

Focus Session: Exploring Semantic Leakage in Edge FPGA Implementations of Neural Networks

Zhuoran Liu

Digital Security Group
Radboud University

Nijmegen, The Netherlands
z.liu@cs.ru.nl

Konstantina Miteloudi

Digital Security Group
Radboud University

Nijmegen, The Netherlands
konstantina.miteloudi@ru.nl

Durba Chatterjee

Digital Security Group
Radboud University

Nijmegen, The Netherlands
durba.chatterjee@ru.nl

Lejla Batina

Digital Security Group
Radboud University

Nijmegen, The Netherlands
lejla@cs.ru.nl

Abstract—Edge neural network implementations can be substantially accelerated on FPGAs. Open-source tools like FINN enable real-world deployment of applications in various domains. However, the privacy and security of FPGA-based edge neural network implementations have often been overlooked. Semantic leakage, a new type of side-channel vulnerability, has been identified in both software and hardware neural network implementations.

In this paper, we provide an initial analysis of FPGA implementations of convolutional neural networks (CNNs) generated using the open-source FINN framework. Our work follows the recent semantic-leakage threat model, in which the adversary aims to differentiate between categories of input data based on side-channel leakage. We mount a side-channel attack on CNNs compiled with FINN for AMD ZCU104 FPGA and show that FINN-generated designs exhibit such leakage. To further explore how leakage varies, we tune various implementation aspects, including storage elements, arithmetic operations in computation elements, and folding. Our experiments demonstrate that implementing the arithmetic operations and storage elements using look-up tables (LUTs) may be less vulnerable to semantic leakage than using the specific-purpose FPGA blocks. More importantly, we show that more folding transformations enhance resistance against semantic leakage.

I. INTRODUCTION

Field-Programmable Gate Arrays (FPGAs) play an important role in neural network inference, balancing the throughput, power efficiency, and flexibility. Edge FPGA SoCs greatly advance machine learning applications, providing flexible, secure, and private edge computation. Common applications range from autonomous driving [1] to efficient Large Language Model (LLM) inference [2]. Over the past several years, there has been substantial research in the privacy and security of machine learning [3], including adversarial examples [4], model stealing [5], membership inference [6] attacks, and so on. Previous research mainly focuses on the security and privacy of the learning algorithms themselves, without placing substantial emphasis on real-world deployments [3].

Edge neural networks are becoming prevalent with improved privacy without relying on centralized cloud computing infrastructure. At the same time, they allow physical proximity of adversaries to the physical devices running the applications. Different *physical side-channel attacks* [7] have been proposed against neural network implementations aiming for architecture extraction [8]–[11], weight extraction [12]–[15],

and input recovery [16], [17]. While physical side-channel attacks on cryptographic hardware aim to recover details about implementation [7], explicit data extraction is not necessary to compromise security and privacy of AI application users. For instance, identifying the input label is often sufficient to infer private information and does not require extracting the exact input data bytes from implementations.

Recently recognized physical side-channel attacks fall within our definition of *semantic leakage*. We introduce semantic leakage as the unintended disclosure of high-level semantics (e.g., data classes and data structures) of the processed data using traces, i.e., signals of physical information such as power consumption or EM emanations collected via physical side channels. For example, the class-wise side-channel differences between groups (i.e., distributions) can be exploited to compromise user privacy [18] in PyTorch [19] implementation on CPU. The primary reason the attack succeeds is that the max pooling layer is implemented in a non-constant time manner. Similar leakage was also observed in hardware implementations on FPGAs and ARM CPUs [20]. It was demonstrated with the help of deep learning-based side-channel analysis [21] that electromagnetic emissions can be used to predict input classes when the neural network is compiled with closed-source Vitis AI [22] that is based on matrix processing engine (MPE).

In this paper, we take a step further in exploring semantic leakage by inspecting the dataflow-style architecture and implementation choices based on the open-source framework FINN [23], [24]. Figure 1 provides the working diagram of our experiments. Taking a trained quantized neural network from Brevitas [25], FINN transcompiles the trained model into HDL, synthesizes the RTL, and generates a bitstream that is deployed on FPGA boards for inference. Different from ASICs and Vitis AI’s MPE [22], the FINN compiler generates tailored dataflow hardware architecture for each neural network, where design elements can be easily customized to fit available on-board resources. This allows us to adjust hardware designs and corresponding deployed implementations to better understand the influence of different hardware design choices on the semantic leakage. To summarize, the core contributions of this work are:

- We introduce and investigate semantic leakage through physical side-channel analysis. Experimentally, we

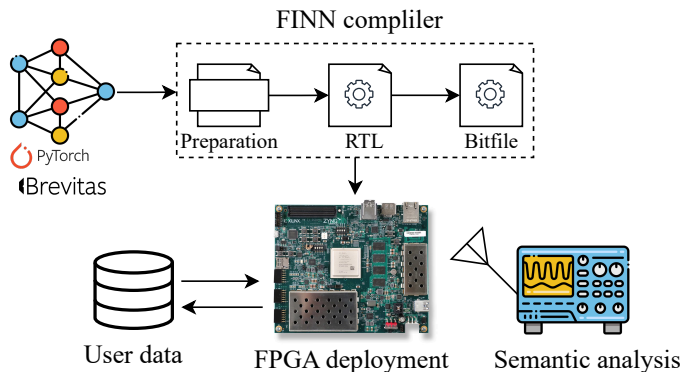


Fig. 1. An illustration of our leakage analysis methodology. Trained AI models are pre-processed and then compiled by FINN [23], [24] to synthesize RTL, based on which bitfiles are generated and deployed on AMD ZCU104 FPGA. During the FPGA inference, user input data is fed to the device, and the model output is returned. During the inference, EM traces are collected by an oscilloscope with a probe. Later, a semantic analysis can be conducted on the collected EM traces to profile user data, e.g., following [18], [20] where the prediction accuracy represents the leakage level.

demonstrate that the dataflow-style architecture generated by FINN exhibits semantic leakage. In particular, we collect EM traces and corresponding data labels and train an attribute prediction classifier to predict private attributes.

- We investigate semantic leakage under different hardware design configurations by implementing various FPGA elements. Our analysis provides insights into how different storage and computation choices influence side-channel vulnerabilities at the design level.
- We provide empirical evidence that folding transformations reduce semantic leakage albeit at the cost of increased latency.

The paper is structured around two sets of experiments that aim to detect the semantic leakage of FINN and explore how different implementation choices relate to the leakage. In Section II, we introduce the general privacy protection background and semantic leakage. In Section III, in the first set of experiments, we show the susceptibility of FINN-generated dataflow implementations to semantic leakage. In the second set of experiments, we explore different implementation choices regarding storage and computational elements and show that folding transformation has the potential to be an obfuscation against deep learning-based side-channel analysis. We also outline directions for future research. In Section IV, we wrap up our findings.

II. BACKGROUND

Linkable information refers to indirect identifiers or quasi-identifiers that, on their own, do not uniquely identify an individual but can be used to do so when combined with other data sources [26]. Unlike Personally Identifiable Information (PII), it might not be possible to identify individuals solely using linkable information, but it becomes feasible when combined with other information [26]. Well-known examples

include demographic attributes such as gender, ZIP code, and date of birth, which individually appear benign but together can uniquely identify a large fraction of the population. As data collection and data sharing practices continue to expand, linkable information is getting more attention.

Privacy attacks on machine learning models aim to infer sensitive information about the model or its training data through interactions with the model, without exploiting physical leakage, e.g., electromagnetic emissions. Model extraction attacks seek to reconstruct a target model’s architecture [11], [27] or parameters [28], [29] by observing its side-channel leakage or intermediates, typically via carefully crafted queries to a prediction API. Membership inference attacks determine whether a specific data record was included in the model’s training set, leveraging prediction confidence or loss [6], focusing on differences between training and non-training samples. Current research on privacy attacks or Personally Identifiable Information (PII) is relatively limited to AI algorithms or privacy law levels. Our work focuses on the privacy leakages from physical implementations of AI algorithms [7].

Semantic leakage has been observed on both software [18] and hardware [20] implementations of neural networks. Early work focuses on PyTorch neural network implementations running on the CPU [18], where the execution time for samples from different classes is recorded during AI model inference. On CIFAR-10 and CIFAR-100 [30], pairwise timing distribution differences between any two classes are calculated based on the t -test, where inference time distributions of different classes are mostly differentiable from each other. Further analysis concludes that max pooling is the most substantial module that introduces the timing difference, and a constant-time implementation is proposed as the countermeasure.

Later, leakage has also been observed on hardware from physical EM side channels [20], where neural network implementations on FPGA have been shown to leak the class of input data from EM emissions. The same has also been observed on Raspberry Pi 3 and 5, including edge large language model implementations [31]. However, these works study only models compiled by closed-source tools such as Vitis AI [22] based on MPE, and therefore lack leakage analysis and understanding across different implementation choices. This paper takes a step further to look at the influence of different FPGA implementation choices on semantic leakage, like storage, and computation elements, including LUTs, Block RAMs (BRAMs), and Digital Signal Processing (DSPs) slices, and especially, folding transformations.

III. SIDE-CHANNEL LEAKAGE ANALYSIS

In this section we describe the experimental setup followed by the leakage analysis results on FINN-generated implementations.

A. Experimental Setup

Attacker setup and victim model. We target a pre-trained quantized CNN model, the CNV-w2a2, provided by Brevitas [25]. The victim model is trained on the CIFAR-10 training

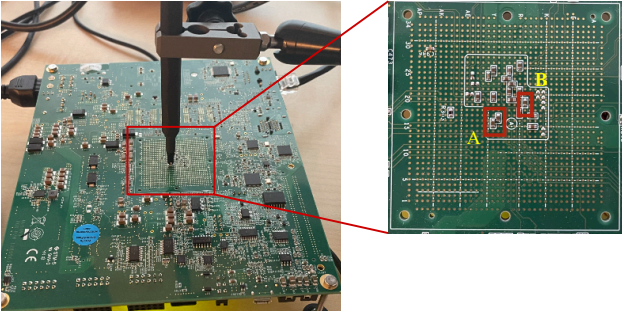


Fig. 2. EM measurement setup for AMD ZCU104 FPGA. A Langer RF-U 2.5-2 near-field probe is placed on the back of the FPGA chip in the Zynq UltraScale+ MPSoC ZCU104 evaluation board. On the right, two positions are highlighted with red boxes showing the probe positioning for LUTRAMs+LUT-MVAU (position B: top right) and others (position A: bottom left).

set with a test accuracy of 89.03%. We make different implementation choices, and after processing and compiling with FINN, the victim model is deployed on an AMD ZCU104 evaluation board [32]. Although different FPGA design choices lead to different implementations, the original trained PyTorch-Brevitas models remain the same. We use *semantic leakage accuracy* to quantify the side-channel leakage. Following the setup of profiled side-channel attacks [33], EM traces are collected when running inference on the test set of CIFAR-10 data set of 10 000 images, where 90% traces are used for training and validation, 10% for test. We train a one-dimensional four-layer CNN (1D-CNN-4) as our attack model, which takes EM traces as inputs to predict classes. The semantic leakage accuracy is the fraction of correct classifications made by the attack model over unseen test samples.

Leakage localization. We scan the top and the back of the FPGA chip, focusing on decoupling capacitors on the opposite side of the main chip. After scanning, we observe that two specific capacitors on the back of the board exhibit high amplitude emissions with limited noise, making it suitable for EM trace collection. Note that exact locations with strong EM emissions are dependent on the physical placement of the hardware design. In other words, different placements of the same neural network model on a given board can introduce substantially different EM leakages. Figure 2 shows our measurement setup and EM probe positioning (denoted by yellow rectangles marked A and B). Most measurements are done on position A. However, it is worth noting that the implementation with a higher folding factor (i.e., more clock cycles) leaks mainly at position B.

H-field Cartography. Figure 3 shows the H-field cartography of the device under test obtained from a 6×6 spatial scan across the chip back, running a LUTRAM+LUT-MVAU implementation. Here, MVAU denotes the Matrix-Vector-Activation Unit. For each measurement position, an electromagnetic trace is recorded using the same inference operation and data, and the maximum voltage peak is recorded. It provides an estimate of the local electromagnetic emission strength. This cartography highlights the spatial locality of electromagnetic leakage and

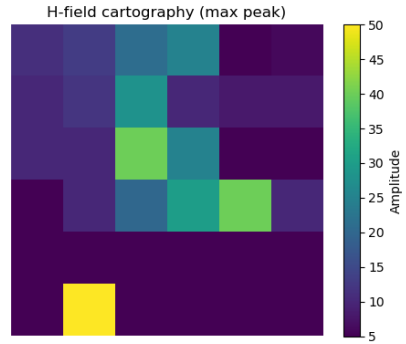


Fig. 3. Highest peaks (mV) observed at different points from a FPGA scan running LUTRAMs+DSP-MVAU implementation. The point with the highest peak (i.e., yellow point) corresponds to location A in Figure 2

serves as a basis for identifying regions of interest for subsequent analysis.

Trace collection and processing. When each input image is fed to the FPGA, a LeCroy oscilloscope is triggered to collect EM traces with a sampling rate of 2.5 GS/second. Static alignment is applied to the collected EM traces to increase temporal consistency across samples. Note that the length of the EM trace depends on the implementation’s total latency, ranging from 500K to 5 000K samples generally. Following the throughput benchmark, Table I reports the latencies of different implementations that share the same medium folding configuration of the MVAU. On one hand, DSPs improve the throughput for both LUTRAM and BRAM setups for our setup. On the other hand BRAMs reduce the throughput in both setups (LUT-MVAU and DSP-MVAU). Figure 4 depicts examples of collected EM traces for different hardware implementations. Different computation blocks are clearly visible from the trace, where simple EM analysis [9], [11] could be further conducted.

TABLE I
THROUGHPUT (IMAGES/SECOND) OF CNV-w2a2 WITH DIFFERENT IMPLEMENTATIONS ON ZCU104 WITH PL 100MHZ, EVALUATED BY BATCH SIZE 1 ON CIFAR-10 TEST SET.

	LUTRAM	BRAM
LUT-MVAU	1531.32	1377.89
DSP-MVAU	1538.06	1522.43

B. Semantic leakage in FINN-generated Implementations

We conduct a leakage analysis of FINN-generated implementations using profiled side-channel analysis, following the approach of [20]. We deploy the pre-trained quantized CNV-w2a2 CNN on ZCU104 with the provided FINN workflow [34]. In particular, we leave design options to ‘auto’ to let FINN make implementation choices automatically. Table II demonstrates that our attacking model achieves a semantic leakage accuracy of 40.07% on raw traces without alignment and 43.80% on aligned traces. Note that the accuracy of a random guess is around 10.00% since test data are uniformly distributed across 10 classes. Here, we follow static alignment [35], and we apply

this approach consistently across the remaining experiments. Even though the semantic leakage accuracy is not as high as in Vitis AI-based implementations [20], it shows that hardware implementations generated by FINN also introduce semantic leakage. Flexible design adjustments enabled by FINN allow us to test semantic leakages with different designs, which forms the focus of the following section.

TABLE II
SEMANTIC LEAKAGE ACCURACY (%) OF CNV-w2a2 ON EM TRACES COLLECTED FROM THE TEST SET OF CIFAR-10. STATIC TRACE ALIGNMENT HELPS THE SEMANTIC LEAKAGE SIDE-CHANNEL ATTACK.

	Raw	w/ Static alignment
CNV-w2a2	40.07	43.80

C. Exploring Different Implementation Choices

Storage and computation components form the fundamental elements in FPGA implementations of neural networks. In the following we discuss several use cases where various implementation and evaluation options result in different semantic leakage results.

1) *LUTs, BRAMs, and DSPs*: To investigate the impact of different implementation choices on semantic leakage, we implement the CNV-w2a2 CNN using both generic and board-specific FPGA resources. For storage, we utilize LUTRAMs and Block RAMs (BRAMs), while for arithmetic operations in the computation block (i.e., MVAU), we employ LUTs and Digital Signal Processors (DSPs). By combining these storage and computation elements, we generate four distinct implementations (LUT-MVAU+LUTRAM, LUT-MVAU+BRAM, DSP-MVAU+LUTRAM and DSP-MVAU+BRAM).

Table III demonstrates the semantic leakage accuracy for these four implementations. LUT-MVAU+LUTRAM shows slightly higher resistance against semantic leakage side-channel attacks with an accuracy of 34.57%. The better resistance of LUT-MVAU+LUTRAM against semantic leakage attack is expected because from a design perspective, this observation could be explained by the fact that LUTs might consume less power than DSPs and BRAMs, which in turn makes them less prominent to contribute to EM side-channel emission. This hypothesis should be evaluated further with more experiments.

TABLE III
SEMANTIC LEAKAGE ACCURACY (%) OF CNV-w2a2 ON CIFAR-10 EM TRACES WITH DIFFERENT IMPLEMENTATIONS.

	LUTRAM	BRAM
LUT-MVAU	34.57	39.60
DSP-MVAU	38.10	39.56

2) *Folding reduces semantic leakage*: FINN supports flexible design folding, enabling the computation to be split into smaller segments processed over several clock cycles, which facilitates balancing throughput, latency, and resource usage to optimize performance on the available hardware [23]. We

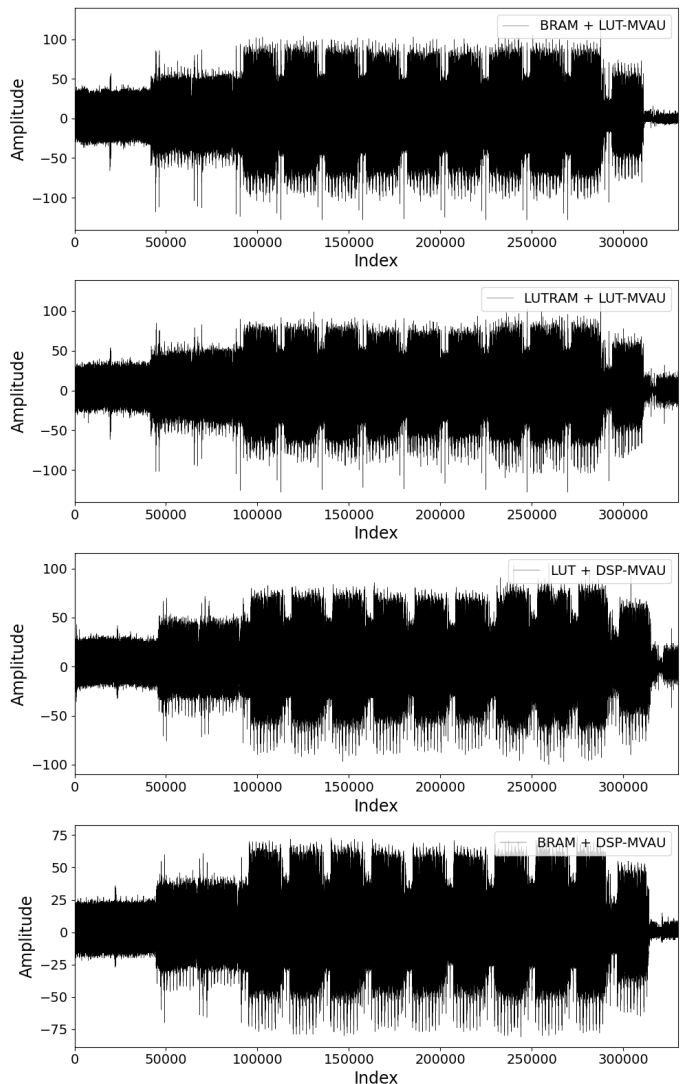


Fig. 4. EM traces of the same input collected during model inference. Four traces are from four different implementations of the same neural network model.

test the semantic leakage across three folding setups on the same FPGA board, where we control the number of processing elements (#PE), and single instruction multiple data (#SIMD). Three different folding levels are experimentally evaluated. In particular, the low-folding configuration is manually tuned to decrease the clock cycles of the bottleneck computation. Table IV demonstrates that the implementation with a higher folding level introduces less semantic leakage with lower throughput.

We observe that the folding factor influences the location of the EM leakage on the chip. Looking at Figure 2, the EM emission from a high folding factor design can only be observed at position B, and low and medium folding factor designs at position A. This suggests that the folding significantly influences the spatial distribution of EM emissions. Higher folding factors involve more clock cycles and shifts the leakage to position B. In contrast, low and medium folding factors

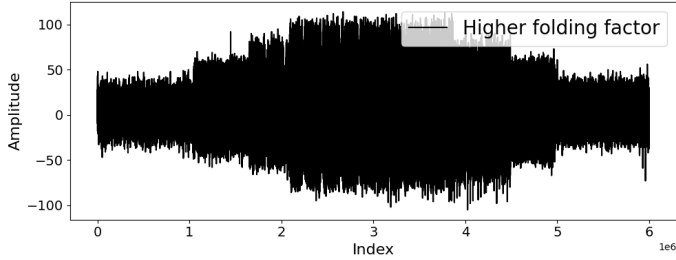


Fig. 5. EM trace with a higher folding factor collected during model inference.

concentrate the emissions at position A.

Figure 5 shows EM traces collected from designs with high folding factors. These traces reflect the data difference, with the signal strength from a higher folding design being weaker and less recognizable. We can see that the patterns of these traces are different from the ones in Figure 4. In addition, we use a machine learning model to differentiate four types of implementations. We collect 1000 sample traces from each implementation, with random inputs, to train and validate a CNN model. On the test set, the CNN model achieves 100% accuracy in classifying different FPGA implementations. The performance is expected, and such methods could be used to assist simple EM analysis. Different implementations can be properly classified by analyzing the EM traces.

Notably, EM traces with higher folding factors are less visually prominent, and they have more noise, indicating that increased folding can obscure clear electromagnetic signatures. We believe that understanding this behavior is crucial for optimizing hardware designs to mitigate potential side-channel vulnerabilities.

TABLE IV
SEMANTIC LEAKAGE ACCURACY(%) WITH LOW, MEDIUM, AND HIGH FOLDING FACTORS FOR CNV-W2A2 ON CIFAR-10.

Folding level	High	Medium	Low
semantic leakage	26.60	39.20	50.20
Throughput (images/second)	241.29	1531.32	3144.16

3) *Cross-design evaluation*: We also provide a cross-design test to explore the generalization of 1D-CNN-4 to unseen implementations of the same neural network. Figure 6 shows the confusion matrix where attacking models trained on each data set are tested on all four data sets with different implementations. As can be seen from the confusion matrix, 1D-CNN-4 fails to generalize across different implementations (as the accuracy is around 10%). This indicates that the inherent information in each trace set is unique, and the attacking model 1D-CNN-4 may not be easily transferred across different trace sets. Transferability between BRAMs+LUT-MVAU and LUTRAMs+LUT-MVAU (as indicated by an accuracy of 15.30% and 14.99%) is relatively higher and more consistent, hinting at the presence of underlying common patterns. We leave this for future exploration.

LUT-DSP	38.10	9.60	10.60	11.80
BRAM-DSP	11.19	39.56	11.59	11.59
BRAM-LUT	10.30	10.90	39.60	15.30
LUT-LUT	10.49	11.09	14.99	34.57
	LUT-DSP	BRAM-DSP	BRAM-LUT	LUT-LUT

Fig. 6. Confusion matrix displaying the cross-implementation semantic leakage accuracy. Attacking models trained on each data set are tested on all four data sets with different implementations. In each label, the first and the second term indicate the storage element and computation block, respectively.

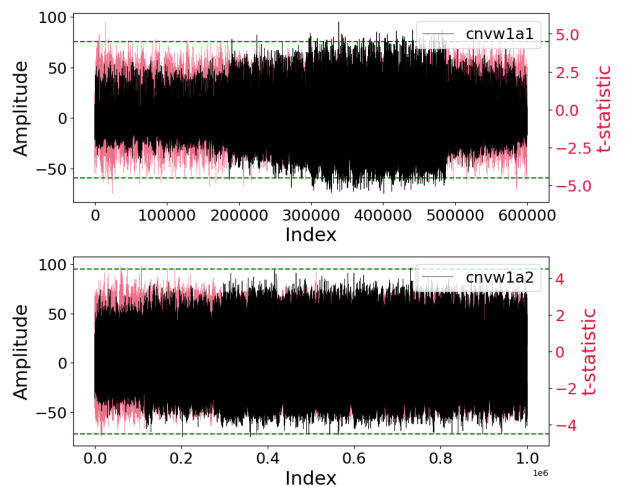


Fig. 7. TVLA on CNV-w1a1 and CNV-w1a2

D. TVLA Analysis on Different Quantized Models

Test Vector Leakage Assessment (TVLA) [36] assesses whether an implementation exhibits data-dependent side-channel leakage by analyzing differences in traces generated from fixed and random test vectors. We provide TVLA analysis of the same model with different quantization transformations, where weights and activations are quantized into different formats. We use two pre-quantized models, denoted by CNV-w1a1 and CNV-w1a2, in which weights (w) and activations (a) are quantized to different bit formats. In semantic leakage, inputs belonging to a specific class (e.g., class 0) are treated as the fixed set and compared against inputs from all other classes (class 0 vs. rest) to test for class-dependent side-channel leakage. Figure 7 shows the TVLA results obtained from two quantized models, where inputs from class 0 are compared against rest. It can be observed that CNV-w1a2 exhibits a longer inference time, as its quantization introduces overhead, but the leakage characteristics of both designs are similar.

E. Future Research

We identify a number of interesting future directions as follows.

Simulation for leakage analysis. As an open-source framework for AI accelerator, FINN could be used for simulation that allows to observe internal signal activity during model execution. These simulation features can be leveraged for leakage analysis by capturing intermediate calculations and internal signals without measuring the hardware. By analyzing simulated traces under controlled inputs, it is possible to study data-dependent behavior, identify potential side-channel leakages, and evaluate the effectiveness of design choices or countermeasures at an early stage of the development flow.

Countermeasures. Semantic leakage in implementations can be mitigated through existing hardware-level countermeasures. Hardware defenses such as electromagnetic (EM) shielding, clock jitter [37], and masking [38] techniques aim to obscure data-dependent signal patterns by reducing the correlation between sensitive information and physical emissions, at the cost of increased design complexity, or performance overhead. In practice, effective protection typically requires a combination of approaches to achieve strong mitigation.

IV. CONCLUSION

In this paper, we explore semantic leakage in FPGA implementations of neural networks compiled by the open-source tool FINN. We show that FINN-generated FPGA designs of neural networks also introduce semantic leakage, similar to Vitis AI. Thanks to the flexibility provided by FINN, we evaluate the semantic leakage using different storage and arithmetic computation modules. We analyzed the influence of different implementation choices in the AMD FPGA design, focusing on differentiating basic and board-specific blocks, including only the most obvious building choices designers could make. We experimentally demonstrate how different implementation choices impact semantic leakage strengths. More importantly, we evaluate across different folding factors, varying the number of PEs and SIMD. Future analyses could consider more varied design choices. Our results establish that implementations with higher folding factors are more resistant to semantic leakage side-channel attacks, providing insights towards side-channel aware implementation strategies.

ACKNOWLEDGMENT

This work was supported by the European Union under Grant Agreement No. 101225691 QARC and by Dutch Research Council (NWO) through the following projects: PROACT (NWA.1215.18.014), TTW PREDATOR 19782, TTW BASES 20858, the CiCS project of the research programme Gravitation (024.006.037) and Open competition project ESCAPE (OCENW.XL.23.112).

REFERENCES

- [1] Xilinx Inc., "Subaru new generation eyesight powered by xilinx," 2020, accessed: 2026-03-03. [Online]. Available: <https://www.xilinx.com/publications/presentations/subaru-xilinx-media-presentation.pdf>
- [2] S. Zeng, J. Liu, G. Dai, X. Yang, T. Fu, H. Wang, W. Ma, H. Sun, S. Li, Z. Huang *et al.*, "FlightLLM: Efficient large language model inference with a complete mapping flow on FPGAs," in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*, 2024.
- [3] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "SoK: Security and privacy in machine learning," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2018.
- [4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR)*, 2014.
- [5] N. Carlini, D. Paleka, K. D. Dvijotham, T. Steinke, J. Hayase, A. Cooper, K. Lee, M. Jagielski, M. Nasr, A. Conmy *et al.*, "Stealing part of a production language model," in *International Conference on Machine Learning (ICML)*, 2025.
- [6] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE Symposium on Security and Privacy (S&P)*, 2017.
- [7] P. Horváth, D. Lauret, Z. Liu, and L. Batina, "SoK: neural network extraction through physical side channels," in *USENIX Security Symposium*, 2024.
- [8] X. Hu, L. Liang, S. Li, L. Deng, P. Zuo, Y. Ji, X. Xie, Y. Ding, C. Liu, T. Sherwood *et al.*, "DeepSniffer: A DNN model extraction framework based on learning architectural hints," in *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020.
- [9] Ł. Chmielewski and L. Weissbart, "On reverse engineering neural network implementation on GPU," in *Applied Cryptography and Network Security (ACNS) Workshops*, 2021.
- [10] H. T. Maia, C. Xiao, D. Li, E. Grinspun, and C. Zheng, "Can one hear the shape of a neural network?: Snooping the gpu via magnetic side channel," in *USENIX Security Symposium*, 2022.
- [11] P. Horváth, Ł. Chmielewski, L. Weissbart, L. Batina, and Y. Yarom, "CNN architecture extraction on edge GPU," in *Applied Cryptography and Network Security (ACNS) Workshops*, 2024.
- [12] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *USENIX Security Symposium*, 2019.
- [13] F. Regazzoni, S. Bhasin, A. A. Pour, I. Alshaer, F. Aydin, A. Aysu, V. Beroulle, G. Di Natale, P. Franzon, D. Hely *et al.*, "Machine learning and hardware security: Challenges and opportunities," in *International Conference on Computer-Aided Design (ICCAD)*, 2020.
- [14] P. Horváth, Ł. Chmielewski, L. Weissbart, L. Batina, and Y. Yarom, "BarraCUDA: Edge gpus do leak dnn weights," in *USENIX Security Symposium*, 2025.
- [15] P. Horváth, I. Shumailov, Ł. Chmielewski, L. Batina, and Y. Yarom, "Kraken: Higher-order em side-channel attacks on dnns in near and far field," in *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2026.
- [16] L. Batina, S. Bhasin, D. Jap, and S. Picek, "Poster: Recovering the input of neural networks via single shot side-channel attacks," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019.
- [17] L. Wei, B. Luo, Y. Li, Y. Liu, and Q. Xu, "I know what you see: Power side-channel attack on convolutional neural network accelerators," in *Annual Computer Security Applications Conference (ACSAC)*, 2018.
- [18] S. Shukla, M. Alam, S. Bhattacharya, P. Mitra, and D. Mukhopadhyay, "Whispering MLaaS: Exploiting timing channels to compromise user privacy in deep neural networks," in *IACR Transactions on Cryptographic Hardware and Embedded Systems (CHES)*, 2023.
- [19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [20] Z. Liu, S. van Hoek, P. Horváth, D. Lauret, X. Xu, and L. Batina, "Real-world edge neural network implementations leak private interactions through physical side channel," *arXiv:2501.14512*, 2025.
- [21] S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina, "SoK: Deep learning-based physical side-channel analysis," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–35, 2023.
- [22] V. Kathail, "Xilinx Vitis unified software platform," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2020.
- [23] Y. Umuroglu, N. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2017.

- [24] M. Blott, T. Preußner, N. Fraser, G. Gambardella, K. O'Brien, Y. Umuroglu, M. Leiser, and K. Vissers, "FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 11, no. 3, pp. 1–23, 2018.
- [25] A. Pappalardo, "Xilinx/brevitas," 2023. [Online]. Available: <https://doi.org/10.5281/zenodo.3333552>
- [26] Piwik PRO, "What is PII (personal data)?" 2024, accessed: 2026-03-03. [Online]. Available: <https://piwik.pro/blog/what-is-pii-personal-data/>
- [27] B. Wang and N. Gong, "Stealing hyperparameters in machine learning," in *IEEE Symposium on Security and Privacy (S&P)*, 2018.
- [28] F. Tramèr, F. Zhang, A. Juels, M. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *USENIX Security Symposium*, 2016.
- [29] T. Orekondy, B. Schiele, and M. Fritz, "Prediction poisoning: Towards defenses against dnn model stealing attacks," in *International Conference on Learning Representations (ICLR)*, 2020.
- [30] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, 2009. [Online]. Available: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>
- [31] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang *et al.*, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.
- [32] AMD, "ZCU104 Evaluation Board," accessed: 2026-03-03. [Online]. Available: <https://www.amd.com/en/products/adaptive-socs-and-fpgas/evaluation-boards/zcu104.html>
- [33] S. Chari, J. Rao, and P. Rohatgi, "Template attacks," in *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2002.
- [34] AMD, "FINN," accessed: 2026-03-03. [Online]. Available: <https://github.com/Xilinx/finn/tree/main/notebooks/advanced>
- [35] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer, 2007.
- [36] G. Goodwill, B. Jun, J. Jaffe, P. Rohatgi *et al.*, "A testing methodology for side-channel resistance validation," in *NIST non-invasive attack testing workshop*, 2011.
- [37] M. Bucci, R. Luzzi, M. Guglielmo, and A. Trifiletti, "A countermeasure against differential power analysis based on random delay insertion," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2005.
- [38] A. Dubey, A. Ahmad, M. Pasha, R. Cammarota, and A. Aysu, "Modulonet: Neural networks meet modular arithmetic for efficient hardware masking," *IACR Transactions on Cryptographic Hardware and Embedded Systems (CHES)*, 2022.