TraceFormer: A Transformer-Based Method for Weight Extraction from AIMC Tiles

Roozbeh Siyadatzadeh*, Fatemeh Mehrafrooz*, Nele Mentens*,†, Todor Stefanov*

*Leiden Institute of Advanced Computer Science (LIACS), Leiden University, The Netherlands
†Department of Electrical Engineering, KU Leuven, Belgium

***Abstract*—Analog In-Memory Computing (AIMC) has emerged as a promising solution to address the performance and energy efficiency limitations of conventional von Neumann architectures for machine learning (ML) applications. This promising approach relies on analog-to-digital converters (ADCs) to enable the integration of AIMC tiles into larger digital systems. In this paper, we investigate the vulnerability of AIMC tiles to power side-channel attacks targeting these ADCs. Specifically, we demonstrate that the numerical values of weights stored in an AIMC tile, that are often a critical asset of an ML model, can be extracted by analyzing the power consumption of the ADCs. With this objective in mind, we propose *TraceFormer*, which is a two-phase method: 1) we train a Transformer neural network (NN) model to translate captured ADC power traces into digital output values; 2) we utilize a novel input-controlled weight isolation technique in order to isolate each individual weight within the AIMC tile, and then reveal the isolated weight's value by ADC power side-channel analysis using the Transformer NN model. We demonstrate the practical applicability and robustness of our proposed method by power side-channel analysis of Oscillator-based ADCs that are typically integrated within AIMC tiles. The experimental results show high accuracy and robustness of our Transformer-based analysis, implying potential vulnerabilities in AIMC tiles.**

***Index Terms*—Analog In-Memory Computing, Vulnerabilities, Power Side-Channel Analysis, ADC Power Analysis.**

I. Introduction

The advancement of machine learning (ML) has led to its widespread adoption in various domains such as image recognition, natural language processing, and autonomous systems. To meet the computational demands of ML models, especially deep neural networks, specialized hardware systems like in-memory computing (IMC) systems have been developed [1]. IMC systems tightly integrate memory and computation, offering significant improvements in speed and energy efficiency when performing ML tasks.

IMC systems are broadly classified into digital in-memory computing (DIMC) systems and analog in-memory computing (AIMC) systems. DIMC systems embed digital logic near memory arrays to perform computations with high precision and noise immunity, although at the cost of lower compute density and sometimes higher energy consumption [2]. In contrast, AIMC systems leverage analog device physics to achieve superior energy efficiency and compute density by using physical properties of devices such as resistance or charge to perform computations directly in the memory array.

As ML models become increasingly integrated into accessible specialized hardware systems containing AIMC tiles, concerns regarding the security of the models and the potential leakage of sensitive information, such as the model topology and weights, arise [3]. Previous research has explored vulnerabilities in AIMC architectures, including methods for extracting ML model topology information from IMC tiles [4]. However, to the best of our knowledge, no existing work has specifically targeted the extraction of the numerical values of weights stored in an AIMC tile. The stored values of the weights after ML model training are very often a critical asset of an ML model because obtaining high-quality data to train ML models remains one of the primary challenges in the ML domain due to, for example, data confidentiality, cost, and the substantial time and effort required for the data collection and model training [5].

Therefore, in this paper, we investigate the vulnerability of AIMC tiles to revealing ML model weights via side-channel analysis, which aims at extracting this potentially sensitive information from the hardware implementation of a system containing AIMC tiles by observing the power consumption of the system. From a security standpoint, the hardware implementation of AIMC tiles might seem inherently robust against power side-channel analysis, as analog computations and circuits themselves do not produce significant switching activity, which is the main source of power side-channel leakage in CMOS-based digital circuits. However, systems containing AIMC tiles necessarily convert analog computation results from the tiles into digital values using analog-to-digital converters (ADCs) that do involve CMOS transistor switching. Consequently, ADCs can serve as potential leakage points vulnerable to power side-channel analysis.

In this paper, we exploit these potential leakage points and demonstrate that extracting ML model weights stored in an AIMC tile is possible by performing power side-channel analysis on the tile's ADCs, thereby revealing AIMC vulnerabilities. More specifically, by carefully controlling and sequencing the inputs provided to the AIMC tile, collecting the corresponding ADC power consumption traces during the analog-to-digital conversion, and then analyzing these traces, we can extract the internal weights of the ML model embedded in the tile. To analyze the power traces and extract the corresponding

This work was supported by European Union (NeuroSoC project - Horizon Europe Grant Agreement n°101070634)

2771-2508/25/$31.00 ©2025 IEEE
DOI 10.1109/DSD67783.2025.00092

weight values, we employ a Transformer neural network (NN) model. The Transformer NN model effectively learns the relationship between the power consumption patterns and the weight values, enabling accurate inference of the weights.

The main novel contributions of this paper are summarized as follows:

- We propose *TraceFormer*, a method to extract ML model weights from an AIMC tile by performing analysis on ADC power consumption traces. Our method utilizes a specific one-hot encoding schema for AIMC tile inputs to isolate the effect of individual weights on the ADC power trace, facilitating accurate extraction of every individual weight value in isolation.
- We construct and train a Transformer NN model to learn the relationships between the power traces, captured during the analog-to-digital conversion process in Oscillator-based ADCs that are typically integrated within AIMC tiles, and the corresponding digital values coming out from this type of ADCs. The Transformer NN model enables accurate extraction of an individual weight value in different ADC operating conditions, such as different temperature, supply voltage, clock speed, and measurement setup.
- We demonstrate the practical applicability of the Transformer NN model by utilizing it for power side-channel analysis of Oscillator-based ADCs that are emulated on a real hardware platform. The experimental results show high accuracy and robustness of our Transformer NN model in translating captured ADC power traces into digital output values, confirming that ADCs are indeed leakage points within AIMC tiles, thereby making such tiles vulnerable. Moreover, the results show that our Transformer NN model is the best option among several NN models, i.e., CNN, RNN, GAN, TCN, and LSTM, we have experimented with, as our model consistently outperforms the others.

The remainder of this paper is organized as follows. Section II discusses related work and provides the context of our investigation. Section III presents an overview of the operational principles of Oscillator-based ADCs and power side-channel analysis. In Section IV, we detail our method for extracting ML model weights stored in an AIMC tile. Section V describes our experimental setup and results. Finally, Section VI concludes the paper and suggests directions for future research.

II. Related Work

In this section, we discuss recent studies that have investigated the application of power analysis and side-channel attacks to reverse engineer ML models running within (embedded) System-on-Chip (SoC) architectures, some of which containing IMC components.

Maji et al. [6] demonstrate that by analyzing power and timing side-channel information, it is possible to recover both the parameters (weight and biases) and inputs of neural networks deployed on embedded microcontrollers. Their experiments reveal that even simple power analysis techniques could effectively extract sensitive information from various network precisions, including floating-point and fixed-point implementations.

Batina et al. [7] explore the feasibility of reverse engineering neural networks using side-channel information. They focus on multilayer perceptrons implemented on ARM Cortex-M3 microcontrollers and demonstrate that an attacker could deduce critical aspects of the network, such as activation functions and layer configurations, solely through non-invasive power measurements. This study highlights the vulnerability of neural network implementations to side-channel attacks.

Neskovic et al. [8] propose a SystemC-based model to estimate information leakage in AI accelerators during the early design stages. They demonstrate successful side-channel attacks, such as correlation power analysis and template attacks, using their model. The study emphasizes the importance of incorporating security evaluations in the design phase to develop attack-resilient AI accelerators.

In recent years, the utilization of ML techniques for power analysis has significantly enhanced the efficacy of side-channel attacks. Researchers have leveraged ML algorithms to analyze power consumption patterns, enabling more precise extraction of sensitive information from cryptographic devices. For example, Ashutosh et al. [9] conduct a comprehensive review of various ML-based power side-channel attacks, focusing on feature extraction techniques and classification methods. Their study highlights the effectiveness of ML in identifying subtle patterns in power traces that traditional analysis methods might overlook, thereby improving the success rate of side-channel attacks.

Moreover, recent studies also have explored power analysis in the context of IMC architectures. For instance, Wang et al. [4] develop a side-channel attack technique targeting IMC systems, demonstrating the feasibility of extracting NN model topology information from power trace measurements without prior knowledge of the neural network. They create a simulation framework to emulate dynamic power traces of IMC macros, highlighting potential vulnerabilities in IMC architectures.

Sayyah Ensan et al. [10] explored the vulnerability of IMC architectures, particularly those utilizing resistive random-access memory (RRAM), to side-channel attacks. They demonstrate that functions implemented within such IMC architectures could be reverse-engineered by analyzing power and timing signatures. Their findings suggest that even without invasive techniques, sensitive intellectual property embedded in IMC architectures can be compromised through side-channel analysis.

Kanellopoulos et al. [11] investigate the security implications of processing-in-memory (PiM) operations, a subset of IMC architectures. They introduced IMPACT, a set of high-throughput main memory-based timing attacks leveraging PiM characteristics to establish covert and side channels. Their study reveals that PiM operations could inadvertently

amplify timing-based side-channel vulnerabilities, leading to significant data leakage risks.

[12] analyzes the vulnerability of Spin Transfer Torque RAM (STTRAM), a promising candidate for last-level cache in IMC systems, to power side-channel attacks. They propose several low-cost countermeasures, such as short retention STTRAM and constant current write drivers, to mitigate these vulnerabilities. Their work underscores the importance of addressing side-channel risks in emerging memory technologies integral to IMC architectures.

While the aforementioned prior research efforts have primarily focused on extracting high-level NN topology information, cryptographic secrets, reverse-engineering functions within IMC or other SoC systems, our proposed method in this paper aims at extracting ML model weights stored in AIMC tiles, thereby highlighting potential and previously unexplored vulnerabilities in systems containing AIMC tiles.

III. Background

In this section, we discuss fundamental principles behind ADCs and their distinctive power consumption characteristics that enable side-channel analysis. Understanding these operational characteristics is essential to comprehending how our proposed method exploits ADC power consumption traces to retrieve the weights of an ML model embedded in an AIMC tile.

ADCs are essential components that convert continuous analog signals into discrete digital representations, and ADCs are integral to AIMC tiles [13]. Many ADC circuits are implemented using the CMOS technology. CMOS circuits exhibit power consumption patterns that are heavily influenced by the transistor switching activity and short-circuit current during the transistors switching, i.e., the dynamic power consumption, as well as the static leakage current, i.e., the static power consumption [14]. The dominant dynamic power consumption P_{dynamic} in CMOS circuits is expressed as

$$P_{\text{dynamic}} = \alpha C V_{\text{DD}}^2 f \tag{1}$$

where α denotes the switching activity factor (the fraction of transistors switching per cycle), C is the effective load capacitance, V_{DD} is the supply voltage, and f is the transistor switching frequency [14]. During ADC operations, each bit decision, comparator switching, or frequency modulation leads to distinct switching patterns, directly influencing the value of α and f. Thus, these operations leave identifiable patterns in the ADC power consumption traces, making the inference of the ADC digital output through power side-channel analysis feasible.

Further, we focus on and briefly discuss Oscillator-based ADCs because this type of ADCs are used in our experimental work and they are typically integrated within AIMC tiles due to their simple design and smaller circuit footprint compared to other ADC types. During operation, an Oscillator-based ADC converts an input voltage signal into a frequency modulated signal. To this end, a voltage-controlled oscillator is used which frequency f_{osc} is directly modulated by the input voltage V_{in} as follows

$$f_{\text{osc}} = f_0 + kV_{\text{in}} \tag{2}$$

where f_0 is the oscillator's nominal frequency and k is its voltage-to-frequency conversion factor [15]. The oscillator triggers digital counting circuits to obtain the digital value corresponding to V_{in}. Moreover, the switching activities of the digital counting circuits correlate directly with the input voltage V_{in} and its corresponding digital representation (value). This results in power consumption traces strongly tied to the input signal due to frequency-dependent switching patterns. Such frequency modulation directly impacts CMOS transistor switching dynamics, leaving distinct power signatures that can be analyzed to infer the digital representation (value) corresponding to V_{in}.

IV. Proposed Method

In this section, we propose *TraceFormer*, a two-phase method to extract the weights of an ML model stored in an AIMC tile using power side-channel analysis on ADCs. To achieve this goal, we rely on the following assumptions. First, as an authorized user or tester of an AIMC-based system, we have full control over the inputs provided to the AIMC tile for computation. Second, we assume that it is possible to capture the ADC power consumption traces, which is a realistic assumption because modern mixed-signal SoCs typically isolate analog components—including ADCs—onto dedicated analog power rails, enabling power measurements via a small series resistor or direct probing on the printed-circuit board [16], [17]. Based on these assumptions, in the first phase, called *Transformer-based Power Trace to Weight Translation*, we train a Transformer NN model to learn the relationships between ADC power consumption patterns and the corresponding digital values coming out from the ADC. After training, this Transformer NN model can accurately translate captured ADC power consumption traces into digital values. However, to achieve our primary objective—extracting the embedded ML model weights—we must ensure that each ADC operation involves exactly one unknown weight at a time. Therefore, in the second phase, called *Input-Controlled Weight Isolation*, we control the AIMC tile inputs using specifically crafted one-hot encoded inputs in order to isolate each individual weight within the AIMC tile, and then reveal the isolated weight's value by ADC power side-channel analysis using the trained Transformer NN model.

A. Phase 1: Transformer-based Power Trace to Weight Translation

This phase focuses on how to translate captured ADC power traces into their corresponding digital values. This is achieved by building a supervised learning pipeline that takes pairs of power traces and known digital outputs as training data. The overall goal is to build a reliable digital output predictor that generalizes across different ADC operating conditions. The following two steps describe how we prepare the training data

and how we design, train, and validate a Transformer NN model that performs this translation with high accuracy.

1) Step 1: Dataset Preparation: In this step, we begin by building a labeled dataset that captures the relationship between ADC power traces and their corresponding digital outputs. This dataset serves as the training foundation for our Transformer NN model.

Each data collection session targets an ADC configured to produce B bits of digital output. Given B bits of digital output, there are 2^B possible output values, ranging from all-zeros to all-ones. We generate corresponding analog inputs that stimulate the ADC to produce each possible digital output value. For every analog input, we capture the resulting ADC power consumption trace, represented as a sequence of power samples $X = [x_1, \ldots, x_L]$, where every sample $x_i \in \mathbb{R}$ and L is the number of samples captured during each conversion process. L stays the same for all captured traces during the dataset generation. The B-bit digital output $Y \in \{0,1\}^B$ is recorded together with each trace X, resulting in the trace-output pair (X, Y).

To capture variations in the ADC operating condition, device behavior, and measurement noise, we repeat the data collection process multiple times for each analog input. Specifically, for every target output value, we apply R times the corresponding analog input under varying conditions such as different temperatures, supply voltages, and measurement setups. This process generates $C \times R$ number of trace-output pairs (X, Y) in the training dataset, where $C = 2^B$ is the number of unique output values and R is the number of repetitions per value. The parameters C, R, and L are configurable and chosen to balance between dataset coverage and practical limitations on storage and measurement time. This dataset forms the basis for training the Transformer NN model described in Step 2.

2) Step 2: Transformer NN Model: To reliably infer digital outputs from raw ADC power traces, we construct and train a Transformer NN model that learns intricate temporal dependencies directly from captured current measurements. Each ADC power trace is initially represented by a real-valued vector $X = [x_1, \ldots, x_L]$, where every sample $x_i \in \mathbb{R}$ and L denotes the number of current samples collected per ADC conversion. Due to the large number of samples and redundancy present in raw power trace data, direct modeling is computationally prohibitive, motivating the need for effective pre-processing steps.

First, we normalize each trace individually, rescaling sample to the interval $[0, 1]$. Such normalization ensures numerical stability during training and standardizes input variations across diverse ADC conversions. Subsequently, we apply Principal Component Analysis (PCA) [18] to reduce the number of elements of each trace while retaining critical variance information essential for distinguishing between different ADC output patterns. Specifically, PCA converts the original potentially long trace X into a shorter trace $X' = [x'_1, \ldots, x'_{L'}]$, where $x'_i \in [0, 1]$ and $L' \ll L$.

Following the PCA conversion, we segment the shorter trace X' into scalar tokens suitable for the Transformer NN model processing. Each scalar token x'_t is individually embedded into a latent vector space of dimension d (embedding size), using a learned affine transformation characterized by weight matrix $W_e \in \mathbb{R}^{d \times 1}$ and bias vector $b_e \in \mathbb{R}^d$:

$$e_t = W_e x'_t + b_e, \qquad e_t \in \mathbb{R}^d, \quad t \in \{1, \ldots, L'\}. \tag{3}$$

Additionally, we prepend a special classification token ([CLS]) at the beginning of each token sequence. This token enables the model to efficiently aggregate global contextual information about the entire trace, facilitating accurate prediction of the underlying digital values.

To effectively handle temporal jitter and subtle timing variations inherent in practical ADC measurements, we integrate relative positional encoding [19] into our Transformer architecture. Relative positional encoding explicitly represents pairwise temporal distances among tokens, rather than relying on fixed absolute positions. This adaptive representation significantly enhances the model's resilience to temporal shifts and jitter, enabling the Transformer to accurately capture timing-sensitive information within ADC traces.

The sequence of embedded tokens, denoted as

$$E = [e_{[\text{CLS}]}, e_1, \ldots, e_{L'}] \in \mathbb{R}^{(L'+1) \times d} \tag{4}$$

is processed through a stack of N_L Transformer encoder layers, where N_L denotes the number of stacked encoder layers. Each encoder layer comprises two primary components: a multi-head self-attention (MHSA) module and a position-wise feed-forward network (FFN). The Transformer employs residual connections around both modules to facilitate stable gradient flow and effective training:

$$E' = \text{MHSA}(E) + E, \tag{5}$$

$$E_{\text{out}} = \text{FFN}(E') + E' \tag{6}$$

To preserve amplitude-sensitive information critical for effective side-channel analysis, we deliberately omit layer normalization, which could otherwise suppress subtle yet meaningful amplitude variations in the input traces.

The final encoder output associated with the special [CLS] token, denoted $\mathbf{h}_{[\text{CLS}]} \in \mathbb{R}^d$, provides a concise summary of the entire input power trace. To predict the digital ADC output, we apply a linear transformation with parameters $W_{\text{out}} \in \mathbb{R}^{N \times d}$ and $b_{\text{out}} \in \mathbb{R}^N$, mapping the summarized trace representation to a vector of logits:

$$\mathbf{o} = W_{\text{out}} \mathbf{h}_{[\text{CLS}]} + b_{\text{out}}. \tag{7}$$

Here, N represents the bit-resolution of the targeted ADC. Subsequently, we apply an element-wise sigmoid activation function, converting the logits to bit-wise probability estimates indicating the likelihood that each bit is one:

$$\hat{\mathbf{Y}} = \sigma(\mathbf{o}), \qquad \hat{\mathbf{Y}} \in [0, 1]^N. \tag{8}$$

Training the Transformer model involves minimizing the binary cross-entropy loss computed independently for each

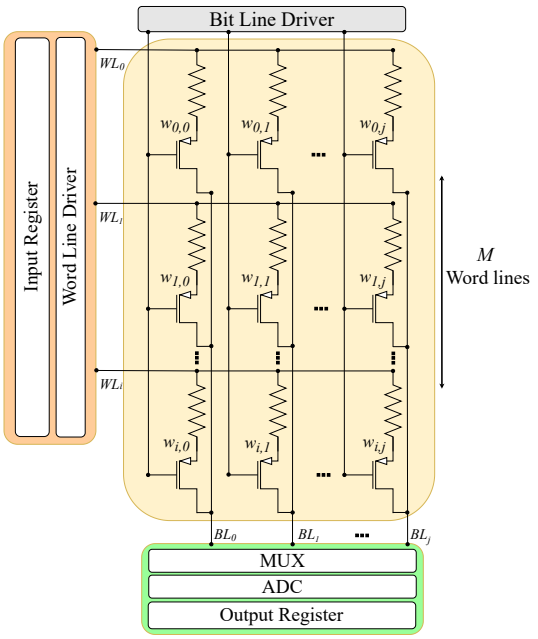


Fig. 1. Structure of an AIMC tile with M word lines. Each crosspoint stores a weight $W_{i,j}$. The Word Line Driver applies input activations to the rows. A multiplexer (MUX) sequentially forwards each bit line to the ADC for digitization, with results stored in the Output Register.

ADC output bit. Given a ground-truth bit-vector $\mathbf{Y} \in \{0,1\}^N$, the training loss is formally expressed as:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\Big[Y_i \ln \hat{Y}_i + (1-Y_i)\ln\big(1-\hat{Y}_i\big)\Big]. \quad (9)$$

The optimization is performed using the AdamW algorithm with specified batch size B (number of traces per training batch) and learning rate lr. Training continues for a predetermined number of epochs, leveraging mixed-precision arithmetic to enhance computational efficiency. Additionally, we incorporate four targeted data augmentation techniques such as uniform random time shifts, additive Gaussian noise, amplitude scaling, and mild temporal warping in order to promote robust generalization against realistic measurement variations encountered during ADC operations.

B. Phase 2: Input-Controlled Weight Isolation

After training the Transformer NN model to translate ADC power traces into digital values in Phase 1, we utilize it to extract individual weights embedded in an AIMC tile. This phase leverages a sequence of one-hot encoded inputs provided to the tile in order to isolate the effect of individual weights on the ADC output, making possible the extraction of each weight from the captured ADC power traces.

As illustrated in Figure 1, an AIMC tile consists of a crossbar array of memory elements arranged along M word lines (rows) and J bit lines (columns). Each crosspoint at coordinate (i, j) stores a weight $W_{i,j}$, typically implemented as a programmable conductance. Input voltages WL_i are applied to the array through the word lines, while the resulting column-wise currents BL_j are collected on the bit lines. The analog computation performed by the tile corresponds to a vector-matrix multiplication, where the current BL_j on bit line j is given by

$$\text{BL}_j = \sum_{i=1}^{M} W_{i,j} \cdot \text{WL}_i \quad (10)$$

where WL_i is the voltage applied to word line i.

To isolate, the contribution of a specific weight $W_{i,j}$ to the current BL_j, we put a one-hot encoded input in the Input Register, shown in Figure 1, that activates the Word Line Driver to apply non-zero voltage only to the i-th word line (e.g., WL_i = 1V) while applying zero volts to all other lines:

$$\forall k \in [1, M], \quad \text{WL}_k = \begin{cases} 1, & \text{if } k = i, \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

Under this condition, the current on bit line j, expressed by Equation 10, becomes

$\text{BL}_j = W_{i,j}$

meaning that the current on bit line j is directly proportional to the isolated weight $W_{i,j}$ at position (i, j) in the AIMC tile crossbar array.

The Bit Line Driver routes the currents to the multiplexer component MUX depicted at the bottom of Figure 1. The MUX selects one bit line j at a time and forwards its analog signal (current BL_j) to the ADC for digitization. This time-multiplexed readout of the outputs of the AIMC tile crossbar array ensures that, during each ADC conversion cycle, exactly one analog signal proportional to the isolated weight $W_{i,j}$ is presented to the ADC as input. The ADC converts this analog signal (input) to a digital output while generating a distinctive power trace resulting from internal CMOS switching activity. This trace reflects the specific characteristics of the digitized value, providing a side-channel signal that can be captured externally. Using the Transformer NN model trained in Phase 1, we analyze this power trace (side-channel signal) to predict the digital representation of the isolated weight.

The aforementioned process is repeated for all word lines and bit lines. For each isolated weight $W_{i,j}$, multiple traces are captured across P independent repetitions to improve measurement reliability through averaging and noise reduction. The controlled on-hot encoded input to the AIMC tile, combined with the sequential selection of bit lines via the MUX and the high-resolution analysis of the ADC power traces using the trained Transformer NN model, provides an effective method for extracting all weights embedded in the AIMC tile.

V. Experimental Evaluation

In this section, we demonstrate the practical applicability of our Transformer NN model by training and utilizing it for power side-channel analysis of an Oscillator-based ADC which is emulated on a real hardware setup. The goal is to evaluate the accuracy and robustness of our model in translating captured ADC power traces into digital output values, thereby experimentally confirming or not that Oscillator-based

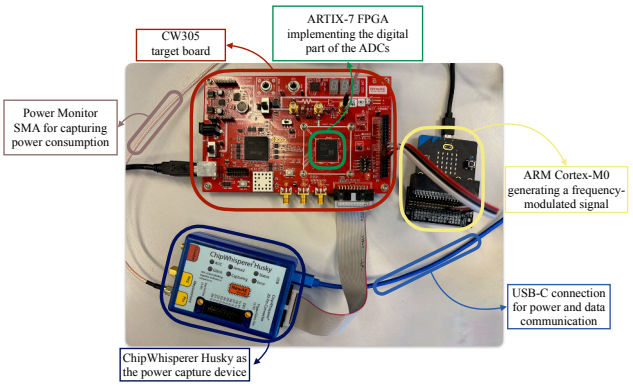


Fig. 2. An overview of the experimental setup with CW305, ARM Cortex-M0, and ChipWhisperer Husky for power trace capturing.

ADCs are potential leakage points if integrated within AIMC tiles. In addition, we compare our Transformer NN model with several other NN models, i.e., CNN, RNN, GAN, TCN, and LSTM to check whether our model is the best option for power side-channel analysis of Oscillator-based ADCs.

A. Experimental Setup

The comprehensive experimental hardware setup is illustrated in Figure 2. It comprises a ChipWhisperer platform consisting of a CW305 Artix-7 FPGA target board and a ChipWhisperer Husky power trace capture device. The FPGA implements the digital back-end of an Oscillator-based ADC [20]. The analog front-end of the ADC, i.e., the voltage-controlled oscillator is emulated on a MicroBit board featuring an ARM Cortex-M0 microcontroller [21]. The microcontroller generates a digitally programmable pulse-width-modulated (PWM) waveform that feeds the digital back-end of the Oscillator-based ADC implemented in the FPGA. The ADC's counter in the digital back-end treats this waveform as the oscillations it would normally receive from its analog front-end that trigger the counter to increment its digital value on the rising edges of the PWM pulses. All PWM waveform parameters, including the carrier frequency, modulation depth, and modulation rate, are controlled via the UART interface of the MicroBit board. This allows us to have fully controlled waveform generation without firmware modifications, supporting repeatable waveforms across different rounds of experiment.

The ChipWhisperer Husky capture device samples the FPGA's instantaneous power consumption at a temporal resolution of 105 mega samples per second (MS/s) with 12-bit amplitude resolution per sample. It captures high-quality power consumption traces, corresponding to the ADC digital back-end operations, and transmits the data to a host computer via a USB-C interface for further power side-channel analysis utilizing our Transformer NN model.

B. Evaluation of the Transformer NN model

In this section, we evaluate the accuracy and robustness of the Transformer NN model (Section IV-A) in inferring ADC digital outputs from power consumption traces. First, we explain how the training dataset for the model is generated using the experimental setup. Then, we introduce the architecture of the Transformer NN model, we have constructed, and the training parameters. Finally, we present our experimental results and compare the performance of our Transformer NN model with several alternative NN models.

1) Dataset Generation: As mentioned earlier in Section V-A, we conduct our experiments using the setup shown in Figure 2. The power traces are captured, via the ChipWhisperer Husky platform by sampling at 105 MS/s, from the digital back-end of the oscillator-based ADC implemented on the ARTIX-7 FPGA of the CW305 board operating at a supply voltage of 0.9 V and clock frequency of 40 MHz.

The dataset is generated for the oscillator-based ADC configured with a resolution of the output $B = 8$ bits, covering all 256 possible digital output values. For each digital output value, we perform $R = 100$ repetitions of data collection (power trace capturing) to account for variations due to device behavior, noise, and environmental conditions, but with the aforementioned fixed voltage (0.9 V) and clock frequency (40 MHz) of the FPGA, resulting in a total dataset size of 25600 trace-output pairs (X, Y). Each power trace X contained exactly $L = 1024$ uniformly captured samples.

2) Transformer NN model Architecture and Training parameters: Our Transformer NN model architecture is described in the last row of Table I. The model starts by applying PCA to reduce each ADC power trace X with length $L = 1024$ samples to trace X' with $L' = 512$ scalar tokens. This step significantly decreases the computational complexity of the model, yet preserving essential trace characteristics. Each scalar token $x'_t \in X'$ is embedded into a learned latent vector space with an embedding width of $d_{\text{model}} = 64$. A special classification ([CLS]) token is prepended to facilitate global information aggregation. Next, the model employs four Transformer encoder layers ($N_L = 4$), each consisting of a MHSA module with 8 heads and a position-wise FFN with dimension 128.

The model architecture is completed with a Dense layer containing 8 neurons and a sigmoid activation function for bit-wise prediction. The AdamW optimizer is utilized with a learning rate of 5×10^{-4}, batch size of 16, dropout rate of 0.1, and the model is trained for 150 epochs to ensure robust convergence.

3) Experimental Results and Comparison: In Figure 3, we present the accuracy of our Transformer NN model (the green bar) in comparison to other models such as CNN, RNN, GAN, TCN, and LSTM. Every bar shows the accuracy of the corresponding model indicated on the x-axis. The architectures of all models are described in Table I. For fair comparison, the data pre-processing and the training dataset are the same for all models, i.e., all models are trained with the same dateset of 25600 trace-output pairs (X, Y) and every trace is reduced in length with the aforementioned PCA. Moreover, the accuracy of all models is evaluated on the same test dataset with unseen power consumption traces, collected under the

TABLE I
MODEL ARCHITECTURES AND TRAINING PARAMETERS USED FOR BENCHMARKING POWER-TRACE-BASED INFERENCE OF ADC OUTPUT VALUES

Model Type	Layers	Optimizer	Learning Rate	Batch Size	Epochs
CNN	Conv1D(32, 5) → ReLU → MaxPool(2) → Conv1D(64, 5) → ReLU → GlobalAvgPool → Dense(8) → Sigmoid	Adam	1e-3	32	60
RNN	SimpleRNN(64) → Dense(8) → Sigmoid	Adam	1e-3	32	110
GAN	Generator: Dense(256) → ReLU → Dense(1024)				
	Discriminator: Dense(256) → ReLU → Dense(1) → Sigmoid	Adam	1e-4	64	170
TCN	TCN(64, kernel_size=3, dilations=[1,2,4]) → Dense(8) → Sigmoid	Adam	1e-3	32	140
LSTM	LSTM(64) → Dense(8) → Sigmoid	Adam	1e-3	32	170
Transformer	Tokenize → Positional Encoding → 4x Encoder(MHSA(8 heads), FFN(128)) → Dense(8) → Sigmoid	AdamW	5e-4	16	150

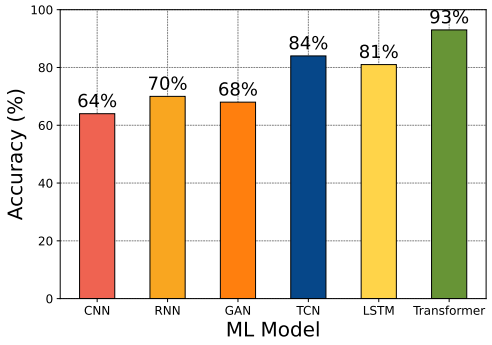
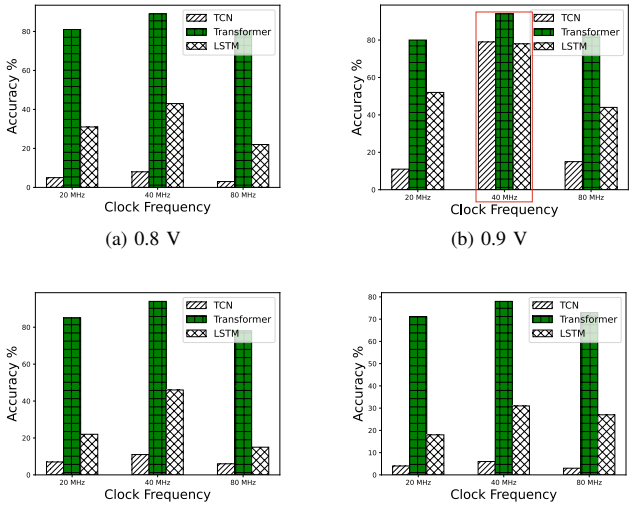


Fig. 3. Accuracy of different machine learning models in predicting ADC outputs from power traces.



(a) 0.8 V (b) 0.9 V

(c) 1 V (d) 1.1 V

Fig. 4. The accuracy of our Transformer NN, TCN, and LSTM models under varying ADC operating conditions, i.e., different supply voltages (0.8 V, 0.9 V, 1.0 V, 1.1 V) and clock frequencies (20 MHz, 40 MHz, 80 MHz). In bar chart (b), the red box highlights the conditions (i.e., 0.9 V, 40 MHz) under which the training dataset is obtained.

same FPGA voltage (0.9 V) and clock frequency (40 MHz) as the training data. The accuracy results, shown in Figure 3, clearly indicate that our Transformer NN model outperforms the other models by consistently achieving higher accuracy, thus justifying our choice of a Transformer type of NN model for effective inference of Oscillator-based ADC output values from power consumption traces.

To evaluate the robustness of our Transformer NN model, we asses its accuracy to infer ADC output values from power consumption traces that are captured under ADC operating conditions different from the conditions at which the aforementioned training dataset is generated. More specifically, we vary the FPGA clock frequency using three values, i.e., 20 MHz, 40 MHz, and 80 MHz as well as we vary the FPGA supply voltage using four values, i.e., 0.8 V, 0.9 V, 1.0 V, and 1.1 V. Changes in the frequency and voltage alter the timing of switching activities in the ADC digital backend circuit, thereby affecting the power consumption traces. Nevertheless, the fundamental pattern of switching activity remains consistent.

The bars in Figure 4 show the accuracy of the three best-performing models, identified from Figure 3 (our Transformer NN, TCN, and LSTM), across the aforementioned varying ADC operating conditions. As a reference point, the red box in Figure 4(b) highlights the accuracy of the three models on traces captured under the same ADC operating conditions as the traces used for the training. The green bars in Figure 4 clearly indicate the high robustness of our Transformer NN model because its accuracy remains very high even under varying ADC operating conditions. In contrast, the gray bars indicate that the TCN and LSTM models are not robust because their accuracy drops down significantly in comparison to the reference point.

VI. CONCLUSIONS

In this paper, we introduced *TraceFormer*, a Transformer-based method specifically designed to extract numerical weights from Analog In-Memory Computing (AIMC) tiles through power side-channel analysis targeting analog-to-digital converters (ADCs). Recognizing ADCs as inherent vulnerabilities within AIMC architectures, we developed a structured two-phase methodology. In the first phase, a Transformer neural network is trained to accurately decode ADC power consumption traces into their corresponding digital output values. In the second phase, we presented an input-controlled weight isolation method using one-hot encoded inputs to individually isolate and extract each weight embedded within an AIMC tile.

We validated *TraceFormer* experimentally using an FPGA-implemented Oscillator-based ADC, demonstrating that our proposed method significantly outperforms other neural net-

work models, including CNN, RNN, GAN, TCN, and LSTM architectures, in terms of both accuracy and robustness. Additionally, *TraceFormer* showed superior generalization under varying operational conditions, including changes in supply voltage and FPGA clock frequency.

Our findings underline the vulnerability of AIMC-based systems due to ADC components, clearly identifying them as critical leakage points susceptible to power side-channel attacks. Future research directions include exploring and implementing robust countermeasures, as well as extending the applicability of *TraceFormer* to additional types of ADCs and diverse AIMC hardware configurations.

REFERENCES

[1] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnology*, vol. 15, no. 7, pp. 529–544, Jul. 2020, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41565-020-0655-z

[2] J. Sun, P. Houshmand, and M. Verhelst, "Analog or digital in-memory computing? benchmarking through quantitative modeling," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–9.

[3] Z. Pan and P. Mishra, "A Survey on Hardware Vulnerability Analysis Using Machine Learning," *IEEE Access*, vol. 10, pp. 49 508–49 527, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9770796

[4] Z. Wang, F.-H. Meng, Y. Park, J. K. Eshraghian, and W. D. Lu, "Side-channel attack analysis on in-memory computing architectures," *IEEE Transactions on Emerging Topics in Computing*, vol. 12, no. 1, pp. 109–121, 2024.

[5] A. Jiménez-Sánchez, S. Albarqouni, and D. Mateus, "Capsule networks against medical imaging data challenges," in *Intravascular Imaging and Computer Assisted Stenting and Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*. Cham: Springer International Publishing, 2018, pp. 150–160.

[6] S. Maji, U. Banerjee, and A. P. Chandrakasan, "Leaky nets: Recovering embedded neural network models and inputs through simple power and timing side-channels—attacks and defenses," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12 079–12 092, 2021.

[7] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 515–532. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/batina

[8] A. Nešković, S. Mulhem, A. Treff, R. Buchty, T. Eisenbarth, and M. Berekovic, "Systemc model of power side-channel attacks against ai accelerators: Superstition or not?" in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2023, pp. 1–8.

[9] A. Ghimire, V. V. Baligodugula, and F. Amsaad, "Power analysis side-channel attacks on same and cross-device settings: A survey of machine learning techniques," in *Internet of Things. Advances in Information and Communication Technology*, D. Puthal, S. Mohanty, and B.-Y. Choi, Eds. Cham: Springer Nature Switzerland, 2024, pp. 357–367.

[10] S. Sayyah Ensan, K. Nagarajan, M. N. I. Khan, and S. Ghosh, "Scare: Side channel attack on in-memory computing for reverse engineering," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 12, pp. 2040–2051, 2021.

[11] K. Kanellopoulos, F. N. Bostanci, A. Olgun, A. G. Yaglikci, I. E. Yuksel, N. M. Ghiasi, Z. Bingol, M. Sadrosadati, and O. Mutlu, "Amplifying main memory-based timing covert and side channels using processing-in-memory operations," 2024. [Online]. Available: https://arxiv.org/abs/2404.11284

[12] A. Iyengar, S. Ghosh, N. Rathi, and H. Naeimi, "Side channel attacks on sttram and low-overhead countermeasures," in *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2016, pp. 141–146.

[13] B. Razavi, *Design of Analog CMOS Integrated Circuits*. McGraw-Hill Higher Education, 2016. [Online]. Available: https://books.google.nl/books?id=hFzmCwAAQBAJ

[14] J. Rabaey, A. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits: A Design Perspective*, ser. Prentice Hall electronics and VLSI series. Pearson Education, 2003. [Online]. Available: https://books.google.nl/books?id=_7daAAAAYAAJ

[15] J. Kim and S. Cho, "A time-based analog-to-digital converter using a multi-phase voltage controlled oscillator," in *2006 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2006, pp. 4 pp.–3937, iSSN: 2158-1525. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1693489

[16] Kevin Chesser, May Porley, "What Are the Basic Guidelines for Layout Design of Mixed-Signal PCBs? | Analog Devices." [Online]. Available: https://www.analog.com/en/resources/analog-dialogue/articles/what-are-the-basic-guidelines-for-layout-design-of-mixed-signal-pcbs.html

[17] V. Gavriilidou, A. Voulkidou, T. Noulis, N. Codreanu, and C. Ionescu, "System on Chip Noise Integrity Simulation," *Chips*, vol. 1, no. 1, pp. 14–29, Jun. 2022, number: 1 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2674-0729/1/1/3

[18] A. Tharwat, "Principal component analysis-a tutorial," *International Journal of Applied Pattern Recognition*, vol. 3, no. 3, pp. 197–240, 2016.

[19] S. Hajra, S. Saha, M. Alam, and D. Mukhopadhyay, "TransNet: Shift Invariant Transformer Network for Side Channel Analysis," in *Progress in Cryptology - AFRICACRYPT 2022*, L. Batina and J. Daemen, Eds. Cham: Springer Nature Switzerland, 2022, pp. 371–396.

[20] J. Wu, S. Hansen, and Z. Shi, "ADC and TDC implemented using FPGA," in *2007 IEEE Nuclear Science Symposium Conference Record*, vol. 1, Oct. 2007, pp. 281–286, iSSN: 1082-3654. [Online]. Available: https://ieeexplore.ieee.org/document/4436331

[21] "nRF51822 Product Specification v3.4." [Online]. Available: https://docs.nordicsemi.com/bundle/nRF51-Series/resource/nRF51822_PS_v3.4.pdf