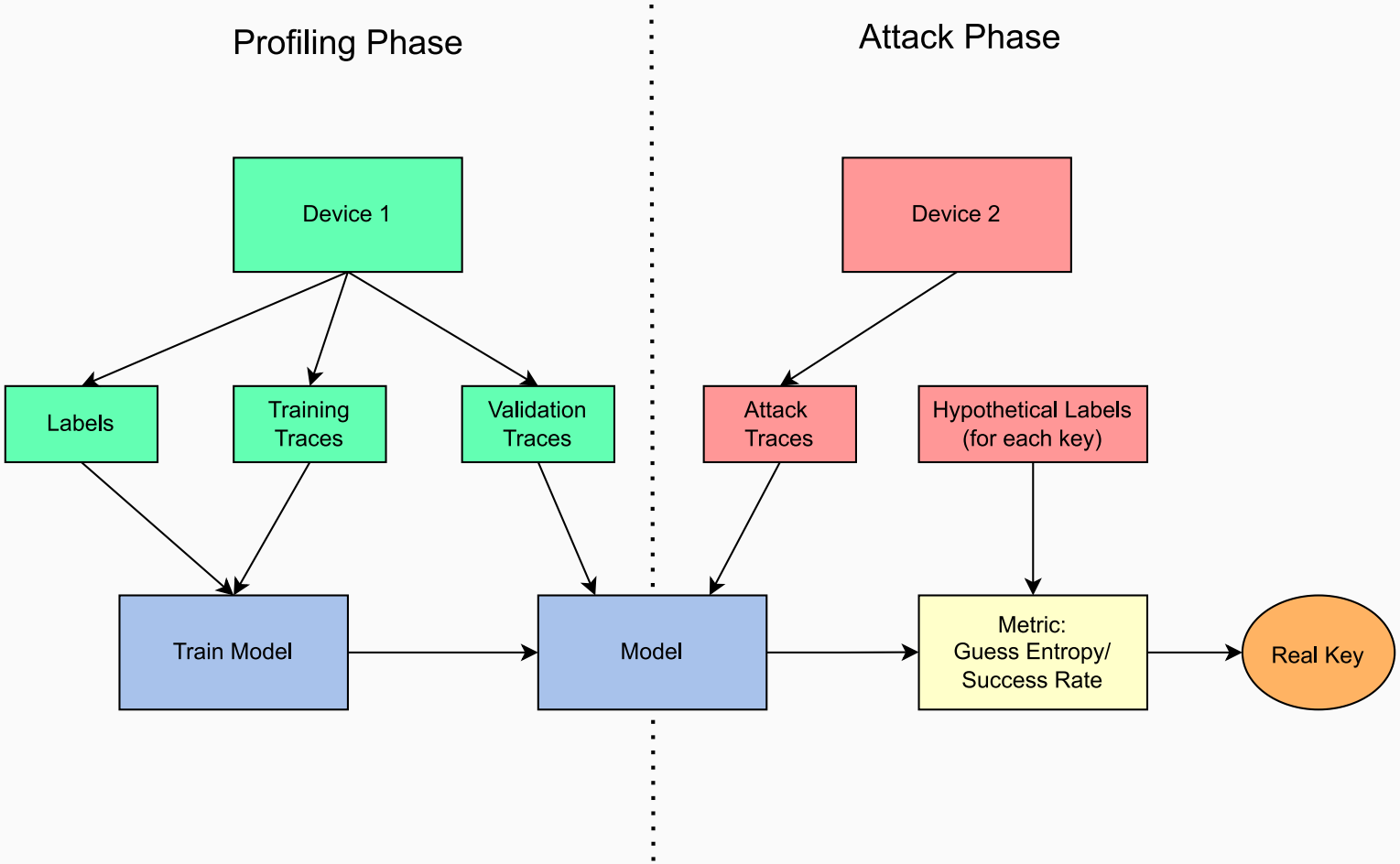
Overview of side channel analysis (SCA)



Overview of side channel analysis (SCA)



Profiling Attack

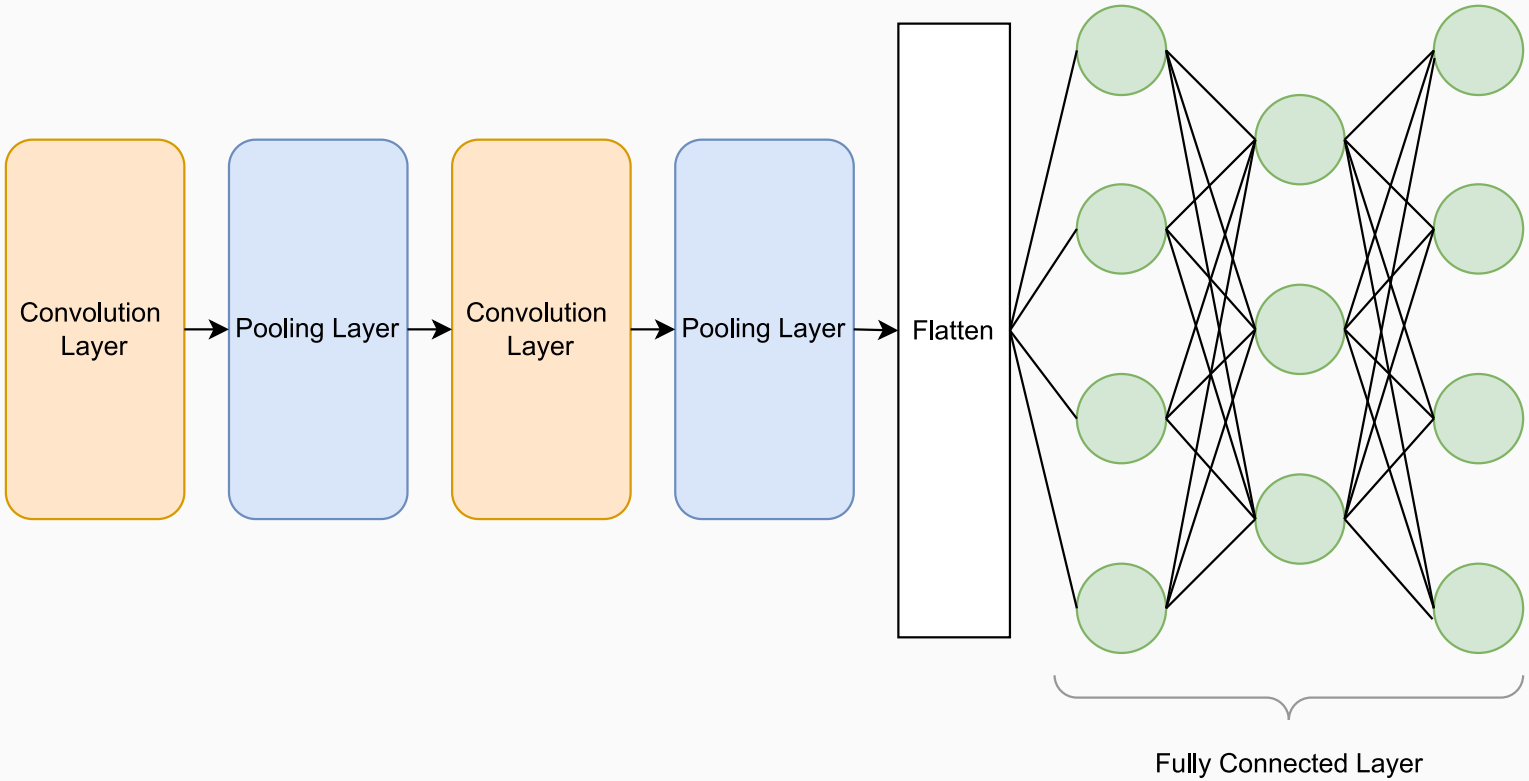


We denote $GE = 0$ if the attack is successful.

We define NTGE to be the number of traces required for $GE = 0$.

A typical model used are the template attack or deep neural networks.

Deep Neural Network (DNN)



DNNs are used as classifiers.

Common DNNs are like Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs)

Motivation

- DNNs were shown to **outperform classical SCA** even in the presence of countermeasures.
- Introduced a **large number of hyperparameters** to tune (e.g., the number of layers, kernel size, type of activation functions, etc.) compared to other machine learning or classical SCA.
- Maghrebi et al. [1] have pointed out that the **performance** of DNNs is **greatly influenced by their hyperparameters**. This pushes for the need for methodologies to find good hyperparameters in the domain of SCA.

Related Works

Manual Hyperparameter Tuning:

- [3] and [4] provided guidelines and offer a more precise methodology that helps to generate smaller and well-performing DNNs manually.

Automatic Hyperparameter Tuning:

- Bayesian Optimization [5],
- Reinforcement Learning [6],
- Evolutionary Algorithm [7].

Most techniques are slow and could run for days.

Motivation

- Due to the large number of IT products to be evaluated, evaluating the security of these products in evaluation labs becomes very **time-sensitive.**
- **Resources such as time are valuable assets** to determine a device's security.
- An evaluator will naturally **set a budget for any resources** like the time needed to quantify the security of the primitive tested.

*Are there automated tools available can produce comparable results while **allocating resources more efficiently**?*

Main Contribution

Multifidelity optimization methods allow speed up in the optimization process by:

- **allocating more resources to promising configurations**
- **stopping evaluations of poorly performing ones early**.

Explore multifidelity optimization method known as Bayesian Optimization HyperBand (BOHB) search for hyperparameters.

Table of contents

Hyperparameter Optimization Problem (HPO)

- The performance score/objective function of a model is

$$f : \Theta \to \mathbb{R}$$

  with $\boldsymbol{\theta} \in \Theta$ as their hyperparameters and Θ is a predefined space that an expert with prior knowledge.
- **Problem:** search for $\boldsymbol{\theta}^*$ such that it satisfies

$$\arg\min_{\boldsymbol{\theta} \in \Theta} f(\boldsymbol{\theta}).$$

Note: f here is not the neural network itself, but the performance score based on the neural network with $\boldsymbol{\theta}$ as their hyperparameters.

Three components of BOHB

- Successive Halving
- HyperBand
- Bayesian Optimization

Successive Halving

- Developed by Jamieson et al. [8].
- Multi-armed bandit strategy,
- Evaluated the configurations' performance based on the budget b. Then it continues to evaluate the performance of the top η^{-1} configurations on a η times larger budget until the maximum budget is attained. Recommended to set $\eta = 3$

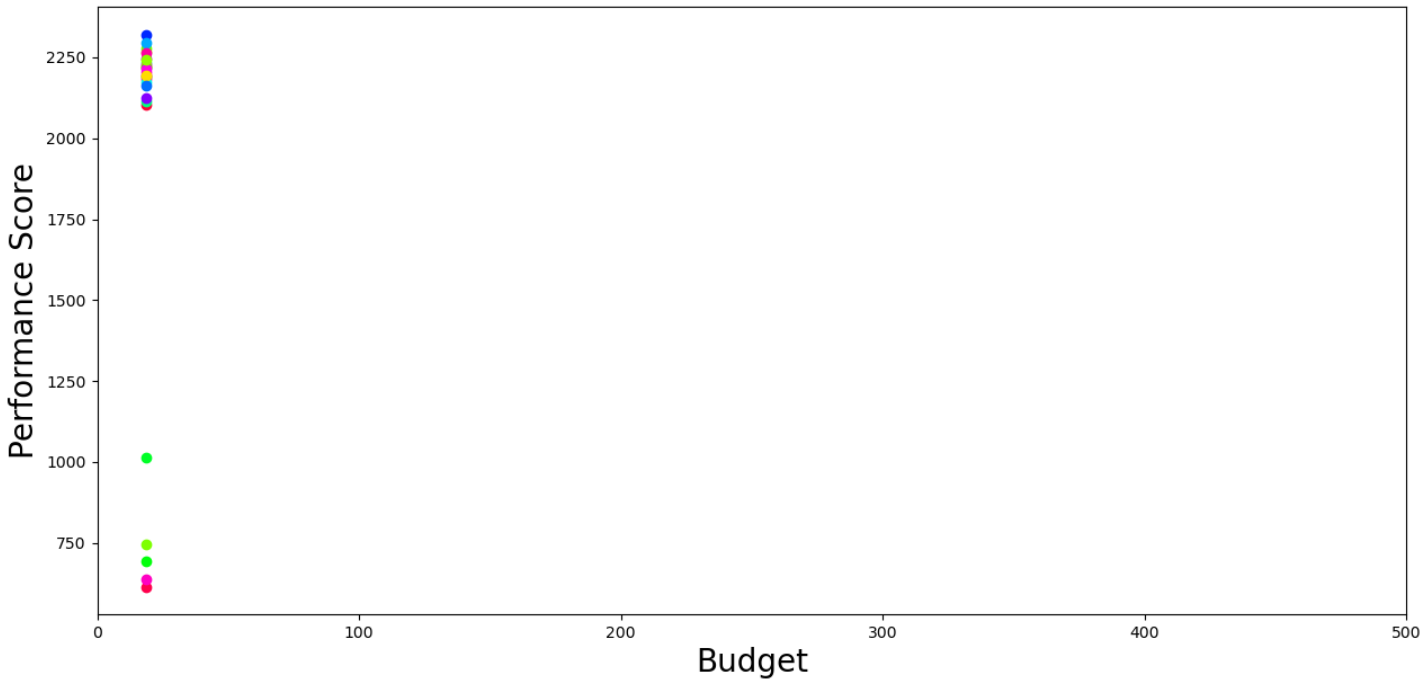
Algorithm 1 SuccessiveHalving

Input: initial budget b_0, maximum budget b_{max}, η, n different configurations $HP = \{\theta_1, \theta_2, \ldots, \theta_n\}$.

1: $b = b_0$
2: **while** $b \leq b_{max}$ **do**
3: Evaluate all configuration in HP with budget b, $L = \{f(\theta, b) : \theta \in HP\}$.
4: Pick the top $\lfloor \frac{|HP|}{\eta} \rfloor$ performing configuration. $HP = top_k(L, HP, \lfloor \frac{|HP|}{\eta} \rfloor)$.
5: Set the next round budget, $b = \eta \times b$.
6: **end while**

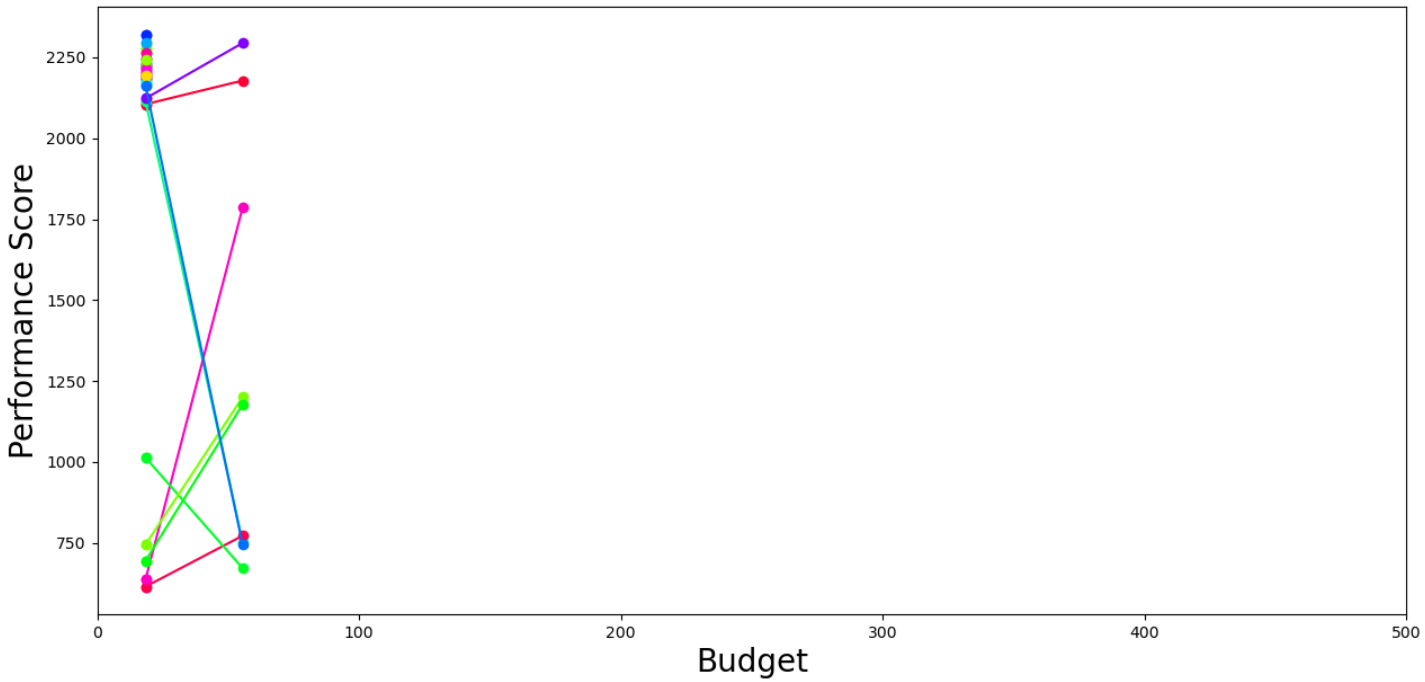
Successive Halving

Set initial budget $b_0 = 18$, maximum budget $b_{max} = 486$ and number of different configurations $n = 27$.



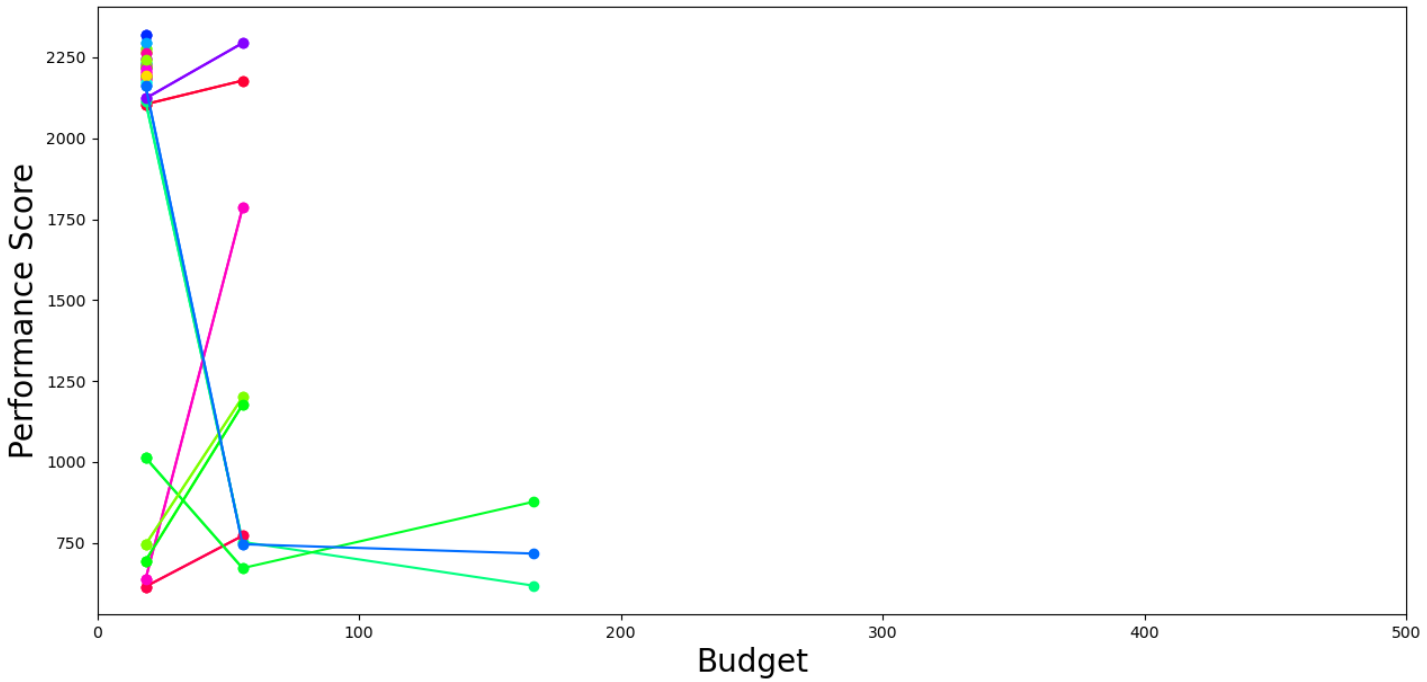
In the first iteration, 27 models are evaluated with budget of 18 each.

Successive Halving



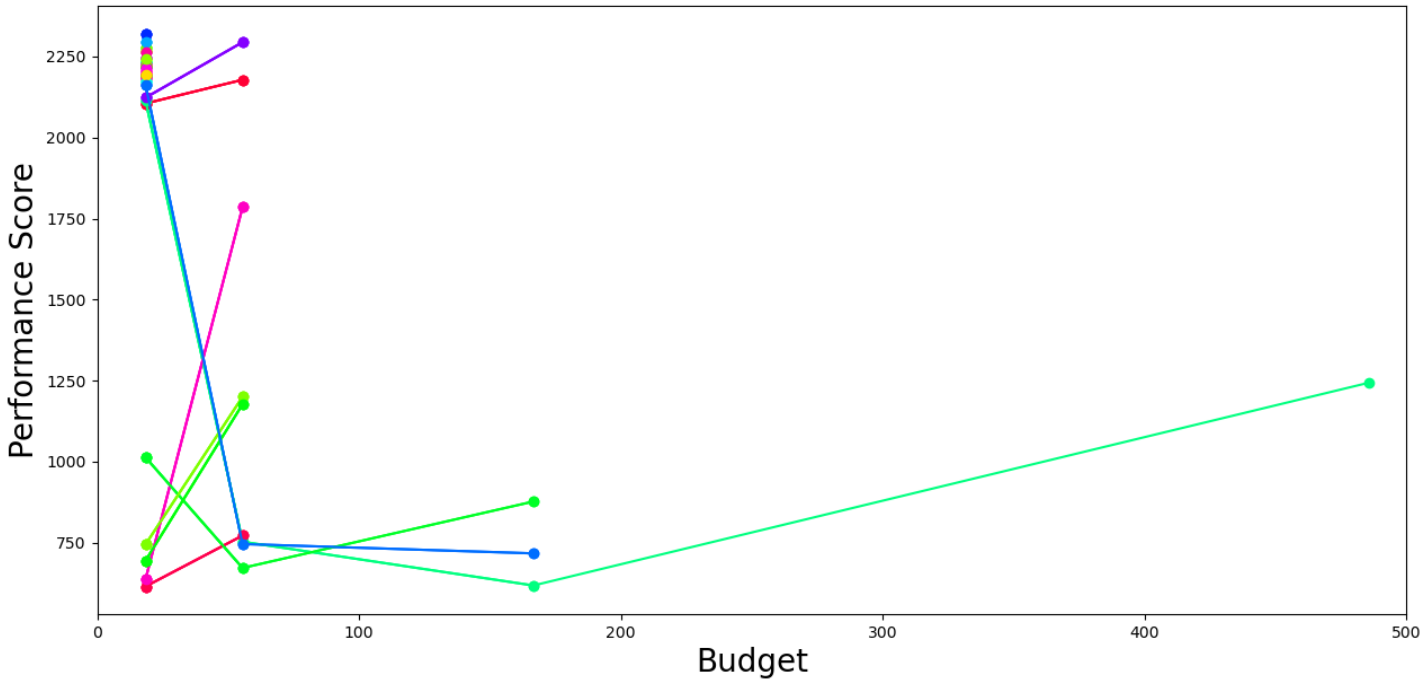
In the second iteration, the top 9 models are rerun with a larger budget of 54 each. (Note: $27 \div \eta = 9$)

Successive Halving



The top 3 models out of the previous 9 models are rerun with a larger budget of 162. (Note: $9 \div \eta = 3$)

Successive Halving



Choose the top performing out of the 3 models and run with the budget of 486. **Limitation:** There is a trade-off in terms of the **number of configurations** to initialize and the **initial budget** to be used.

HyperBand

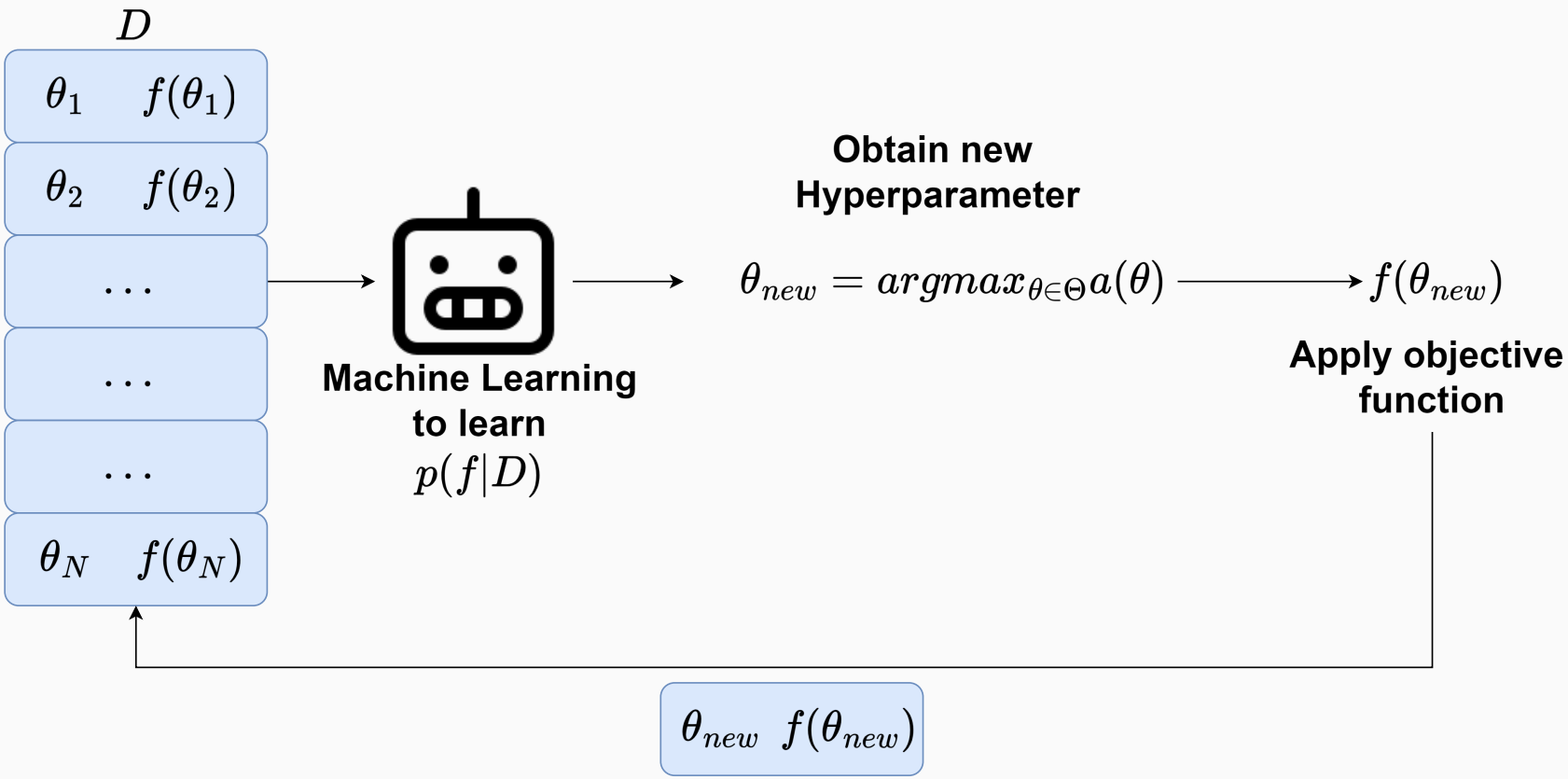
Li et al. [9] created HyperBand to solve this issue by repeatedly applying the SuccessiveHalving with **different starting number configurations and initial budget**.

Algorithm 2 HyperBand

Input: minimum and maximum budgets per configuration b_{min} and b_{max}, η

1: $s_{max} = \lfloor log_\eta \frac{b_{max}}{b_{min}} \rfloor$
2: **for** s from s_{max} to 0 **do**
3: sample $n = \lfloor \lfloor \frac{s_{max}+1}{s+1} \rfloor \times \eta^s \rfloor$ configurations $HP = \{\theta_1, \theta_2, \ldots, \theta_n\}$
4: run $SuccessiveHalving(b_0, b_{max}, \eta, HP)$ with budget $b_0 = \eta^s \times b_{max}$.
5: **end for**

Bayesian Optimization



The acquisition function $a : \Theta \to \mathbb{R}$ **based on the** $p(f|D)$

Bayesian Optimization HyberBand (BOHB)

Falkner et al. combine Bayesian Optimization and HyperBand by proposing BOHB [10].

Bayesian Optimization HyberBand (BOHB)

- HyperBand to decide the number of configurations and the budget in which these configurations are to be evaluated.
- A fraction of the random run are randomly sampled while the rest are sampled using the Bayesian Optimization.

Table of contents

Objective Functions

From Previous Work [5]**:**

- Validation Loss (denoted as **Val_Loss**)
- L_m

New Objective function:

- ge_{+ntge}

Prior Objective functions: Val_Loss

It was shown that **minimizing the categorical cross-entropy loss** is equivalent to **maximizing the generalization of the mutual information** between the leakage model and the traces (also known as perceived information).

Prior Objective functions: L_m

L_m as an objective function based on

$$LDD(k, k^*) = \sum_{i=0}^{Q} ||LM(p_i, k^*) - LM(p_i, k))||^2, k \in \mathcal{K},$$

where LM is the leakage model, p_i is the public data and k is the corresponding key. Then is L_m define as the correlation between the key guessing vector $\boldsymbol{G}$ and LDD:

$$L_m(\mathbf{LDD}, \boldsymbol{G}) = corr(argsort(LDD), \boldsymbol{G}).$$

[5] have found that L_m is the best objective function compared to key rank and validation loss.

New Objective functions: ge_{+ntge}

The main goal of the hyperparameter search:

- one should include $NTGE$ in the objective function.

New Objective functions: ge_{+ntge}

$$ge_{+ntge}(\theta) = \begin{cases} NTGE & \text{if } GE = 0, \\ GE + N_a + c & \text{otherwise} \end{cases}$$

where c is a small positive constant and N_a maximum number of attack traces.

- Since we want to show how far off the given configuration is to recover the key (i.e., $GE = 0$), we add GE to N_a.
- The constant c is further added to give an extra penalty for not recovering the key within the given number of attack traces. We set $c = 100$ throughout.

Table of contents

Experimental Setting

Fixed the iteration of BOHB to 50. Train the DNNs with categorical cross-entropy loss function. Set $\eta = 3$ and $\rho = \frac{1}{3}$ (fraction of randomly sampled configuration).

Table 1: Hyperparameter search space.

Hyperparameter	Options
MLP	
Number of Dense Layers	1 to 8 in a step of 1
Neurons per layer	$10, 20, 50, 100, 200, 300, 400, 500$
CNN	
Convolution layers	1 to 4 in step of 1
Convolution filters	4 to 16 in step of 4
Kernel size	26 to 52 in step of 2
Padding	0 to 16 in step of 2
Pooling type	Average or Max
Pooling size	2 to 10 in step of 2
Number of Dense Layers	1 to 8 in a step of 1
Neurons per layer	$10, 20, 50, 100, 200, 300, 400, 500$
Others	
Batch size	100 to 1000 in a step of 100
Activation function	*ReLU*, *SeLU*, *ELU* or *tanh*
Optimizer	Adam or RMSprop
Learning Rate	$1e-3, 1e-4, 5e-4, 1e^{-5}, 5e^{-5}$
Weight Initializer	Random Uniform or Glorot Uniform or He Uniform

We choose a larger hyperparameter search space than [5] and [6].

Budget Considered: Number of Epochs

- The budget could be any resource, like the **amount of time taken** or the number of epochs to train a neural network.

*Therefore, we consider **the number of epochs as the budget** as the parameters for BOHB.*

Based on [7], the smallest epochs to recover key is 8. We fixed our minimum budget $b_{min} = 10$. Now, we explore the impact of b_{max}

Elapse Time

Table 2: Total time taken to run BOHB.

Max Budget b_{max}	50	100	200	500
ASCADf	$\approx 3hrs$	$\approx 7.5hrs$	$\approx 12hrs$	$\approx 1day13hrs$
ASCADr	$\approx 4hrs$	$\approx 10hrs$	$\approx 14.5hrs$	$\approx 1day21hrs$
AES_HD	$\approx 5hrs$	$\approx 12hrs$	$\approx 17hrs$	$\approx 2day6hrs$
CTF2018	$\approx 4hrs$	$\approx 10hrs$	$\approx 14hrs$	$\approx 1day21hrs$

- Larger the max budget b_{max}, the longer the time required, as more hyperparameters/configurations are sampled for evaluation.

CTF2018

Table 3: $NTGE_{best}$ on the CTF2018 for HW leakage model. The best $NTGE_{best}$ among the MLP and CNN setting are marked in blue and red respectively.

Max Budget b_{max}	50	100	200	500
MLP: ge_{+ntge}	180	450	936	200
MLP: Val_loss	713	742	269	288
MLP: L_m	1219	893	1237	773
CNN: ge_{+ntge}	141	122	135	82
CNN: Val_loss	101	149	185	89
CNN: L_m	115	147	140	91

CTF2018

Table 4: $NTGE_{best}$ on the CTF2018 for ID leakage model. The best $NTGE_{best}$ among the MLP and CNN setting are marked in blue and red respectively.

Max Budget b_{max}	50	100	200	500
MLP: ge_{+ntge}	$GE = 5$	2691	2975	2983
MLP: Val_loss	2991	$GE = 1$	2987	2955
MLP: L_m	$GE = 2$	2864	2999	2523
CNN: ge_{+ntge}	2992	$GE = 2$	2999	2927
CNN: Val_loss	2912	$GE = 1$	2987	2907
CNN: L_m	$GE = 1$	2989	$GE = 1$	2958

Which Objective functions to choose?

- ge_{+ntge} obtain the best $NTGE_{best}$ in 8 out of 14 scenarios. In comparison, L_m and Val_loss both attain best $NTGE_{best}$ in 3 different scenarios. This shows that ge_{+ntge} can be considered as a **better objective function** for BOHB.
- Type of objective function could be **dataset dependent**.
- When resources/budgets are scarce, we suggest that ge_{+ntge} be the preliminary objective function to be used with BOHB.

Experimental Results: Compare to Prior Works

	Dataset	Epochs	No. of parameters	$NTGE_{best}$
[3]	ASCADf (ID)	50	16,960	191
	AES_HD	20	3,282	1,050
	ASCADf_desync50 (ID)	50	87,279	244
[7]	ASCADf (ID)	8	15,107	130
	ASCADr (ID)	8	317,408	120
	AES_HD	33	102,757	170
[5]	ASCADf (HW)	10	1,388,457	447
	ASCADf (ID)	10	1,544,776	120
	ASCADr (HW)	10	1,314,009	496
	ASCADr (ID)	50	1,539,320	1,568
	CTF2018 (HW)	50	2,418,085	618
[6]	ASCADf (HW)	50	8,480	1,246
	ASCADf (ID)	50	79,439	202
	ASCADr (HW)	50	15,241	911
	ASCADr (ID)	50	70,492	490
	CTF2018 (HW)	50	33,788	122
	ASCADf_desync50 (HW)	50	516,361	1,592
	ASCADf_desync50 (ID)	50	41,321	443
Ours	ASCADf (HW)	56	845,109	849
	ASCADf (ID)	200	10,596	201
	ASCADr (HW)	34	659,409	879
	ASCADr (ID)	17	1,465,056	1,568
	AES_HD	34	1,725,856	1,030
	CTF2018 (HW)	500	3,645	82
	CTF2018 (ID)	18	25,596	2,523
	ASCADf_desync50 (HW)	500	10,401	2,4698
	ASCADf_desync50 (ID)	500	91,976	1,311

Table of contents

Future Works

- Look into the capability of different multifidelity optimization in the SCA domain, such as DEHB, in comparison to BOHB.
- Study objective functions that consider the size of the network.

References

[1] Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking Cryptographic Implementations Using Deep Learning Techniques. pp. 3–26 (12 2016). https://doi.org/10.1007/978- 3-319-49445-6 1

[2] Cagli, E., Dumas, C., Prouff, E.: Convolutional Neural Networks with Data Aug- mentation Against Jitter-Based Countermeasures. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2017. pp. 45–68. Springer International Publishing, Cham (2017)

[3] Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for Efficient CNN Architectures in Profiling Attacks. IACR Transac- tions on Cryptographic Hardware and Embedded Systems 2020(1), 1–36 (Nov 2019). https://doi.org/10.13154/tches.v2020.i1.1-36, https://tches.iacr.org/index.php/TCHES/article/view/8391

[4] Wouters, L., Arribas, V., Gierlichs, B., Preneel, B.: Revisiting a Method- ology for Efficient CNN Architectures in Profiling Attacks. IACR Trans- actions on Cryptographic Hardware and Embedded Systems 2020(3), 147–168 (Jun 2020). https://doi.org/10.13154/tches.v2020.i3.147-168, https://tches.iacr.org/index.php/TCHES/article/view/8586

[5] Wu, L., Perin, G., Picek, S.: I Choose You: Automated Hyperparameter Tuning for Deep Learning-based Side-channel Analysis. IEEE Transactions on Emerging Top- ics in Computing pp. 1–12 (2022). https://doi.org/10.1109/TETC.2022.3218372

[6] Rijsdijk, J., Wu, L., Perin, G., Picek, S.: Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems 2021(3), 677–707 (Jul 2021). https://doi.org/10.46586/tches.v2021.i3.677-707, https://tches.iacr.org/index.php/TCHES/article/view/8989

[7] Acharya, R.Y., Ganji, F., Forte, D.: Information theory-based evo- lution of neural networks for side-channel analysis. IACR Transac- tions on Cryptographic Hardware and Embedded Systems 2023(1), 401–437 (Nov 2022). https://doi.org/10.46586/tches.v2023.i1.401-437, https://tches.iacr.org/index.php/TCHES/article/view/9957

[8] Jamieson, K.G., Talwalkar, A.: Non-stochastic best arm identifica- tion and hyperparameter optimization. CoRR abs/1502.07943 (2015), http://arxiv.org/abs/1502.07943

[9] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. Journal of Machine Learning Research 18(185), 1–52 (2018), http://jmlr.org/papers/v18/16-558.html

[10] Falkner, S., Klein, A., Hutter, F.: BOHB: robust and efficient hyperparameter optimization at scale. CoRR abs/1807.01774 (2018), http://arxiv.org/abs/1807.01774

Thank You!