POLITECNICO DI TORINO
Repository ISTITUZIONALE

Deep Learning-Based Power Side-Channel Evaluation from RTL Switching-Activity Proxies

Original
Deep Learning-Based Power Side-Channel Evaluation from RTL Switching-Activity Proxies / Farnaghinejad, Behnam; Ruospo, Annachiara; Savino, Alessandro; Di Carlo, Stefano; Sanchez, Ernesto. - ELETTRONICO. - (2026), pp. 1-6. (27th IEEE Latin American Test Symposium (LATS 2026) Florianópolis (BRA) 17-20 March 2026) [10.1109/lats70329.2026.11480302].

Availability:
This version is available at: 11583/3010128 since: 2026-04-21T07:12:02Z

Publisher:
IEEE

Published
DOI:10.1109/lats70329.2026.11480302

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

25 May 2026

Deep Learning-Based Power Side-Channel Evaluation from RTL Switching-Activity Proxies

Behnam Farnaghinejad, Annachiara Ruospo, Alessandro Savino, Stefano Di Carlo and Ernesto Sanchez
Department of Control and Computer Engineering, Politecnico di Torino, Torino, Italy
Email: {behnam.farnaghinejad, annachiara.ruospo, alessandro.savino, stefano.dicarlo, ernesto.sanchez}@polito.it

***Abstract*—Deep learning has become a central tool in power side-channel analysis, supporting not only key recovery from measured traces but also the detection of key-dependent patterns, templates, and leakage structure. Despite this progress, the use of deep learning for leakage assessment at the Register-Transfer Level (RTL) remains largely unexamined. RTL-level switching activity provides the earliest point in the design flow where power side-channel leakage can be assessed, offering fast analysis while avoiding the complexity and cost associated with gate-level evaluation. This paper presents the first study that applies both unsupervised and supervised deep learning to power side-channel evaluation at RTL-level. An unsupervised Convolutional Long Short-Term Memory (ConvLSTM) autoencoder is used to learn temporal representations without key labels, and a supervised Convolutional Neural Network (CNN) is included to provide a profiling-style reference. Both models are evaluated using 40,000 RTL switching-activity proxies generated with VeriSide for AES-128 executions on a CVA6 core with an AES-64 accelerator. The unsupervised model achieves precise key distinguishability and consistent leakage localization, approaching the supervised baseline with moderately higher trace requirements. The results show that deep learning can expose exploitable leakage directly at the RTL stage, motivating the integration of mitigation strategies early in the digital design process.**

***Index Terms*—Side-channel analysis, deep learning, pre-silicon security, RTL simulation, AES, RISC-V, VeriSide.**

I. Introduction

Power side-channel attacks exploit data-dependent variations in power consumption to recover secret information from hardware implementations. Deep learning has become a central tool in this domain, extracting dependency patterns from measured traces and reducing the reliance on handcrafted leakage models or manual Point-of-Interest (PoI) selection. Prior work has shown that neural networks can automatically learn discriminative features, cope with desynchronization, and outperform classical profiling attacks on several AES platforms [1]–[3].

While both supervised and unsupervised deep learning have been explored in the context of physical power measurements, their application to Register-Transfer Level (RTL) side-channel evaluation has received limited attention. RTL simulation provides a rapid opportunity for early-stage leakage analysis, as switching activity can serve as a proxy for power consumption without the computational cost associated with gate-level simulation. Recent pre-silicon studies have demonstrated that RTL-generated switching activity contains meaningful key-dependent structure and can approximate post-silicon leakage trends [4], [5], [7].

Despite this progress, deep sequence models have not been systematically evaluated on RTL-derived switching-activity proxies, and comparisons between supervised and unsupervised methods in this setting are missing. In particular, it remains unclear how well unsupervised representation learning can expose key-dependent behavior from simulation traces and how its performance relates to supervised profiling models trained with explicit leakage labels. This work focuses strictly on RTL-level leakage assessment, which is the earliest stage where full design visibility is available. Although prior studies suggest that RTL-derived switching activity correlates with post-silicon trends, we do not claim direct generalization to gate-level or measured traces. Future work includes validating our unsupervised latent representations against real power measurements to quantify cross-level consistency.

This work performs such an evaluation using RTL-derived power proxies generated with VeriSide [5] for AES-128 executions on a CVA6 core [6] equipped with an AES64 accelerator. An unsupervised Convolutional Long Short-Term Memory (ConvLSTM) autoencoder is used to learn temporal representations without key labels, while a supervised Convolutional Neural Network (CNN) serves as a profiling baseline targeting a standard Hamming Weight (HW) model. Both models operate on roughly 40,000 standardized traces aligned to the AES execution window.

The goal of comparing supervised and unsupervised models is not to claim equivalence, but to determine whether unsupervised learning, despite lacking labeled data, can achieve a comparable key-ranking capability. This demonstrates whether latent representations alone are sufficient to expose leakage, which is crucial for scenarios where designers do not have access to intermediate values or prefer not to rely on a specific leakage model. The results show that the unsupervised model extracts clear key-dependent structure, achieves reliable key ranking, and identifies the same leakage window as the supervised baseline, requiring only moderately more attack traces. These findings demonstrate that deep learning can reveal exploitable side-channel leakage as early as the RTL simulation stage, enabling earlier detection and mitigation in the digital design flow.

This work adopts the key-ranking view used in the leakage assessment literature, as it aligns with design-time evaluation

This work was funded by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

goals: it quantifies how strongly the traces support a hypothetical attack, rather than performing the attack itself. Section III details this methodology.

The remainder of the paper is organized as follows. Section II reviews relevant background and related work. Section III details the dataset, preprocessing, and model architectures. Section IV presents the experimental evaluation. Section VI concludes with implications for early-stage mitigation.

II. Background and Related Work

A. Power Side-Channel Analysis

Power side-channel attacks exploit data-dependent variations in the dynamic power consumption of digital circuits. These variations originate from switching activity in the underlying CMOS logic, which causes distinct transitions when manipulating different intermediate values during cryptographic computations. A central objective of side-channel analysis is to model and evaluate how closely such switching behavior correlates with sensitive internal states.

Classical leakage models approximate dynamic power using simple abstractions such as the HW of a state or the Hamming Distance (HD) between successive states. These models form the basis of widely used analysis techniques, including Correlation Power Analysis (CPA), template attacks, and statistical leakage-detection methods such as Test Vector Leakage Assessment (TVLA). While effective for many designs, these techniques assume relatively simple leakage structures and often require manual identification of PoIs. They may also underestimate leakage that is distributed across multiple cycles or arises from nonlinear interactions. These limitations have motivated the exploration of machine learning and deep learning approaches that can learn more expressive leakage representations directly from traces.

B. Deep Learning for Side-Channel Analysis

Deep learning has significantly reshaped the landscape of side-channel attacks across different labeling assumptions. In supervised profiling settings, neural networks are trained using intermediate-value labels derived from a known key or symbolic execution. Early work demonstrated that Multi-Layer Perceptrons (MLPs) and CNNs can outperform classical template attacks on AES power datasets [1]. Follow-up studies showed that CNNs combined with data augmentation can remain effective under substantial trace misalignment and jitter, substantially improving profiling robustness [2]. Systematic evaluations and surveys further established supervised deep learning as a strong baseline for key recovery, highlighting the influence of architecture, augmentation, and training procedures on generalization [3], [8], [9].

Beyond profiling, deep learning has also been explored in non-profiled and unsupervised scenarios where intermediate labels are unavailable. Earlier machine-learning approaches demonstrated that classifiers such as Least-Squares Support Vector Machines (LS-SVMs) can replace analytical leakage models [11]. Subsequent non-profiled deep learning attacks employed clustering or gradient sensitivity to extract key-dependent structure from unlabeled traces [12]. More recent unsupervised and self-supervised techniques, such as autoencoder-based feature learning and mutual–information–driven clustering, have shown that deep models can construct latent spaces that capture subtle key-dependent behavior without requiring explicit leakage labels [13], [14]. Collectively, these efforts demonstrate that deep learning is effective across a spectrum of assumptions, from fully supervised profiling to completely unlabeled analysis.

C. Leakage Assessment from RTL Switching Activity

Simulation-based power analysis provides an opportunity to assess leakage early in the digital design flow. Instead of relying on physical measurements, designers can inspect switching activity during functional simulation and treat it as a proxy for dynamic power. Survey studies summarize evaluation methodologies spanning behavioral, register-transfer, gate, and layout levels [15]. Prior work has shown that considering switching activity into approximate power traces enables the application of classical leakage metrics for pre-silicon analysis [4].

VeriSide contributes to this direction by generating compact switching-activity proxies through the aggregation of HW or HD transitions across multiple internal signals [5]. This approach facilitates scalable leakage estimation even for large designs.

Other structural techniques, including graph-based prediction on control–data flow representations, provide complementary assessments of vulnerability without exhaustive trace simulation [16]. Studies of RISC-V cryptographic extensions further indicate that simulation-derived switching activity can approximate the leakage trends observed from FPGA measurements [7].

Although deep learning has been studied on measured traces, its application to simulation-derived switching activity, particularly with temporal neural networks in both supervised and unsupervised forms, has received limited exploration. This motivates evaluating whether deep sequence models can extract key-dependent information directly from these proxies and comparing their behavior under different training assumptions.

III. Methodology

A. Threat Model

This work does not simulate an adversarial attack, but instead performs a design-time leakage assessment. The evaluator observes RTL switching activity proxies generated for random plaintexts and a fixed, unknown key. The evaluation mirrors the information an attacker would receive on real hardware, enabling early estimation of susceptibility to key-recovery attacks.

The goal is to assess whether the simulated traces exhibit key-dependent leakage and to quantify its strength. The analysis focuses on identifying (i) the presence of exploitable structure in the proxies, (ii) the number of traces needed for reliable key ranking, and (iii) the time regions contributing most to leakage. The supervised CNN represents a profiling-style evaluation

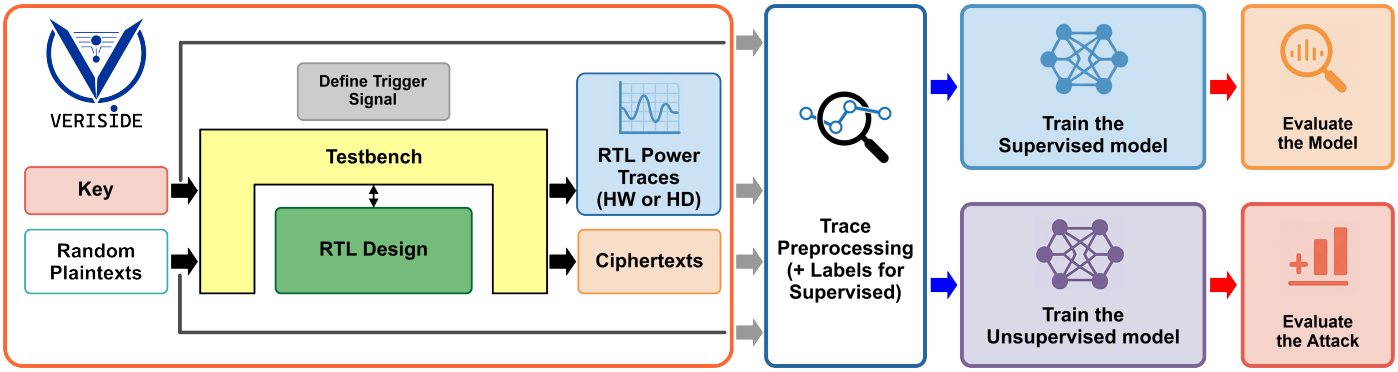

Fig. 1: End-to-end analysis workflow derived from the RTL testbench, Hamming Distance (HD) or Hamming Weight (HW) extraction by VeriSide, preprocessing, and supervised/unsupervised learning pipelines.

that relies on a predefined leakage model. In contrast, the unsupervised ConvLSTM autoencoder corresponds to a label-free setting where no intermediate values or key information are used during training. The threat model excludes physical tampering, fault injection, and countermeasures that mask the system's behavior, which are outside the scope of this study.

B. Design Under Test and Trace Generation

The evaluation is intentionally limited to one RISC-V microarchitecture (CVA6) and one cryptographic primitive (AES-128). The VeriSide framework instruments the simulation to aggregate HDs across internal signals, producing a compact scalar proxy per clock cycle. Extending the analysis to masked accelerators, lightweight ciphers, and larger SoCs is part of future work.

Each trace corresponds to one AES-128 computation. A dedicated debug CSR signal is used to align the traces so that the internal round operations appear at comparable time indices. The acquisition window spans the interval from plaintext loading through ciphertext write-back, yielding traces of 115 samples.

The final dataset contains approximately 40,000 standardized switching-activity proxies. A summary of relevant characteristics is provided in Table I. All models in this study operate on this same dataset.

TABLE I: Dataset characteristics for RTL power proxies.

Parameter	Value
Cipher	AES-128
CPU core	CVA6 (64-bit RISC-V)
Crypto extension	AES64 accelerator
Power proxy	RTL Hamming-Distance aggregation
Number of traces	40,000
Trace length	115 time samples
Key	Fixed, unknown to models
Plaintexts	i.i.d. random
Alignment	Trigger signal

C. Analysis Workflow

Figure 1 illustrates the complete analysis workflow used in this study. The process begins with the RTL design integrated into a testbench, where a trigger signal defines the acquisition window for each AES-128 execution. VeriSide aggregates switching activity across selected internal signals, producing cycle-level Hamming-weight or Hamming-distance power proxies together with the corresponding ciphertexts.

The resulting traces undergo a preprocessing stage, which includes standardization and the generation of intermediate-value labels when required by the supervised branch. The processed data is then fed into two parallel learning pipelines. In the first, a supervised convolutional neural network is trained to model a classical profiling-style scenario. In the second, a contrastive ConvLSTM autoencoder is trained without any key or intermediate labels, forming a purely unsupervised representation of the traces.

Both branches ultimately output key-ranking metrics and success-rate (SR) curve results.

D. Preprocessing and Standardization

For each time index $t \in \{1, \ldots, 115\}$, the empirical mean μ_t and standard deviation σ_t are computed over the entire dataset. Each trace x is then standardized component-wise as

$$\tilde{x}[t] = \frac{x[t] - \mu_t}{\sigma_t + \varepsilon}, \tag{1}$$

where ε is a small constant to avoid division by zero. All models use these standardized traces $\tilde{x}$ as input.

Traces are partitioned into disjoint subsets for training, validation, and evaluation. Training subsets are used to optimize model parameters, validation subsets guide early stopping and hyperparameter selection, and evaluation subsets are used to generate success-rate curves and confidence intervals.

E. Unsupervised Model Exploration

Before selecting the ConvLSTM autoencoder, several unsupervised architectures were evaluated, including plain and weighted MLP-based autoencoders, One-Dimensional Convolution (Conv1D) autoencoders, and β-Variational Autoencoder (VAE) variants ($\beta \in \{0.1, 0.5, 1.0, 2.0\}$). Across these models, the reconstruction loss saturated rapidly and the latent representations largely reproduced the input traces, resulting in limited key-ranking performance (typically SR $\approx$ 0.68–0.75) and consistent difficulty on weakly leaking bytes.

The β-VAE models further degraded performance as Kullback–Leibler Divergence (KL) regularization suppressed the subtle RTL-level leakage present in the traces, with the success rate dropping to 0.50–0.62 for $\beta \geq 0.5$.

Conv1D autoencoders improved locality modeling but remained limited by the reconstruction-only objective and did not yield substantial gains over the MLP-based variants.

In contrast, the ConvLSTM autoencoder combined temporal modeling with a contrastive loss, which prevented latent-space collapse and reliably amplified key-dependent structure, yielding markedly higher and more stable performance (up to $\mathrm{SR} = 0.94$) across seeds and configurations.

This motivated its selection as the final unsupervised model.

F. ConvLSTM Autoencoder

The unsupervised model is a one–dimensional ConvLSTM autoencoder optimized with a self-supervised objective combining reconstruction and contrastive learning. Its architecture, shown in Figure 2, consists of an encoder composed of stacked Conv1D layers followed by an LSTM that integrates temporal dependencies. The final LSTM state is projected to a latent vector $z \in \mathbb{R}^d$ through a linear layer. The decoder mirrors this structure: an LSTM reconstructs a temporal sequence from z, followed by Conv1D layers that generate the reconstructed trace $\hat{x}$.

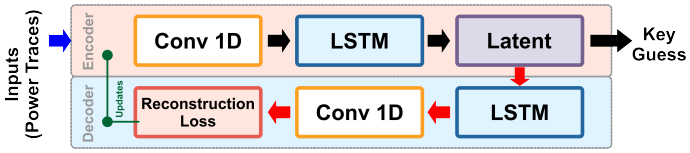

Fig. 2: Contrastive ConvLSTM autoencoder

Training minimizes a combination of reconstruction error and contrastive loss. The reconstruction cost is the mean squared error

$$\mathcal{L}_{\mathrm{rec}} = \frac{1}{B} \sum_{i=1}^{B} \|x_i - \hat{x}_i\|_2^2, \tag{2}$$

computed over a batch of size B. For contrastive learning, a positive pair for each trace is generated by Gaussian augmentation:

$$x_i^+ = x_i + \eta_i, \qquad \eta_i \sim \mathcal{N}(0, \sigma^2 I), \tag{3}$$

where σ controls augmentation strength. All other traces in the batch serve as negative examples.

Latent similarity is measured via cosine similarity,

$$\mathrm{sim}(u, v) = \frac{u^\top v}{\|u\|\|v\|}. \tag{4}$$

The contrastive objective is the InfoNCE (Information Noise-Contrastive Estimation) loss, which for sample i is

$$\mathcal{L}_{\mathrm{InfoNCE},i} = -\log \frac{\exp(\mathrm{sim}(z_i, z_i^+)/\tau)}{\sum_k \exp(\mathrm{sim}(z_i, z_k)/\tau)}, \tag{5}$$

with temperature parameter $\tau > 0$. The batch-level contrastive loss is

$$\mathcal{L}_{\mathrm{InfoNCE}} = \frac{1}{B} \sum_{i=1}^{B} \mathcal{L}_{\mathrm{InfoNCE},i}. \tag{6}$$

The total optimization objective is

$$\mathcal{L} = \lambda_{\mathrm{rec}} \mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{con}} \mathcal{L}_{\mathrm{InfoNCE}}, \tag{7}$$

where λ_{rec} and λ_{con} balance reconstruction and contrastive learning. Optimization uses the Adam algorithm with early stopping. Latent dimension d, contrastive weight λ_{con}, and augmentation noise level σ are swept across multiple configurations and random seeds.

Although $\mathcal{L}_{\mathrm{rec}}$ enforces accurate reconstruction, relying on reconstruction alone often leads to *latent collapse*, where the encoder maps different traces to nearly identical latent vectors ($z_i \approx z_j$). This occurs because minimizing $\|x_i - \hat{x}_i\|^2$ does not require the latent space to preserve discriminative factors. The contrastive term counteracts this effect by enforcing a margin:

$$\mathrm{sim}(z_i, z_i^+) \gg \mathrm{sim}(z_i, z_j), \qquad j \neq i, \tag{8}$$

which prevents collapse and forces the encoder to retain the small but systematic variations across switching-activity sequences. These variations include precisely the key-dependent differences the analysis aims to detect. Thus, the combined loss acts as an effective regularizer, amplifying trace-dependent structure and yielding a latent space with significantly improved unsupervised discriminability.

After training, the encoder is frozen and used to map each evaluation trace to a latent vector. Unsupervised key ranking is performed by fitting a multivariate Gaussian distribution in the latent space for each HW class of the hypothetical AES intermediate. Given a key hypothesis k and byte index b, the hypothetical intermediate value $Y_{b,i}(k)$ is computed from the ciphertext of trace i, its HW determines the Gaussian cluster, and the trace likelihood is

$$\log p(z_i \mid Y_{b,i}(k)). \tag{9}$$

Accumulating over N attack traces gives the key score

$$\mathrm{LL}_{\mathrm{unsup}}(k; N) = \sum_{i=1}^{N} \log p(z_i \mid Y_{b,i}(k)). \tag{10}$$

Keys are ranked according to $\mathrm{LL}_{\mathrm{unsup}}(k; N)$, and the success rate is the probability that the correct key achieves rank one.

G. Supervised CNN Baseline

A supervised CNN is trained to provide a profiling-style reference model on the same standardized switching-activity traces used in the unsupervised pipeline. The network consists of several one-dimensional convolutional layers with non-linear activations and pooling, followed by fully connected layers and a softmax output. The classifier predicts either the 256 possible values of the last-round intermediate for a selected byte or, equivalently, the HW class derived from this intermediate.

Training minimizes the cross-entropy loss between the predicted distribution and the correct intermediate-derived label for

each trace. Optimization uses the Adam algorithm with early stopping. The same training–validation split as the unsupervised model is used to ensure comparability, and evaluation traces remain strictly unseen during training.

At evaluation time, for each key hypothesis k and byte index b, the network outputs the probability

$$p_\theta\big(Y_{b,i}(k) \mid x_i\big), \tag{11}$$

where $Y_{b,i}(k)$ is the hypothetical intermediate derived from ciphertext byte $C_{b,i}$. Supervised log-likelihood scores accumulate over N attack traces:

$$\mathrm{LL}_{\mathrm{sup}}(k; N) = \sum_{i=1}^{N} \log p_\theta\big(Y_{b,i}(k) \mid x_i\big). \tag{12}$$

Keys are ranked according to $\mathrm{LL}_{\mathrm{sup}}(k; N)$, and the success rate $\mathrm{SR}_{\mathrm{sup}}(N)$ denotes the fraction of evaluations in which the correct key is ranked first.

H. Evaluation Metrics and Protocol

Both the supervised and unsupervised pipelines are evaluated using key-ranking metrics derived from log-likelihood scores. The primary metric is the success rate $\mathrm{SR}(N)$, defined as the probability that the correct key obtains rank one when N attack traces are used. Success-rate curves are computed by repeatedly sampling trace subsets and averaging across multiple random seeds, and 95% confidence intervals quantify variability.

A second metric is the log-likelihood gap,

$$\Delta(N) = \mathrm{LL}(k^\star; N) - \max_{k \neq k^\star} \mathrm{LL}(k; N), \tag{13}$$

which measures statistical separation between the correct key $k^\star$ and the closest competing hypothesis. Increasing positive gaps indicate stronger discrimination.

For the unsupervised model, robustness is assessed by training several configurations spanning different latent dimensionalities, contrastive weights, and augmentation noise levels. Each configuration is run across multiple random seeds, and stability is evaluated by analyzing the mean success rate, its variance, and the distribution of key ranks.

I. Hyperparameter Study for the ConvLSTM Autoencoder

The contrastive ConvLSTM autoencoder is evaluated across a controlled grid of hyperparameters to explore architectural and training robustness. The grid spans three latent dimensions $d \in \{8, 16, 32\}$, two contrastive weights $\lambda_{\mathrm{con}} \in \{0.1, 0.3\}$, and two augmentation noise levels $\sigma \in \{0.00, 0.02\}$. For each combination, several independent training runs are executed with different random seeds.

For every trained model, performance is summarized using the full-key success rate, the distribution of key ranks, and the behavior of partial-trace success rates at fixed values of N. These statistics provide insight into the stability of the learned latent representation and its sensitivity to hyperparameter choices. The resulting trends enable identification of regimes that yield consistently strong discrimination, informing the configuration used in the main comparison of Section IV.

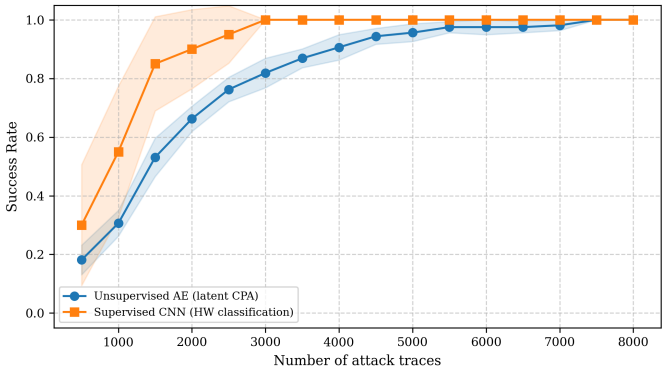

Fig. 3: Success rate versus number of attack traces for the supervised CNN and the unsupervised ConvLSTM autoencoder.

IV. Experimental Results

Figure 3 compares the key–recovery success rate as a function of the number of attack traces for the supervised CNN and the unsupervised ConvLSTM autoencoder. Each curve is averaged over several random seeds, with shaded bands indicating the corresponding confidence intervals. The supervised CNN reaches 100% success rate with roughly $2,500$–$3,000$ traces, while the unsupervised autoencoder follows a very similar trajectory but requires slightly more traces to attain the same success level. Beyond about $4,500$–$5,000$ traces, both methods saturate near $\mathrm{SR} = 1$, confirming that RTL switching-activity proxies contain enough key-dependent structure for reliable key ranking even in the absence of labels.

The influence of the primary autoencoder hyperparameters is summarized in Figure 4. The figure reports the mean full-key success rate for each combination of latent dimension d, contrastive weight λ_{con}, and augmentation noise standard deviation σ. All tested settings yield high success rates, with values typically above 0.9. Increasing the latent dimension from $d = 8$ to $d = 32$ provides a small but consistent gain, while changes in λ_{con} and the presence or absence of light Gaussian noise have only a minor effect. This behavior indicates that the proposed unsupervised approach is robust to reasonable hyperparameter choices and does not rely on delicate tuning.

To provide intuition on the physical relevance of RTL-derived proxies, Figure 5 reports their correlation with real FPGA power consumption; this analysis is not intended to validate the proposed deep learning methodology against measured traces.

In the figure, the x-axis corresponds to RTL simulation cycles, while the y-axis represents FPGA power bins. Each FPGA bin aggregates multiple ADC samples acquired at a higher sampling frequency and summarizes them using an RMS-based power estimate over a fixed temporal window. This binning allows the high-resolution measured trace to be approximately aligned with the lower-rate RTL simulation timeline.

The presence of a visible diagonal structure indicates that specific simulation cycles consistently correlate with particular regions in the measured power trace. This suggests that RTL switching-activity proxies capture coarse-grained timing infor-

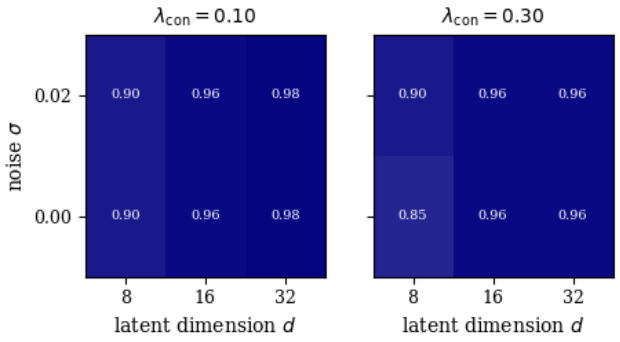

Fig. 4: Effect of ConvLSTM autoencoder hyperparameters on mean full-key success rate.

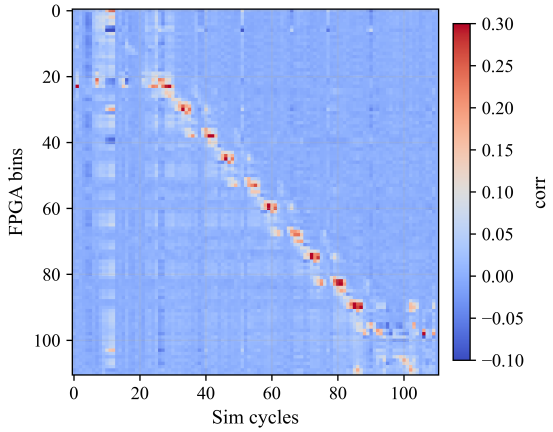

Fig. 5: Correlation map between RTL simulation cycles and measured FPGA power consumption.

mation that is preserved in real hardware power consumption. A quantitative comparison between deep learning models trained on RTL proxies and models trained on measured power traces is left as future work.

V. Discussion

The results show that RTL switching activity contains a clear key-dependent structure that both supervised and unsupervised models can exploit. The supervised CNN converges quickly. At the same time, the contrastive ConvLSTM autoencoder approaches comparable accuracy with only a slight increase in required traces, confirming that useful leakage patterns can be learned without labels. The hyperparameter study indicates stable behavior across latent dimensions, contrastive weights, and noise levels, suggesting that extensive tuning is unnecessary.

VI. Conclusion and Future Work

This work demonstrates that deep sequence models applied to RTL switching activity can effectively assess design-time leakage. An unsupervised ConvLSTM autoencoder achieves reliable key ranking with performance comparable to that of a supervised CNN, despite not utilizing key labels. These findings suggest that unsupervised learning offers a practical tool for early-stage evaluation before synthesis or layout. Future work includes a systematic comparison against gate-level and physical measurements, extending to masked or hardened implementations, and incorporating structural information, such as control–dataflow graphs, to improve interpretability and reduce sample complexity.

References

[1] H. Maghrebi, "Deep Learning based Side Channel Attacks in Practice," IACR Cryptol. ePrint Arch., 2019, Accessed: Nov. 17, 2025. [Online]. Available: https://www.semanticscholar.org/paper/Deep-Learning-based-Side-Channel-Attacks-in-Maghrebi/c0897cfbb2d0abc93a751e10cd2fc16f1cf7769c

[2] E. Cagli, C. Dumas, and E. Prouff, "Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures," in Cryptographic Hardware and Embedded Systems – CHES 2017, W. Fischer and N. Homma, Eds., Cham: Springer International Publishing, 2017, pp. 45–68. doi: 10.1007/978-3-319-66787-4_3.

[3] "SoK: Deep Learning-based Physical Side-channel Analysis." Accessed: Nov. 17, 2025. [Online]. Available: https://eprint.iacr.org/2021/1092

[4] N. Pundir, J. Park, F. Farahmandi, and M. Tehranipoor, "Power Side-Channel Leakage Assessment Framework at Register-Transfer Level," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 30, no. 9, pp. 1207–1218, Sept. 2022, doi: 10.1109/TVLSI.2022.3175067.

[5] B. Farnaghinejad, A. Porsia, A. Ruospo, A. Savino, S. Di Carlo, and E. Sanchez, "Late Contribution: VeriSide: A Modified Verilator for Leakage Assessment at the RTL Level," in 2025 IEEE 26th Latin American Test Symposium (LATS), Mar. 2025, pp. 1–2. doi: 10.1109/LATS65346.2025.10963943. [Online]. Available: https://github.com/smilies-polito/VeriSide

[6] F. Zaruba and L. Benini, The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology. (Jul. 2019). doi: 10.1109/TVLSI.2019.2926114.

[7] B. Farnaghinejad et al., "Power Side-Channel Vulnerabilities of a RISC-V Cryptography Accelerator Integrated into CVA6 via Core-V eXtension Interface (CV-X-IF)," in 2025 IEEE International Test Conference (ITC), Sept. 2025, pp. 233–242. doi: 10.1109/ITC58126.2025.00030.

[8] "Shift-Invariance Robustness of Convolutional Neural Networks in Side-Channel Analysis." Accessed: Nov. 17, 2025. [Online]. Available: https://www.mdpi.com/2227-7390/12/20/3279

[9] H. Li and G. Perin, "A Systematic Study of Data Augmentation for Protected AES Implementations," 2023, 2023/1179. Accessed: Nov. 17, 2025. [Online]. Available: https://eprint.iacr.org/2023/1179

[10] R. Capoferri, A. Barenghi, L. Breveglieri, N. Izzo, and G. Pelosi, "A Comparison of Deep Learning Approaches for Power-Based Side-Channel Attacks," in Secure IT Systems, L. Horn Iwaya, L. Kamm, L. Martucci, and T. Pulls, Eds., Cham: Springer Nature Switzerland, 2025, pp. 101–120. doi: 10.1007/978-3-031-79007-2_6.

[11] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: a first study," J Cryptogr Eng, vol. 1, no. 4, pp. 293–302, Dec. 2011, doi: 10.1007/s13389-011-0023-x.

[12] B. Timon, "Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis," IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 107–131, Feb. 2019, doi: 10.13154/tches.v2019.i2.107-131.

[13] K. Ramezanpour, P. Ampadu, and W. Diehl, "SCAUL: Power Side-Channel Analysis With Unsupervised Learning," IEEE Transactions on Computers, vol. 69, no. 11, pp. 1626–1638, Nov. 2020, doi: 10.1109/TC.2020.3013196.

[14] N. Yang et al., "Unsupervised side-channel power analysis based on invariant information clustering," Journal of Electronic Science and Technology, vol. 23, no. 4, p. 100333, Dec. 2025, doi: 10.1016/j.jnlest.2025.100333.

[15] "PreSCAN: A Comprehensive Review of Pre-Silicon Physical Side-Channel Vulnerability Assessment Methodologies." Accessed: Nov. 17, 2025. [Online]. Available: https://www.mdpi.com/2674-0729/3/4/16

[16] A. Srivastava et al., "SCAR: Power Side-Channel Analysis at RTL Level," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 32, no. 6, pp. 1110–1123, June 2024, doi: 10.1109/TVLSI.2024.3390601.