NO TPU LEFT BEHIND: RETROFITTING SIDE-CHANNEL PROTECTION INTO EDGE TPUS

Anonymous authors
Paper under double-blind review

ABSTRACT

Side-channel attacks can recover neural network parameters from physical signals, even on commercial edge accelerators. Existing defenses require changes to hardware, instruction set, or compiler, and cannot be deployed on fixed-function platforms such as TPUs. We present the first training-time defense that protects models on off-the-shelf TPUs without modifying the hardware or firmware. Our approach trains multiple functionally equivalent parameter versions per layer and randomly composes them at inference. This reduces the correlation that side-channel attacks rely on while preserving model accuracy. We enforce diversity between parameter versions by adding a regularization term in the loss function during training. We show that this diversity increases leakage variance while leaving the mean signal unchanged, which provably reduces the signal-to-noise ratio exploited by attackers. We derive theoretical bounds that relate leakage to the number of parameter versions and their pairwise distance, and provide a simple calibration method to predict leakage for new configurations before deployment or side-channel measurements. We implement our method on a Google Edge TPU and evaluate it on representative and real-world models. Our defense, in a high-diversity configuration, can hide leakage by reducing the Test Vector Leakage Assessment t-score below the standard leakage detection threshold of 4.5 with less than 1% accuracy change and moderate overhead. Our results thus show, for the first time, that training-time defenses can provide practical side-channel protection for widely deployed AI hardware.

1 INTRODUCTION

Machine learning models are high-value intellectual assets, representing substantial investments in computational resources, expert labor, and proprietary data collection. Although their superior performance has led to widespread adoption in diverse applications, this ubiquity has been met with a rise in model theft attacks (61; 50; 49). These adversaries aim to produce high-fidelity (34) replicas by targeting learned parameters (weights and biases) (10; 23). Because model theft offers a low-cost alternative to the intensive effort of training from scratch, it has become an increasingly appealing strategy for attackers. Furthermore, successful model extraction facilitates more sophisticated threats, such as membership inference (56; 57) and input poisoning (13). Ultimately, these vulnerabilities undermine the reliability of AI systems and compromise safety-critical applications, making model protection a fundamental requirement for AI security.

Side-channel attacks with physical measurements represent a distinct and growing threat to model confidentiality, particularly for edge AI devices deployed in uncontrolled environments (36). Edge AI offers real-time inference by operating in close proximity to end-users over data center processing (59). However, this physical accessibility exposes them to side-channel exploitation, where adversaries monitor unintended physical emanations such as power or electromagnetic (EM) signals to extract sensitive information. Prior literature underscores the gravity of this threat, demonstrating the high-accuracy recovery of model parameters across multiple target platforms (32; 24; 66; 43; 31; 65).

In response to these vulnerabilities, various defense mechanisms have been proposed ranging from execution shuffling (9; 20), and pixel pruning (16) to masking (19; 18; 17; 8), and hardware-level modifications (68; 64). These mechanisms either need hardware customization, architecture/compiler support for low-level arithmetic/boolean instructions, or both. By contrast, existing edge AI platforms, such as TPUs do not have these properties, leaving them vulnerable. This creates a significant gap for a defense mechanism that does not require hardware or instruction set modifications and can be used as a plug-and-play solution.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

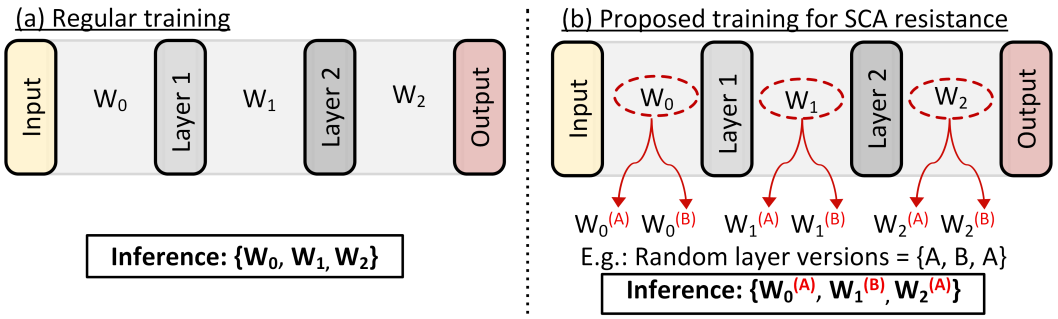


Figure 1: Overview of the proposed defense. **(a)** Standard model inference utilizes a single, static set of weights (e.g., $\{W_0, W_1, W_2\}$) for every execution, producing deterministic side-channel leakage. **(b)** Our proposed approach purposefully trains multiple functionally equivalent weight versions per layer (e.g., versions A and B). During each inference, one weight version per layer is randomly selected. For example, a random selection of $\{A, B, A\}$ layer versions configures the network to use $\{W_0^{(A)}, W_1^{(B)}, W_2^{(A)}\}$. This dynamic, layer-wise shuffling reduces the side-channel leakage.

In this paper, we propose a fundamentally new technique that can be retrofitted into existing, commercial off-the-shelf TPUs—the first one to do so. Our key idea is to use a novel training strategy that generates multiple parameter versions for each layer in a model, ensuring that any combination of these versions is composable while maintaining near-baseline accuracy. By doing so, these layer versions can be dynamically swapped during inference, presenting different parameter variants for every inference. To ensure these versions remain distinct and do not converge to identical weights, we leverage stochastic mini-batch gradient descent and introduce a regularization term that enforces parameter dissimilarity during training. Our evaluation is grounded in the theoretical dissimilarity between model versions and is validated empirically through side-channel Test Vector Leakage Assessment (TVLA) metric across representative and real-world models and datasets. The main contributions of this paper include the following:

- We identify a critical dependency of existing side-channel countermeasures on specialized hardware, firmware, or compiler modifications. These requirements make current defenses incompatible with specialized, off-the-shelf devices like Google Edge TPUs.
- To address the limitations of existing defenses, we propose the first countermeasure that requires no architectural, hardware, compiler or firmware changes, demonstrated on a Google Edge TPU and tested against adaptive adversary. Our approach leverages multiple parameter versions shuffled during inference to impede extraction. We ensure diversity between versions by introducing stochasticity and dissimilarity enforcing regularization term in the loss function, which increases the Euclidean distance between versions during training. We also propose a method to embed this shuffling behavior into the target TPU by stacking multiple weights in a matrix and using one-hot encoding to select between them.
- We establish the mathematical foundation that connects side-channel resilience to the model dissimilarity between versions of the trained parameters. Building on this theory, this work demonstrates how leakage scales with the number of versions and introduces a methodology to estimate leakage for a given platform, version count, and parameter distance.
- We comprehensively evaluate our defense across diverse datasets and architectures, encompassing both representative and real-world models. The evaluation spans full models and highly vulnerable layers of EfficientNet, which was previously attacked (32). The side-channel resistance is quantified using TVLA while analyzing inference overhead and accuracy change to ensure real-world practicality. The results show a security improvement of $3.81 - 48.04\times$ with $3.43 - 5.14\times$ increase in inference time, $2.29 - 4.33\times$ model size increase and 0.3-1.3% accuracy change.

2 Motivation, Design Rationale and Threat Model

2.1 Related Work and Our Solution

Parameter stealing attacks aim to recover the weights and biases of an ML model, thereby compromising the developer's intellectual property. While cryptanalytic attacks (14; 10) that use structured queries to target parameters have been demonstrated in Machine Learning as a Service setting, defenses against them exist (40), and they are considered out of scope for this work. In the edge AI setting, however, side-channel analysis (SCA) has emerged as the predominant threat, where

adversaries monitor physical emanations such as power or EM signal during inference to decipher internal model parameters. Prior research has demonstrated the efficacy of these attacks across a diverse landscape of platforms, ranging from low-power microcontrollers to high-performance edge accelerators. For instance, studies have extracted weights from ARM Cortex-M processors, FPGAs (19; 31; 3), edge GPUs (32) and, TPU systolic-array-based accelerators (66; 43; 63). Even when parameters are binarized for efficiency (65) or stored within secure, encrypted hardware IP like AMD-Xilinx DPUs (24), attackers have achieved high-accuracy model recovery. Collectively, these works underscore a systemic vulnerability in the edge AI deployment scenario.

To counteract these side-channel-based parameter extraction attacks, several defense mechanisms have been proposed, yet each introduces deployment barriers and cannot be adopted on existing TPUs. Techniques that shuffle the execution sequence of neuron computations or the order of multiplication (9; 20; 21) rely on the assumption of sequential processing, rendering them incompatible with the parallel execution models of modern TPUs. Similarly, importance-aware pixel pruning (16), which randomly deactivates input pixels and prunes the corresponding operations via random operation skipping, requires hardware and firmware-level changes to skip selected operations dynamically. Other approaches, such as boolean masking (8; 18) and hardware-based garbled circuits (29), provide formal security guarantees but require extensive hardware-level changes for modifying standard arithmetic into specialized leakage-resistant logic primitives. Furthermore, effective masking of feedforward networks necessitates custom hardware-level gate implementations to manage the resulting computational complexity, making them difficult to deploy on general-purpose accelerators (8). Other physical-layer defenses, including clock frequency switching (68) and dual-rail precharge logic (64), are inherently tied to custom ASIC or FPGA designs. Even advanced hardware-software co-design frameworks (8) depend on custom hardware and compilers.

We overcome the limitations of prior hardware- and software-centric defenses by shifting the security to the model training phase. Figure 1 illustrates our proposed methodology which generates multiple distinct versions of each layer and dynamically shuffles them between inferences. By ensuring these distinct layer versions maintain baseline accuracy and are mutually composable, the total number of model permutations scales exponentially; for example, a network with L layers and V versions per layer yields V^L potential model configurations. To prevent these versions from converging to identical weight values during training, we leverage the inherent stochasticity of mini-batch gradient descent alongside a specialized regularization term that explicitly enforces parameter dissimilarity.

2.2 Threat model

We follow standard threat model assumptions in side-channel analysis, model training, and deployment (3; 40; 67; 32; 23; 13; 46). This assumes a black-box setting in which an adversary has physical access to the target device during inference but lacks knowledge of its internal microarchitecture, compiler, or instruction set. The attacker knows the model's hyperparameters and seeks to extract the proprietary model parameters. By providing chosen inputs of known dimensions, the adversary can capture physical emanations, such as EM or power signals, generated during execution. Our defense applies to the trusted training phase, which is performed once for a model for any target device before deployment. The defender can modify the training loss function. The objective of the defender is to make the attack harder such that model extraction requires more measurements than without defense. Other physical attack vectors, such as fault (5; 44; 7), micro-architectural (33; 52) and cold boot attacks (62), are out of scope and can be protected with orthogonal defenses (1; 35; 55; 30).

3 Proposed Defense

Our defense is the first to counter side-channel based model parameter extraction attacks on Edge TPU without requiring modifications to the hardware or firmware stack. We achieve improved side-channel resistance by shuffling different parameter versions of similar task accuracy across inferences. We propose a novel training strategy that generates multiple model versions that maintains near-baseline accuracy while obfuscating the model parameters. In conventional training, a single weight matrix is initialized for each layer and iteratively optimized to minimize the task-specific loss, such as Cross-Entropy. During each epoch, the dataset is partitioned into mini-batches, and gradients are computed via backpropagation to update the entire parameter set. This creates a static model whose repeated parameters produce exploitable side-channel leakage.

3.1 Proposed Diversity-Enforced Stochastic Training

Algorithm 1 presents the proposed training methodology designed to produce a side-channel resilient model by enforcing parameter diversity. Unlike traditional training, which optimizes a single set of

Algorithm 1 Proposed Diversity-Enforced Stochastic Training

'Input: Dataset $\mathcal{D}$, Stacked weights $W_l^{(1)\dots(V)}$, coefficient λ_{div}, learning rate η
Initialize V versions for each layer: $\{W_l^{(1)}, \dots, W_l^{(V)}\}$
for each epoch **do**
  Shuffle $\mathcal{D}$ and partition into mini-batches $\mathcal{B}$
  for each mini-batch $(x, y) \in \mathcal{B}$ **do**
    for each layer $l \in \{1, \dots, L\}$ **do**
      $r_l \sim \text{Uniform}(1, V)$ {Stochastic Path Selection}
    end for
    $\hat{y} \leftarrow \text{ForwardPass}(x, \{W_1^{(r_1)}, \dots, W_L^{(r_L)}\})$
    $\mathcal{L}_{task} \leftarrow \text{CrossEntropy}(y, \hat{y})$
    $\mathcal{L}_{div} \leftarrow \sum_{l=1}^{L} \sum_{p<q} \text{Similarity}(W_l^{(p)}, W_l^{(q)})$
    $\mathcal{L}_{total} \leftarrow \mathcal{L}_{task} - \lambda_{div} \cdot \mathcal{L}_{div}$
    $\nabla W_l^{(i)} \leftarrow \begin{cases} \text{Backpropagate}(\mathcal{L}_{total}) & i = r_l \\ 0 & i \neq r_l \end{cases} \quad \forall l$
    $W_l^{(r_l)} \leftarrow W_l^{(r_l)} - \eta \cdot \nabla W_l^{(r_l)} \quad \forall l$ {Update active path only}
  end for
end for
return $W_l^{(1)\dots(V)}$

weights, this algorithm maintains V distinct versions per layer and uses stochasticity to ensure they are both accurate and dissimilar. The algorithm begins by creating a stacked architecture. For every layer l in an $L-$layer network, the defender initializes V independent versions of the weight matrices $W_l^{(1)}, \dots, W_l^{(V)}$. This expands the parameter space, as any version of layer l must eventually be compatible with any version of layer $l+1$. For every mini-batch of data, the algorithm performs path stochasticity. It samples a random index r_l for each layer from a uniform distribution. This defines a unique *active path* through the stacked model. Because this selection happens at every iteration, the model learns to be *composable*—the final accuracy does not depend on a specific combination of weights, but rather on the collective functionality of all versions. The loss function is defined by:

- Task Loss ($\mathcal{L}_{task}$): Cross-Entropy loss enforces task accuracy for the selected path.
- Diversity Loss ($\mathcal{L}_{div}$): This term enforces structural diversity among the parameter versions by calculating the pairwise similarity (cosine similarity) between all V versions within each layer. Because the objective is to maximize dissimilarity, this metric is subtracted from the overall loss. We use cosine similarity as the metric as it is bounded compared to Euclidean distance and therefore results in faster loss convergence.
- Total Loss ($\mathcal{L}_{total}$): The terms are combined using a diversity coefficient λ_{div}. This forces the optimizer to maintain accuracy while pushing weight versions apart in the parameter space.

After backpropagating the total loss, the algorithm applies a gradient mask. This is a critical step for maintaining the independence of the versions. The algorithm iterates through every layer and version. If a version was not part of the randomly selected active path ($i \neq r_l$), its gradient is zeroed out. As a result, only the weights that actually participated in the forward pass are updated for that specific mini-batch. Finally, the weights are updated using the masked gradients. Over many epochs, this selective update strategy ensures that all V versions of each layer converge to task-accurate states. However, because they are constantly being pushed away from each other by the $\mathcal{L}_{div}$ term and are updated on different subsets of data due to the path stochasticity, they remain distinct.

3.2 Implementation of random parameter selection

During inference, parameters are dynamically selected for each layer from the set of available versions. Unlike traditional processors, the target TPU (26) does not support conditional logic (such as if/else statements) or custom instructions. This limitation makes all prior countermeasures incompatible. Our work addresses this gap by providing the first defense designed to secure parameters within these strict constraints. For the Google Edge TPU, models must be compiled by a command-line tool `edgetpu-compiler` before deployment (27). This imposes several requirements: the parameters must be 8-bit quantized, have a constant size at compile-time and can only use native operations (25). Inference on the Google Edge TPU uses the LiteRT libraries.

To adhere to these hardware constraints, we implement weight stacking, where all V versions of a layer's parameters are concatenated into a single tensor. At inference time, the hardware generates a one-hot encoded vector representing the randomly selected index r. By performing a dot product

between this one-hot vector and the stacked weight tensor, the hardware effectively masks all inactive versions, extracting only the desired parameters for the computation. This approach ensures the model remains compatible with the Edge TPU's requirement for constant-sized tensors and fixed operation sets, while still enabling the dynamic parameter shuffling necessary for side-channel resilience. For example, consider a layer with three weight versions X, Y, Z; these are first concatenated into a single stacked tensor, $W_{stacked} = [X, Y, Z]$. To select a specific version, the system generates a corresponding one-hot encoded vector, R_{one_hot}. If the second version ($r = 1$) is selected, the vector becomes $[0, 1, 0]$. The hardware then computes the output via a *reduce_sum* of the element-wise product: $W_{stacked} \cdot R_{one_hot} = (X \cdot 0) + (Y \cdot 1) + (Z \cdot 0)$, which simplifies to Y. This approach ensures that the active parameter is extracted using only multiplication and addition, allowing the defense to execute on hardware that lacks native support for conditional instructions.

However, a challenge arises when applying weight stacking to convolutional layers. The `edgetpu-compiler` requires the weights for TensorFlow convolution ($tf.conv$) to be constant at compile-time; however, a dynamic value is produced by the one-hot multiplication, resulting in a compilation error. To resolve this, we implement an output accumulation strategy. Rather than selecting the weights before the convolution, we compute the convolutions for all V versions independently in random order and accumulate the results using the one-hot vector R_{one_hot}. For example, with three versions ordered as $\{X, Y, Z\}$ and $r = 1$, the final output is $\sum_{i=1}^{N}(tf.conv(W^{(i)}) \cdot r_i) = tf.conv(X) \cdot 0 + tf.conv(Y) \cdot 1 + tf.conv(Z) \cdot 0$. This ensures that $tf.conv$ operations utilize static weights, while compute sequence and final output are randomized.

4 Theoretical Analysis of Side-Channel Leakage Reduction

To evaluate leakage, we employ the TVLA methodology, a commonly used metric for side-channel analysis (54; 32). TVLA relies on Welch's t-test to determine whether side-channel traces (e.g., power or EM measurements) collected under two different input conditions are statistically distinguishable. If the statistical test concludes that the two leakage distributions are indistinguishable, the device is considered secure against first-order attacks. We utilize the *fixed versus random* test methodology using the hamming weight leakage model. The fixed set contains measurements collected while the model processes a fixed input, while the random set contains measurements collected while processing uniformly random inputs. The distinguishability between these two sets is calculated using the t-statistic:

$$t = (\mu_f - \mu_r) / \sqrt{\sigma_f^2/N_f + \sigma_r^2/N_r} \tag{1}$$

where μ, σ^2, and N denote mean leakage, variance, and trace count, with subscripts f and r indicating the fixed and random sets. In a baseline model, static weights are used for both sets. The only randomness in this case comes from the random inputs. This lack of randomness makes the two sets distinguishable, resulting in a higher t-score and indicating that the design leaks information. The randomness introduced by our defense in the weight selection per execution increases the variances of both sets (σ_f^2 and σ_r^2), thereby reducing the distinguishability between the sets. We theoretically show the increase in variance with our defense, while the means of the sets remain constant, in Section 4.1.

The remaining section summarizes the theoretical foundation of our defense. We answer three questions: (i) why alternating between dissimilar parameter versions reduces side-channel leakage, (ii) how this reduction scales with an arbitrary number of versions V, and (iii) how a designer can predict the leakage of a new configuration without re-running the full analysis. The complete proofs are provided in Appendix A.1 and A.2.

4.1 Leakage Reduction from Parameter Dissimilarity

We begin with the simplest case: a layer that alternates between two functionally equivalent weight versions, W_a and W_b, selected with equal probability at each inference. The goal is to establish how the Euclidean distance $\|\Delta W\|$ separating the two versions influences the TVLA t-score. We analyze the leakage under first-order approximation model as $L = \mathcal{L}(W \cdot x) + \epsilon$, where $\mathcal{L}$ is the data-dependent emission and ϵ is independent measurement noise. The hardware alternates between W_a and W_b during both the fixed-set (constant input x_f) and random-set (random input x_r) acquisitions of the Welch's t-test. Each component of the t-statistic is derived separately as a function of W_a and W_b, and then re-expressed in terms of the centroid $\bar{W} = (W_a + W_b)/2$ and the divergence $\Delta W = W_a - W_b$. This yields two key conclusions: **(1) Numerator (mean difference) is constant:** The fixed-set mean μ_f and random-set mean μ_r both reduce to expressions centered on $\bar{W}$, with the contributions of $+\Delta W/2$ and $-\Delta W/2$ canceling by symmetry. Their difference $\mu_f - \mu_r$ therefore collapses to a baseline signal K that is independent of the divergence between versions. **(2) Denominator**

(variance) grows quadratically with $\|\Delta W\|$: The fixed-set variance σ_f^2 is driven by the squared leakage gap $(L_a - L_b)^2$ between the two weight versions. The random-set variance σ_r^2 decomposes via the Law of Total Variance into three parts: a within-weight term (variation from random inputs), a between-weight term (variation from alternating weight versions), and the measurement noise. The between-weight term mirrors the fixed-set structure, while the within-weight term contributes a constant input-noise baseline. Both version-dependent contributions scale with $\|\Delta W\|^2$. Substituting these components into the Welch's t-statistic produces the relationship:

$$t \;\approx\; K\big/\sqrt{C\,\|\Delta W\|^2 + C_0} \tag{2}$$

where C aggregates the structural (version-induced) variance and C_0 aggregates the input and measurement noise. Equation 2 establishes the foundational mechanism of the defense: increasing the Euclidean distance between the alternating parameters inflates the TVLA denominator while the numerator is constant, thereby suppressing the leakage detectability.

4.2 Generalization to V Parameter Versions

The same reasoning extends to an ensemble of $V \geq 2$ versions selected uniformly at random. The parameter set is now characterized by its centroid $\bar{W} = \frac{1}{V}\sum_j W_j$, its minimum pairwise Euclidean distance $d_{\min}$, and its average squared pairwise distance $\bar{D}^2$. Applying the same term-by-term strategy, now aggregated across V uniformly-weighted states, the conclusions are:

- The numerator $\mu_f - \mu_r$ remains a constant baseline signal K, *independent of both V and the pairwise distances*. The symmetric cancellation that held for two versions carries over through the zero-sum property $\sum_j (W_j - \bar{W}) = 0$.
- The denominator aggregates the variance across all V versions via pairwise identity, introducing the ensemble-scaling factor $(V - 1/V)$. The structural variance is proportional to $(V - 1/V) \cdot \bar{D}^2$, so adding versions tightens the ensemble's contribution toward a saturating limit as $V \to \infty$.

Taking the worst-case substitution $\bar{D}^2 \geq d_{\min}^2$, assuming balanced acquisitions ($N_f = N_r = N/2$), and grouping constants produces the master bound:

$$\boxed{t(V, d_{\min}) \;\leq\; K\big/\sqrt{A \cdot (V - 1/V) \cdot d_{\min}^2 \;+\; B}} \tag{3}$$

where K captures the baseline signal, A the per-trace structural variance, and B the per-trace aggregate noise (measurement and input). All three constants are independent of V and $d_{\min}$. For $V = 2$, the factor $(V - 1/V) = 1/2$, and Equation 3 algebraically reduces to Equation 2, confirming consistency with the two-version result in Section 4.1.

4.3 Empirical Calibration for Predictive Use

Equation 3 is predictive, but the constants A, B, and K depend on hardware-specific noise profiles that are impractical to compute analytically. To make the bound operational, we derive a one-time two-point calibration that eliminates them using measured t-scores. Let t_1 be the t-score of the unprotected model ($V = 1$), and t_2 the t-score of a two-version profiling run at a known separation d. Substituting these two measurements into Equation 3 cancels the absolute scale constants and leaves:

$$\boxed{t(V, d_{\min}) \;\leq\; t_1\big/\sqrt{\frac{V-1}{V} \cdot \left(\frac{A}{B}\right) \cdot d_{\min}^2 + 1}\;, \qquad \frac{A}{B} \;=\; \frac{2}{d^2}\left[\left(\frac{t_1}{t_2}\right)^2 - 1\right]} \tag{4}$$

This reduces security prediction to two profiling measurements. Given t_1, t_2, and d for a target platform, Equation 4 bounds the t-score of any future configuration $(V, d_{\min})$, enabling a designer to predict the t-score for an ensemble size and separation without hardware experiments.

5 Empirical Evaluation and Results

Our empirical evaluation examines the leakage reduction, robustness, and deployment cost of parameter shuffling on the target Edge TPU. We first describe the measurement setup and model configurations used across the evaluation. We then evaluate side-channel leakage reduction, presenting representative results demonstrating that measured leakage remains below the standard TVLA threshold of $|t| = 4.5$ (54; 22). We next evaluate whether a defense-aware attacker can recover the hidden parameter-version selector through unsupervised clustering. Finally, we quantify the accuracy, training-time, inference-time, and model-size trade-offs of the defense.

5.1 Experimental Setup

We summarize the experimental setup here; Appendix A.4.1 provides measurement, acquisition, and alignment details, including the full setup in Figure 4. We perform all side-channel measurements on a Google Coral Dev Board featuring an Edge TPU System-on-Module (SoM) running Mendel Linux (25). To measure localized EM side-channel leakage, we use Keysight's EM measurement setup comprising a high-sensitivity EM probe, a probe station, and a motorized XYZ table (38). We capture EM traces using a PicoScope 6000E oscilloscope at 1.25 GS/s (51). After removing the SoM cooling fan, we scan the exposed package for peak TPU leakage and fix the probe there for all acquisitions (Figure 5). We collect traces using fixed-vs-random TVLA[1] (54). We apply piecewise-static alignment across layer-specific regions to address the timing variation introduced by OS scheduling, memory activity, and interleaved CPU/TPU operations. We identify these regions using prior hyperparameter-extraction heuristics (41), then generate predicted-operation templates from single-layer sub-models and match them against runtime measurement profiles. After alignment, we compute the TVLA t-statistic on the aligned layer-specific regions.

5.2 Pushing Side-Channel Leakage Below TVLA Threshold

We train all protected models using the stochastic layer-wise procedure from Section 3, which samples one parameter version per protected layer per mini-batch and enforces inter-version diversity. We consider three representative Edge TPU workloads that were primarily targeted in prior attacks (24; 32; 65): an MNIST (42) multilayer perceptron (MLP) with architecture 784–100–10, a CIFAR-10 (39) convolutional neural network (CNN) with four 3×3 convolutional layers containing 32, 32, 64, and 64 filters, 2×2 max-pooling, and two dense layers with 256 and 10 neurons, and EfficientNet-EdgeTPU-S (60; 28) evaluated on ImageNet-1K (15; 53). For the MNIST MLP and CIFAR-10 CNN, we apply the defense to all trainable layers, while for EfficientNet-EdgeTPU-S we apply our defense to the first two vulnerable layers attacked in prior work (32). Appendix A.4.2 provides the complete training and evaluation details.

Our theoretical model captures both the number of parameter versions V and their separation. Increasing V can increase version-induced variance, while stronger diversity regularization λ_{div} can increase pairwise separation. We protect the CIFAR-10 CNN using $\lambda_{\text{div}} = 100$, with $V = 2$ for the CONV layers and $V = 8$ for the DENSE layers. Figure 2 shows that, using 20K total traces[2], all samples in the protected `DENSE_2` region remain below standard TVLA leakage threshold of $|t| = 4.5$, compared with a peak t-statistic of 154 for the unprotected model. Under the evaluated trace budget, TVLA does not detect leakage in `DENSE_2`, providing evidence that parameter shuffling can suppress side-channel leakage. In practice, the achievable leakage suppression depends on the protection configuration, layer, and trace budget. We therefore evaluate how well the model predicts TVLA across configurations and how the t-statistic evolves with trace count to guide the choice of V and λ_{div}. Appendix A.5 presents these analyses in detail.

5.3 Feasibility of Defense-Aware Adaptive Clustering Attack

We analyze whether the hidden parameter-version selector values leave separable signatures in the measured EM side-channel traces. If such signatures exist, an adaptive attacker could group traces that use the same parameter version and analyze each group separately for parameter extraction, weakening the protection provided by dynamic parameter shuffling. Since parameter-extraction attacks are typically performed layer-wise (32), we focus on `DENSE_1` of the MNIST MLP as a representative case. We examine trace separability using principal component analysis (PCA) (37) and project the attacker-visible trace features onto PC1–PC2. Appendix A.6 provides the complete attacker assumptions, feature-selection procedure, and clustering techniques.

The trace groups corresponding to different parameter versions fail to form distinct clusters in the measured trace space. Figure 3 shows overlap among traces generated using different parameter versions in the PC1–PC2 projections for two example settings with $V = 3$ and $V = 8$, with no visually distinct groups. Consistent with this overlap, K-means (48) clustering (see Appendix A.6) achieves best-mapped accuracies of 34.62% and 13.43%, close to the corresponding random chance levels of 33.3% and 12.5%, with adjusted Rand index (ARI) values near zero. Thus, under the evaluated PC1–PC2 attack, K-means does not reliably group traces by the parameter version used during inference, limiting the ability to undo the randomization before parameter extraction.

[1] Trace counts denote the total number of traces, split equally between the fixed and random TVLA sets.

[2] 20K traces were regarded as sufficient for "long" executions like public key encryption (12)

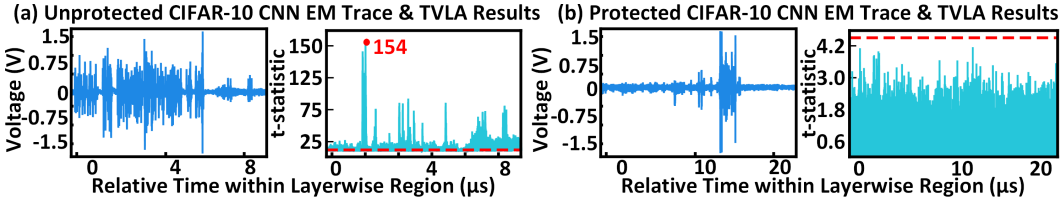


Figure 2: EM traces and fixed-vs-random TVLA results for the DENSE_2 layer of the CIFAR-10 CNN using 20K traces. (a) Unprotected model with peak t-score of 154 (b) protected model using $V = 8$ and $\lambda_{\text{div}} = 100$. Our defense suppresses the leakage below the TVLA threshold of 4.5.

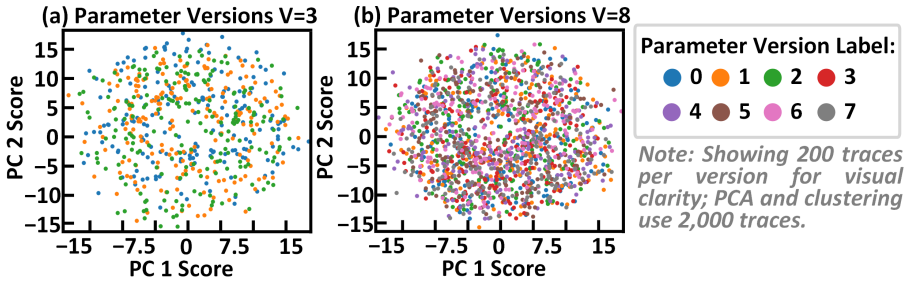


Figure 3: PC1–PC2 projections for the protected MNIST MLP with $V = 3$ and $V = 8$. Traces from different parameter versions strongly overlap in both configurations, limiting an attacker's ability to group traces by version and undo the randomization before parameter extraction.

5.4 Balancing Leakage Reduction and Computational Cost

We next examine the computational cost of our defense alongside its security benefit, quantified by the additional traces required for the protected model to reach the same TVLA t-statistic as the unprotected model. Table 1 shows that all protected models preserve task accuracy within 1.3 percentage points of their unprotected models. For MNIST MLP, the one-time training overhead ranges from $1.85\times$ to $30.10\times$, while inference and model-size overheads remain within $3.43\times$–$5.57\times$ and $2.29\times$–$4.33\times$ respectively. Importantly, the higher overhead at larger V is accompanied by increased security benefit: across the MNIST configurations, the protected model requires $2.23\times$–$42.21\times$ more traces to reach the same TVLA t-statistic as the unprotected model. The smaller security improvement is observed in the first layer due to input dependent leakage consistent with prior works (18) (see Appendix A.5). The CIFAR-10 and EfficientNet-EdgeTPU-S models show a similar cost–security balance, with deployment overhead below $3.2\times$ for a security improvement of upto $48.04\times$ and $15.72\times$, respectively, for the evaluated configurations. These results demonstrate the trade-off between computational cost and security benefit, which designers can balance against desired leakage suppression and available deployment resources.

Table 1: Accuracy, computational overhead, and security improvement of the proposed defense

Model	Dataset	Variant	Accuracy (%)	Training Time (s)	Inference Time (ms)	Model Size (MB)	Security Improvement
MLP	MNIST	U	96.6	4.03	2.197	0.485	–
		P ($\lambda = 0.3$, $V = 2$)	97.2 (+0.6 pp)	7.46 (1.85×)	7.53 (3.43×)	1.11 (2.29×)	2.23 – 3.81×
		P ($\lambda = 0.3$, $V = 3$)	96.09 (−0.51 pp)	18.25 (4.53×)	7.83 (3.56×)	1.18 (2.43×)	5.06 – 9.37×
		P ($\lambda = 0.3$, $V = 4$)	95.3 (−1.3 pp)	32.3 (8.01×)	8.03 (3.65×)	1.26 (2.60×)	4.83 – 7.80×
		P ($\lambda = 0.3$, $V = 5$)	95.7 (−0.9 pp)	51.9 (12.88×)	10.84 (4.93×)	1.95 (4.02×)	6.36 – 15.39×
		P ($\lambda = 0.3$, $V = 6$)	96.03 (−0.57 pp)	75.4 (18.71×)	12.24 (5.57×)	2.05 (4.23×)	6.17 – 20.43×
		P ($\lambda = 0.3$, $V = 7$)	95.7 (−0.9 pp)	98.2 (24.37×)	10.78 (4.91×)	2.09 (4.31×)	6.17 – 7.05×
		P ($\lambda = 0.3$, $V = 8$)	96.1 (−0.5 pp)	121.3 (30.10×)	11.30 (5.14×)	2.10 (4.33×)	7.28 – 42.21×
CNN	CIFAR-10	U	72.4	2.407	2.22	1.13	–
		P ($\lambda = 60$, $V = 2$)	71.9 (−0.5 pp)	5.76 (2.39×)	5.72 (2.58×)	2.33 (2.06×)	1.48 – 48.04×
EfficientNet -EdgeTPU-S	ImageNet-1K	U	87.4	42.3	19.521	5.46	–
		P ($\lambda = 0.3$, $V = 3$)	87.1 (−0.3 pp)	114.7 (2.7×)	62.01 (3.17×)	5.69 (1.04×)	4.16 – 15.72×

U: Unprotected model; P: Protected model. Inference time is averaged over 100 runs; parentheses show accuracy change in percentage points and overhead relative to the corresponding unprotected model. Security improvement measures the factor increase in traces needed for the protected model to reach the same TVLA t-score relative to the unprotected model.

6 DISCUSSION

Generalization to other platforms and architectures: Our defense can be retrofitted to other platforms, as the core methodology relies on stochastic training performed offline before deployment. While our implementation addresses the constraints of the Google Edge TPU, the associated overhead could be reduced on platforms with native support for conditional constructs, enabling more efficient parameter swapping during inference. While our empirical evaluation covers MLPs and CNNs, the training-based defense is architecture-agnostic and not fundamentally limited to these families. We did not evaluate LLMs in this work because LLMs have not been attacked in prior work. Nevertheless, our defense could, in principle, extend to LLMs as these attacks emerge.

Choosing the number of versions and parameter dissimilarity: The selection of the number of parameter versions and the diversity strength λ_{div} involves balancing the overhead with the task accuracy to achieve side-channel leakage below 4.5. In practice, the defender can first determine the optimum number of versions V supported by the hardware's memory and latency budget. By substituting this value and the target t-score into Equation 4, the required minimum Euclidean distance d_{min} can be derived. Finally, λ_{div} is tuned via a parameter sweep to obtain the target d_{min}. Careful selection of V and λ enables the proposed defense to keep leakage below 4.5 (Section 5.2). Achieving this in every model configuration requires tuning of V and λ, but the defense reliably reduces leakage across all tested configurations.

Limitations: Security improvement is not free, but inference overhead scales linearly with the number of parameter versions across training time, inference latency, and model size, while attack hardness increases by $48.04\times$ at high diversity setting. To manage overhead in practice, the defense can be selectively applied to the most vulnerable layers rather than the full network, as shown on EfficientNet, whose first two layers are attacked in prior work.

Leakage evaluation using TVLA: TVLA stands as the primary evaluation methodology for end-to-end side-channel leakage analysis, as mandated in certification frameworks like FIPS 140-3, ISO/IEC 17825, and NIST specifications. While correlation power analysis based attacks test only specific parameter extractions (e.g., single weight recovery under a single leakage model), TVLA is the de-facto evaluation used to expose comprehensive leakage. Following standard practice across the literature of attacks on AI/ML, we rely on first-order TVLA as the sole leakage evaluation metric (29; 2; 20; 6; 45). Higher-order TVLA is designed to evaluate masking-based countermeasures, where secrets are split into multiple shares and leakage must be assessed across higher statistical moments or joint time samples e.g., second-order attacks on first-order masking (54; 22; 58; 11; 12). In contrast, our defense does not employ secret-sharing; rather, it stochastically selects complete, functionally equivalent parameter values across measurements. Because there are no secret shares to combine, higher-order TVLA is inapplicable to our defense technique. Additionally, our analysis is conducted in a high-sensitivity setup where we show a reduction from over 500, highlighting that prior works under-report TVLA leakage by relying on lower-sensitivity setups with the unprotected baseline scores less than 50 (32; 18; 17; 47; 29).

Other defenses: Our proposed technique is the **only** approach that can be retrofitted to Edge TPU. Alternative countermeasures such as masking (17; 18; 19), garbled circuits (29), and physical-layer defenses (64; 68) report overheads from $3\times$ to $50\times$ in terms of resource utilization and latency and more importantly, cannot be deployed on this device at all, since they require hardware, firmware, or compiler modifications. Our contribution is a defense that can be readily deployed on Edge TPUs.

7 CONCLUSION

No TPU should be left behind vulnerable against side-channel attacks, including those that are deployed today. Existing defenses change hardware, firmware, or compilers, so they cannot retrofit fixed-function accelerators such as the Edge TPU. We present the first retrofit defense for the TPU. Our training method creates multiple functionally equivalent parameter versions and swaps them at inference time. Our proposed diversity regularizer keeps these versions distinct, breaking repeated leakage patterns. Our theoretical framework predicts leakage before deployment. Experiments across models and datasets show TVLA reduction, with leakage falling below 4.5 in a high-diversity configuration and less than 1% accuracy change in most settings.

AI USE STATEMENT

In this work, we used generative AI tools for writing assistance, specifically to check grammar and improve prose. We did not use generative AI for other tasks, such as generating research ideas or writing theoretical proofs.

ETHICS STATEMENT

This paper does not include studies that involve human subjects, practices to data set releases, potentially harmful insights, methodologies and applications, potential conflicts of interest and sponsorship, discrimination/bias/fairness concerns, privacy and security issues, legal compliance, and research integrity issues.

REPRODUCIBILITY STATEMENT

We propose a novel training methodology (Algorithm 1) to produce a side-channel-resistant neural network by enforcing parameter diversity. We provide a theoretical framework that estimates side-channel leakage for a given target, version count and parameter distance in Section 4 and provide complete proof and assumption in Appendix A.1 and A.2. We provide our training, inference code and models in the GitHub repo: `https://anonymous.4open.science/r/No-TPU-Left-Behind/`

REFERENCES

[1] Kasra Ahmadi, Saeed Aghapour, Mehran Mozaffari Kermani, and Reza Azarderakhsh. MAED: Mathematical Activation Error Detection for Mitigating Physical Fault Attacks in DNN Inference, 2026.

[2] Konstantinos Athanasiou, Thomas Wahl, A Adam Ding, and Yunsi Fei. Masking Feedforward Neural Networks Against Power Analysis Attacks. *proceedings on privacy enhancing technologies*, 2022.

[3] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 515–532, 2019.

[4] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[5] Jakub Breier, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. DeepLaser: Practical Fault Attack on Deep Neural Networks, 2018.

[6] Jakub Breier, Dirmanto Jap, Xiaolu Hou, and Shivam Bhasin. A Desynchronization-Based Countermeasure Against Side-Channel Analysis of Neural Networks. In *International Symposium on Cyber Security, Cryptology, and Machine Learning*, pages 296–306. Springer, 2023.

[7] Jakub Breier, Dirmanto Jap, Xiaolu Hou, Shivam Bhasin, and Yang Liu. SNIFF: Reverse Engineering of Neural Networks With Fault Attacks. *IEEE Transactions on Reliability*, 71(4):1527–1539, 2022.

[8] Manuel Brosch, Matthias Probst, Matthias Glaser, and Georg Sigl. A Masked Hardware Accelerator for Feed-Forward Neural Networks With Fixed-Point Arithmetic. *IEEE Trans. Very Large Scale Integr. Syst.*, 32(2):231–244, February 2024.

[9] Manuel Brosch, Matthias Probst, and Georg Sigl. Counteract Side-Channel Analysis of Neural Networks by Shuffling. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1305–1310, 2022.

[10] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic Extraction of Neural Network Models. In *Annual international cryptology conference*, pages 189–218. Springer, 2020.

[11] Yajing Chang, Yingjian Yan, Chunsheng Zhu, and Yanjiang Liu. A High-performance Masking Design Approach for Saber against High-order Side-channel Attack. *ACM Transactions on Design Automation of Electronic Systems*, 28(6):1–19, 2023.

[12] Keng-Yu Chen and Jiun-Peng Chen. Masking Floating-Point Number Multiplication and Addition of Falcon. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(2):276–303, 2024.

[13] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[14] Yi Chen, Xiaoyang Dong, Jian Guo, Yantian Shen, Anyu Wang, and Xiaoyun Wang. Hard-Label Cryptanalytic Extraction of Neural Network Models. Cryptology ePrint Archive, Paper 2024/1403, 2024.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[16] Ruyi Ding, Cheng Gongye, Davis Ranney, Aidong Adam Ding, and Yunsi Fei. MACPruning: Dynamic Operation Pruning to Mitigate Side-Channel DNN Model Extraction. In *2025 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 12–22, 2025.

[17] Anuj Dubey, Afzal Ahmad, Muhammad Adeel Pasha, Rosario Cammarota, and Aydin Aysu. ModuloNET: Neural Networks Meet Modular Arithmetic for Efficient Hardware Masking. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 506–556, 2022.

[18] Anuj Dubey, Rosario Cammarota, and Aydin Aysu. BoMaNet: Boolean masking of an entire neural network. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–9, 2020.

[19] Anuj Dubey, Rosario Cammarota, and Aydin Aysu. MaskedNet: The First Hardware Inference Engine Aiming Power Side-Channel Protection. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 197–208. IEEE, 2020.

[20] Anuj Dubey, Rosario Cammarota, Vikram Suresh, and Aydin Aysu. Guarding machine learning hardware against physical side-channel attacks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 18(3):1–31, 2022.

[21] Karthik Ganesan, Michal Fishkin, Ourong Lin, and Natalie Enright Jerger. BlackJack: Secure machine learning on IoT devices through hardware-based shuffling, 2023.

[22] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136, 2011.

[23] Cheng Gongye, Yunsi Fei, and Thomas Wahl. Reverse-Engineering Deep Neural Networks Using Floating-Point Timing Side-Channels. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[24] Cheng Gongye, Yukui Luo, Xiaolin Xu, and Yunsi Fei. Side-Channel-Assisted Reverse-Engineering of Encrypted DNN Hardware Accelerator IP and Attack Surface Exploration. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2023.

[25] Google. Coral Dev board SoM datasheet. `https://www.coral.ai/docs/dev-board/datasheet/`. Accessed: 2026-09-24.

[26] Google. Edge TPU. `https://www.coral.ai/docs/dev-board/get-started/`. Accessed: 2026-09-24.

[27] Google. Edge tpu compiler. `https://www.coral.ai/docs/edgetpu/compiler/#system-requirements`. Accessed: 2026-09-24.

[28] Google Research. EfficientNet-EdgeTPU: Creating Accelerator-Optimized Neural Networks with AutoML. `https://research.google/blog/efficientnet-edgetpu-creating-accelerator-optimized-neural-networks-with-automl/`. Accessed: 2026-09-24.

[29] Mohammad Hashemi, Steffi Roy, Domenic Forte, and Fatemeh Ganji. HWGN2: Side-channel Protected Neural Networks through Secure and Private Function Evaluation, 2022.

[30] MD Mahady Hassan, Shanto Roy, and Reza Rahaeimehr. Memory Under Siege: A Comprehensive Survey of Side-Channel Attacks on Memory, 2025.

[31] Sihan He, Weibin Wu, Yanbin Li, Lu Zhou, Liming Fang, and Zhe Liu. Recovering the Weights of Convolutional Neural Network via Chosen Pixel Horizontal Power Analysis. In *Wireless Algorithms, Systems, and Applications: 17th International Conference, WASA 2022, Dalian, China, November 24–26, 2022, Proceedings, Part II*, page 93–104, Berlin, Heidelberg, 2022. Springer-Verlag.

[32] Peter Horvath, Lukasz Chmielewski, Léo Weissbart, Lejla Batina, and Yuval Yarom. BarraCUDA: Edge GPUs do leak DNN weights. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 4017–4034, 2025.

[33] Weizhe Hua, Zhiru Zhang, and G. Edward Suh. Reverse Engineering Convolutional Neural Networks Through Side-channel Information Leaks. In *Proceedings of the 55th Annual Design Automation Conference*, DAC '18, New York, NY, USA, 2018. Association for Computing Machinery.

[34] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High Accuracy and High Fidelity Extraction of Neural Networks. In *29th USENIX security symposium (USENIX Security 20)*, pages 1345–1362, 2020.

[35] Mojan Javaheripi, Jung-Woo Chang, and Farinaz Koushanfar. AccHashtag: Accelerated Hashing for Detecting Fault-Injection Attacks on Embedded Neural Networks. *J. Emerg. Technol. Comput. Syst.*, 19(1), December 2022.

[36] Xin Jin, Charalampos Katsis, Fan Sang, Jiahao Sun, Ashish Kundu, and Ramana Kompella. Edge security: Challenges and issues, 2022.

[37] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 04 2016.

[38] Keysight Technologies. EM Probe Station 5 Datasheet. `https://www.keysight.com/us/en/assets/3124-1811/data-sheets/EM-Probe-Station-5.pdf`. Accessed: 2026-09-24.

[39] Alex Krizhevsky and Geoffrey Hinton. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009.

[40] Ashley Kurian and Aydin Aysu. Train to Defend: First Defense Against Cryptanalytic Neural Network Parameter Extraction Attacks. *Advances in Neural Information Processing Systems*, 38:23926–23946, 2026.

[41] Ashley Kurian, Anuj Dubey, Ferhat Yaman, and Aydin Aysu. TPUXtract: An Exhaustive Hyperparameter Extraction Framework. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(1):78–103, 2025.

[42] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[43] Ge Li, Mohit Tiwari, and Michael Orshansky. Power-based attacks on spatial dnn accelerators. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 18(3):1–18, 2022.

[44] Shaofeng Li, Xinyu Wang, Minhui Xue, Haojin Zhu, Zhi Zhang, Yansong Gao, Wen Wu, and Xuemin (Sherman) Shen. Yes, One-Bit-Flip Matters! Universal DNN Model Inference Depletion with Runtime Code Fault Injection. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1315–1330, Philadelphia, PA, August 2024. USENIX Association.

[45] Zhuoran Liu, Senna van Hoek, Péter Horváth, Dirk Lauret, Xiaoyun Xu, and Lejla Batina. Real-world edge neural network implementations leak private interactions through physical side channel. *arXiv preprint arXiv:2501.14512*, 2025.

[46] Henrique Teles Maia, Chang Xiao, Dingzeyu Li, Eitan Grinspun, and Changxi Zheng. Can one hear the shape of a neural network?: Snooping the GPU via Magnetic Side Channel, 2021.

[47] Saurav Maji, Utsav Banerjee, Samuel Fuller, and Anantha Chandrakasan. A Threshold Implementation-Based Neural Network Accelerator With Power and Electromagnetic Side-Channel Countermeasures. *IEEE Journal of Solid-State Circuits*, PP:1–14, 01 2022.

[48] James B McQueen. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th berkeley symposium on math. stat. and prob.*, pages 281–297, 1967.

[49] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff Nets: Stealing Functionality of Black-Box Models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4954–4963, 2019.

[50] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[51] Pico Technology. PicoScope 6000E Series 3 GHz 5 GS/s Deep Memory MSO USB Oscilloscope. `https://www.picotech.com/oscilloscope/picoscope-6000e-series-3-ghz-5gss-deep-memory-mso-usb-oscilloscope`. Accessed: 2026-09-24.

[52] Adnan Siraj Rakin, Md Hafizul Islam Chowdhuryy, Fan Yao, and Deliang Fan. DeepSteal: Advanced Model Extractions Leveraging Efficient Weight Stealing in Memories, 2021.

[53] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[54] Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, Lecture Notes in Computer Science, pages 495–513. Springer, 2015.

[55] Hoseok Seol, Minhye Kim, Taesoo Kim, Yongdae Kim, and {Lee Sup} Kim. Amnesiac DRAM: A Proactive Defense Mechanism against Cold Boot Attacks. *IEEE Transactions on Computers*, 70(4):539–551, April 2021. Publisher Copyright: © 1968-2012 IEEE.

[56] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[57] Shubhi Shukla, Manaar Alam, Sarani Bhattacharya, Pabitra Mitra, and Debdeep Mukhopadhyay. "Whispering MLaaS": Exploiting Timing Channels to Compromise User Privacy in Deep Neural Networks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 587–613, 2023.

[58] François-Xavier Standaert. How (not) to use welch's t-test in side-channel security evaluations. In *International conference on smart card research and advanced applications*, pages 65–79. Springer, 2018.

[59] Weixing Su, Linfeng Li, Fang Liu, Maowei He, and Xiaodan Liang. AI on the edge: a comprehensive review. *Artificial Intelligence Review*, 55(8):6125, 2022.

[60] Mingxing Tan and Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International conference on machine learning*, pages 6105–6114. PmLR, 2019.

[61] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618, 2016.

[62] Yoo-Seung Won, Soham Chatterjee, Dirmanto Jap, Arindam Basu, and Shivam Bhasin. DeepFreeze: Cold Boot Attacks and High Fidelity Model Recovery on Commercial EdgeML Device. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, page 1–9. IEEE Press, 2021.

[63] Le Wu, Liji Wu, and Xiangmin Zhang. Catch the Star: Weight Recovery Attack Using Side-Channel Star Map Against DNN Accelerator. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 44(10):3697–3709, 2025.

[64] Le Wu, Liji Wu, Xiangmin Zhang, and Munkhbaatar Chinbat. Dual-Rail Precharge Logic-Based Side-Channel Countermeasure for DNN Systolic Array. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 32(9):1740–1743, 2024.

[65] Ville Yli-Mäyry, Akira Ito, Naofumi Homma, Shivam Bhasin, and Dirmanto Jap. Extraction of Binarized Neural Network Architecture and Secret Parameters Using Side-Channel Information. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2021.

[66] Kota Yoshida, Takaya Kubota, Shunsuke Okura, Mitsuru Shiozaki, and Takeshi Fujino. Model Reverse-Engineering Attack using Correlation Power Analysis against Systolic Array Based Neural Network Accelerator. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2020.

[67] Honggang Yu, Haocheng Ma, Kaichen Yang, Yiqiang Zhao, and Yier Jin. DeepEM: Deep Neural Networks Model Recovery through EM Side-Channel Information Leakage. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 209–218. IEEE, 2020.

[68] Lu Zhang, Dejun Mu, Yuxuan Huang, Jingyu Wang, Yifan He, Yaolei Li, Lizhou Liu, Kaiwei Zou, Huazhong Yang, and Yongpan Liu. Pareto Frequency-Aware Power Side-Channel Countermeasure Exploration on CNN Systolic Array. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 70(3):1124–1128, 2023.

A Technical appendices and supplementary material

A.1 Proof of TVLA Reduction with Increasing Parameter Dissimilarity

This section proves the inverse relationship between parameter dissimilarity measured via Euclidean distance and side-channel leakage. By decomposing the components of the t-test used in the TVLA, we analytically express the fixed and random set mean/variances as functions of the weight difference ($\|\Delta W\|$). The proof demonstrates that while the mean signal difference remains asymptotically constant, the induced leakage variance scales quadratically with the Euclidean distance. Consequently, we establish that enforcing weight divergence suppresses the TVLA t-score, effectively masking the side-channel leakage.

The side-channel leakage for a computation involving weight W and input x is defined as: $L = \mathrm{HW}(W \cdot x) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ where HW denotes the Hamming Weight and ϵ is additive Gaussian measurement noise, independent of both W and x. For the defense evaluation, we assume that a layer alternates between two weight versions, W_a and W_b, with equal probability ($p = 0.5$). We apply the standard TVLA using Welch's t-test to compare leakage from a *fixed set* (N_f measurements using a constant input x_f) against a *random set* (N_r measurements using uniformly random inputs x_r). The stochastic weight selection is continuously applied across all measurements in both sets. We now derive each component of the t-statistic in terms of the parameter versions W_a and W_b.

A.1.1 MEAN OF THE FIXED SET (μ_f)

For a fixed input vector x_f, each collected trace i utilizes a stochastically selected weight version. The instantaneous leakage is thus:

$$L_f^{(i)} = \begin{cases} \mathrm{HW}(W_a \cdot x_f) + \epsilon_i & \text{with probability } 0.5 \\ \mathrm{HW}(W_b \cdot x_f) + \epsilon_i & \text{with probability } 0.5 \end{cases} \tag{5}$$

Let $L_a = \mathrm{HW}(W_a \cdot x_f)$ and $L_b = \mathrm{HW}(W_b \cdot x_f)$ denote the noise-free leakage limits for each weight. Because the independent measurement noise ϵ has a mean of zero, the expected leakage for the fixed set resolves to the arithmetic mean of the two leakage states:

$$\mu_f = \frac{L_a + L_b}{2} \tag{6}$$

A.1.2 MEAN OF THE RANDOM SET (μ_r)

The random set measures side-channel leakage across uniformly distributed random inputs (x_r) while the hardware stochastically alternates between W_a and W_b. Because the weight selection ($p = 0.5$) is statistically independent of the input data, the Law of Total Expectation yields:

$$\mu_r = \frac{1}{2}\,\mathbb{E}_{x_r}[\mathrm{HW}(W_a \cdot x_r)] + \frac{1}{2}\,\mathbb{E}_{x_r}[\mathrm{HW}(W_b \cdot x_r)] \tag{7}$$

where $\mathbb{E}_{x_r}[\cdot]$ denotes the expectation (average) taken over the distribution of the random inputs x_r. Defining $\mu_a = \mathbb{E}_{x_r}[\mathrm{HW}(W_a \cdot x_r)]$ and $\mu_b = \mathbb{E}_{x_r}[\mathrm{HW}(W_b \cdot x_r)]$ as the expected leakage for each specific weight version, the total mean simplifies to their arithmetic average:

$$\mu_r = \frac{\mu_a + \mu_b}{2} \tag{8}$$

A.1.3 VARIANCE OF THE FIXED SET (σ_f^2)

For a constant input x_f, the noise-free leakage alternates between L_a and L_b with equal probability. We compute the structural variance of this alternating signal using the standard identity $\mathrm{Var}(L) = \mathbb{E}[L^2] - (\mathbb{E}[L])^2$. First, we determine the expected square and the squared expectation of the leakage:

$$\mathbb{E}[L^2] = \frac{1}{2}L_a^2 + \frac{1}{2}L_b^2 = \frac{L_a^2 + L_b^2}{2} \tag{9}$$

$$(\mathbb{E}[L])^2 = \left(\frac{L_a + L_b}{2}\right)^2 = \frac{L_a^2 + 2L_aL_b + L_b^2}{4} \tag{10}$$

Subtracting the squared expectation from the expected square resolves the variance of the discrete two-point distribution:

$$\mathrm{Var}(L) = \mathbb{E}[L^2] - (\mathbb{E}[L])^2 = \frac{2(L_a^2 + L_b^2) - (L_a^2 + 2L_aL_b + L_b^2)}{4} = \frac{(L_a - L_b)^2}{4} \tag{11}$$

Because the Gaussian measurement noise ϵ is statistically independent of the active weights, its variance (σ_ϵ^2) is strictly additive. Therefore, the total variance for the fixed set is:

$$\sigma_f^2 = \frac{(L_a - L_b)^2}{4} + \sigma_\epsilon^2 \tag{12}$$

A.1.4 VARIANCE OF THE RANDOM SET (σ_r^2)

To determine the variance of the random set, we account for two independent sources of variation: the uniformly distributed random inputs (x_r) and the stochastically alternating weights (W). We apply the Law of Total Variance to decompose the total structural variance into within-weight and between-weight components, appending the independent measurement noise at the end:

$$\sigma_r^2 = \underbrace{\mathbb{E}_W[\mathrm{Var}_{x_r}[L|W]]}_{\text{within-weight variance}} + \underbrace{\mathrm{Var}_W[\mathbb{E}_{x_r}[L|W]]}_{\text{between-weight variance}} + \sigma_\epsilon^2 \tag{13}$$

The **within-weight variance** captures the average leakage fluctuation caused by the random inputs for a given weight. Let $\sigma_a^2 = \mathrm{Var}_{x_r}[\mathrm{HW}(W_a \cdot x_r)]$ and $\sigma_b^2 = \mathrm{Var}_{x_r}[\mathrm{HW}(W_b \cdot x_r)]$ denote the

input-dependent leakage variances for each respective weight. Because the system selects each weight with equal probability, the expected variance is their arithmetic mean:

$$\mathbb{E}_W[\mathrm{Var}_{x_r}[L|W]] = \frac{\sigma_a^2 + \sigma_b^2}{2} \tag{14}$$

The **between-weight variance** captures how the conditional means (μ_a and μ_b) vary across the alternating weight versions. Since the hardware alternates between two states with equal probability, this resolves to a discrete two-point variance identical in mathematical structure to the fixed set variance derived in equation 12:

$$\mathrm{Var}_W[\mathbb{E}_{x_r}[L|W]] = \frac{(\mu_a - \mu_b)^2}{4} \tag{15}$$

Substituting these derived components into the Law of Total Variance yields the random set variance:

$$\sigma_r^2 = \frac{\sigma_a^2 + \sigma_b^2}{2} + \frac{(\mu_a - \mu_b)^2}{4} + \sigma_\epsilon^2 \tag{16}$$

A.1.5 Relationship between t-score and Euclidean Distance $\|\Delta W\|$

To ensure the functional accuracy of the protected layer is preserved, the divergence between the alternating parameters must be bounded. Therefore, we treat the difference ΔW as a minor perturbation around the center weight $\bar{W}$. To analyze the impact of weight divergence, we define the geometric center and the Euclidean difference of the parameters:

$$\bar{W} = \frac{W_a + W_b}{2}, \qquad \Delta W = W_a - W_b \qquad \Longrightarrow \qquad W_{a,b} = \bar{W} \pm \frac{\Delta W}{2} \tag{17}$$

Numerator: Independent of $\|\Delta W\|$

To ensure the functional accuracy of the protected layer is preserved, the divergence between the alternating parameters must be bounded. Therefore, we treat the difference ΔW as a perturbation around the center weight $\bar{W}$. Let $h = \frac{\Delta W}{2} \cdot x_f$ denote this small parameter perturbation. While the HW model is discrete, physical side-channel leakage (power or EM emission) is continuous. We abstract the physical emission as a continuous, differentiable leakage function $\mathcal{L}$ with respect to the parameter space. Because the divergence h is bounded, this physical continuity allows us to isolate the dominant leakage components using a Taylor expansion around the center point $\bar{W} \cdot x_f$:

$$L_a = \mathcal{L}(\bar{W} \cdot x_f) + \mathcal{L}'(\bar{W} \cdot x_f) \cdot h + \tfrac{1}{2}\mathcal{L}''(\bar{W} \cdot x_f) \cdot h^2 + \cdots \tag{18}$$

$$L_b = \mathcal{L}(\bar{W} \cdot x_f) - \mathcal{L}'(\bar{W} \cdot x_f) \cdot h + \tfrac{1}{2}\mathcal{L}''(\bar{W} \cdot x_f) \cdot h^2 + \cdots \tag{19}$$

Recall from equation 6 that the mean of the fixed set is $\mu_f = \frac{L_a + L_b}{2}$. Substituting the expansions into this definition, the first-order linear terms cancel due to $\pm$ symmetry:

$$\mu_f = \mathcal{L}(\bar{W} \cdot x_f) + O(\|\Delta W\|^2) \tag{20}$$

Applying an identical derivation for the random set over the distribution of inputs x_r yields:

$$\mu_r = \mathbb{E}_{x_r}[\mathcal{L}(\bar{W} \cdot x_r)] + O(\|\Delta W\|^2) \tag{21}$$

Finally, substituting equation 20 and equation 21 into the numerator of the t-statistic yields the signal difference. Because the second-order corrections $O(\|\Delta W\|^2)$ are negligible compared to the bigger zeroth-order baseline, the numerator evaluates to an approximately constant base signal K:

$$\mu_f - \mu_r \approx \mathcal{L}(\bar{W} \cdot x_f) - \mathbb{E}_{x_r}[\mathcal{L}(\bar{W} \cdot x_r)] \approx K \tag{22}$$

Denominator: Variances Scale Quadratically

Fixed Set Variance (σ_f^2). We now analyze the variance by subtracting L_b from L_a. Relying on the same continuous Taylor expansions defined in equation 18 and equation 19, the zeroth and even-order terms cancel during subtraction. This preserves the first-order linear derivative term as the dominant component:

$$L_a - L_b \approx 2\mathcal{L}'(\bar{W} \cdot x_f) \cdot h = \mathcal{L}'(\bar{W} \cdot x_f)\langle \Delta W, x_f \rangle \tag{23}$$

Letting $\alpha = \mathcal{L}'(\bar{W} \cdot x_f)$, we square the difference and apply the Cauchy-Schwarz inequality ($|\langle \Delta W, x_f \rangle| \leq \|\Delta W\|\|x_f\|$) to bound the term independently of the specific input x_f:

$$(L_a - L_b)^2 \leq \alpha^2 \|x_f\|^2 \|\Delta W\|^2 \implies (L_a - L_b)^2 \propto \|\Delta W\|^2 \tag{24}$$

Substituting this proportionality into equation 12 shows that the fixed set variance scales directly with the squared Euclidean distance: $\sigma_f^2 \propto \|\Delta W\|^2 + \sigma_\epsilon^2$.

Random Set Variance (σ_r^2). We analyze the structural components of equation 16 using the same continuous abstraction:

- Input Variation ($\frac{\sigma_a^2+\sigma_b^2}{2}$): Taylor expanding the input-dependent variance around $\bar{W}$, the first-order corrections carry opposite signs and cancel, leaving this baseline term approximately constant (σ_{input}^2).
- Weight Variation ($\frac{(\mu_a-\mu_b)^2}{4}$): Following the exact same subtraction logic as $L_a - L_b$, only odd-order terms survive. Defining $\gamma = \mathbb{E}_{x_r}[\mathcal{L}'(\bar{W} \cdot x_r) \cdot x_r]$, we find $\frac{(\mu_a-\mu_b)^2}{4} \propto \|\gamma\|^2\|\Delta W\|^2$.

Grouping the constant baseline variance terms together as C_{input}, the total random set variance simplifies to:

$$\sigma_r^2 \approx C_{\text{input}} + C_2\|\Delta W\|^2 \tag{25}$$

Substitution:

With a constant numerator K and the variances scaling quadratically with the Euclidean distance, we now synthesize these components into the final Welch's t-test statistic. Substituting the established proportionalities into equation 1, we introduce constant scaling factors C_1 and C_2 for the fixed and random set variances respectively. This allows us to expand the t-statistic as follows:

$$t \approx \frac{K}{\sqrt{\dfrac{C_1\|\Delta W\|^2 + \sigma_\epsilon^2}{N_f} + \dfrac{C_2\|\Delta W\|^2 + \sigma_{\text{input}}^2 + \sigma_\epsilon^2}{N_r}}} \tag{26}$$

To simplify this relationship, we isolate the terms dependent on the weight divergence from the baseline constants. We define $C = \frac{C_1}{N_f} + \frac{C_2}{N_r}$ to represent the combined distance-scaling coefficient, and we group the constant input variation and measurement noise terms into a single baseline denominator $C_0 = \frac{\sigma_\epsilon^2}{N_f} + \frac{\sigma_{\text{input}}^2+\sigma_\epsilon^2}{N_r}$. This reduction yields the final governing equation for the side-channel detectability:

$$\boxed{t \approx \frac{K}{\sqrt{C\|\Delta W\|^2 + C_0}}} \tag{27}$$

This formulation establishes two distinct operational regimes for the defense:

- **Noise-Dominated Regime ($C\|\Delta W\|^2 \ll C_0$):** When the Euclidean distance between the alternating weights is small, the structurally induced variance is masked by the inherent measurement and input noise. Here, the t-score remains relatively static, bottlenecked by the baseline noise profile C_0.
- **Weight-Dominated Regime ($C\|\Delta W\|^2 \gg C_0$):** As the divergence between the parameters grows, the quadratic variance overtakes the constant baseline noise. In this regime, the side-channel t-score drops inversely proportional to the Euclidean distance:

$$t \propto \frac{1}{\|\Delta W\|} \tag{28}$$

Ultimately, this proves the foundational mechanism of the defense: enforcing a large, bounded Euclidean distance between the alternating parameters suppress the t-score, thereby reducing the leakage.

A.2 Proof of TVLA generalization with multiple parameter versions

Having established the leakage reduction for two parameter versions, we now generalize the proof to an arbitrary number of versions.

A.2.1 SETUP AND NOTATION

Definition 1 (Multi-Version Configuration)**.** *A protected layer employs $V \geq 2$ functionally equivalent weight matrices $\{W_1, \ldots, W_V\} \in \mathbb{R}^M$. During each inference, the hardware stochastically selects one matrix uniformly at random, meaning each version operates with a probability of $1/V$.*

To analyze the structural variance induced by this ensemble, we define the geometric centroid (the average parameter center) $\bar{W}$, and the deviation δ_j of each individual weight matrix from this center:

$$\bar{W} = \frac{1}{V}\sum_{j=1}^{V} W_j, \qquad \delta_j = W_j - \bar{W} \tag{29}$$

By definition of the geometric centroid, the sum of all individual deviations cancels out. This establishes the **key zero-sum property**:

$$\sum_{j=1}^{V} \delta_j = 0 \tag{30}$$

Furthermore, we define two distance metrics to evaluate the parameter ensemble. The metric $d_{\min}$ represents the minimum Euclidean distance between any two distinct weight versions, effectively capturing the worst-case vulnerability of the defense. Meanwhile, $\bar{D}^2$ denotes the average squared pairwise distance across all unique pairs, which quantifies the overall spread of the entire ensemble:

$$d_{\min} = \min_{i \neq j} \|W_i - W_j\|_2, \qquad \bar{D}^2 = \frac{2}{V(V-1)} \sum_{i<j} \|W_i - W_j\|_2^2 \tag{31}$$

A.2.2 PAIRWISE VARIANCE DECOMPOSITION

To link the statistical variance of the leakage to the physical Euclidean distance between the weight matrices, we rely on the following fundamental statistical identity.

Lemma 2 (Pairwise Variance Identity)**.** *The variance of a uniformly distributed signal across V discrete states can be computed by summing the squared differences between all distinct pairs. For a set of values $\{z_1, \ldots, z_V\}$ selected with uniform probability $1/V$:*

$$\frac{1}{V}\sum_{j=1}^{V}(z_j - \bar{z})^2 = \frac{1}{V^2}\sum_{i<j}(z_i - z_j)^2 = \frac{V-1}{2V}\bar{D}_z^2 \tag{32}$$

Proof. Both sides of the equation simplify to $\frac{1}{V}\sum_j z_j^2 - \bar{z}^2$. This is achieved by applying the algebraic expansion $\sum_{i<j}(z_i - z_j)^2 = V\sum_j z_j^2 - (\sum_j z_j)^2$. □

Remark 3. *This scaling factor behaves predictably as the number of versions changes. For $V = 2$, the coefficient $\frac{V-1}{2V}$ evaluates to $1/4$, which recovers the $\frac{(L_a - L_b)^2}{4}$ variance factor derived in Section A.1. For $V = 3$, the factor is $1/3$, and as $V \to \infty$, the coefficient asymptotically approaches $1/2$.*

A.2.3 GENERALIZATION OF T-SCORE NUMERATOR (MEANS) WITH V VERSIONS

We now prove that, regardless of how many weight versions are used, the numerator of the Welch's t-test remains constant.

Theorem 4 (Numerator Independence for V Versions)**.** *The difference in expected leakage between the fixed and random sets evaluates to an approximately constant baseline K, independent of the number of versions V and their specific pairwise distances $\{d_{ij}\}$.*

Proof. By the Law of Total Expectation, the fixed and random set means are the uniform averages across all V states: $\mu_f = \frac{1}{V}\sum_j L_j$ and $\mu_r = \frac{1}{V}\sum_j \mu_j$. Following the methodology from Section A.1, we abstract the physical leakage as a continuous, differentiable function $\mathcal{L}$. Let $L_j = \mathcal{L}(W_j \cdot x_f)$ denote the fixed-input leakage, and $\mu_j = \mathbb{E}_{x_r}[\mathcal{L}(W_j \cdot x_r)]$ denote the expected random-input leakage.

We treat each parameter version as a perturbation around the centroid, letting $h_j = \delta_j \cdot x_f$. Applying a Taylor expansion to the sum of the fixed set leakages yields:

$$\sum_{j=1}^{V} L_j = V \cdot \mathcal{L}(\bar{W} \cdot x_f) + \mathcal{L}'(\bar{W} \cdot x_f) \cdot \underbrace{\sum_{j=1}^{V} h_j}_{=0} + O(\|\delta\|^2) \tag{33}$$

Crucially, because the sum of the deviations $\sum_j \delta_j = 0$, the aggregate sum of the perturbations $\sum_j h_j$ is zero. Consequently, all first-order linear terms cancel across the ensemble. Applying the identical logic to the random set sum $\sum_j \mu_j$ causes its first-order terms to vanish as well. Dividing by V and subtracting the two sets leaves only the massive zeroth-order baseline and negligible second-order corrections:

$$\mu_f - \mu_r \approx \mathcal{L}(\bar{W} \cdot x_f) - \mathbb{E}_{x_r}[\mathcal{L}(\bar{W} \cdot x_r)] \approx K \tag{34}$$

Thus, the signal difference is independent of the number of versions V. □

A.2.4 Generalization of t-score Denominator (Variances) with V Versions

Having established that the signal difference (the numerator) remains constant, we now analyze how the statistical variances scale when operating across an ensemble of V weight versions.

Theorem 5 (Generalized Fixed Set Variance)**.** *The total variance of the fixed set leakage scales directly with the average pairwise squared distance $\bar{D}^2$ of the weight ensemble:*

$$\sigma_f^2 = \frac{V-1}{2V} \cdot \beta \cdot \bar{D}^2 + \sigma_\epsilon^2 \tag{35}$$

where $\beta = \alpha^2 \|x_f\|^2$ represents the bounded proportionality constant, with $\alpha = \mathcal{L}'(\bar{W} \cdot x_f)$.

Proof. Applying Lemma 2, the structurally induced signal variance over V discrete weight states is given by the sum of their squared pairwise differences:

$$\sigma_{f,\text{signal}}^2 = \frac{1}{V^2} \sum_{i<j} (L_i - L_j)^2 \tag{36}$$

Recall from Section A.1 that by applying a continuous Taylor expansion around the centroid $\bar{W}$, the even-order terms of the physical leakage cancel during subtraction. Bounding the dominant first-order linear derivative via the Cauchy-Schwarz inequality yields the proportionality $(L_i - L_j)^2 \leq \alpha^2 \|x_f\|^2 \|W_i - W_j\|^2$. Letting $d_{ij}^2 = \|W_i - W_j\|^2$ denote the squared Euclidean distance between any two distinct versions, and defining our proportionality bound as β, we substitute this relationship into the pairwise sum:

$$\sigma_{f,\text{signal}}^2 \leq \frac{\beta}{V^2} \sum_{i<j} d_{ij}^2 = \frac{V-1}{2V} \cdot \beta \cdot \bar{D}^2 \tag{37}$$

Adding the independent Gaussian measurement noise σ_ϵ^2 to this induced variance completes the proof. □

Theorem 6 (Generalized Random Set Variance)**.** *The total variance of the random set leakage similarly scales with the average pairwise squared distance, augmented by the inherent input noise:*

$$\sigma_r^2 \approx \sigma_{\text{input}}^2 + \frac{V-1}{2V} \cdot \beta' \cdot \bar{D}^2 + \sigma_\epsilon^2 \tag{38}$$

where $\beta' = \|\gamma\|^2$ represents the bounded proportionality constant across the random input distribution, with $\gamma = \mathbb{E}_{x_r}[\mathcal{L}'(\bar{W} \cdot x_r) \cdot x_r]$.

Proof. To compute the variance of the random set across V versions, we decompose the total variance into within-weight and between-weight variation, mirroring the breakdown from Section A.1.

Within-Weight Variation (Input Noise): The average variance caused by the random inputs is $\frac{1}{V} \sum_{j=1}^{V} \sigma_j^2$. When we apply the Taylor expansion to these input-dependent variances around the centroid $\bar{W}$, the first-order corrections cancel out. This occurs via the exact same zero-sum mechanism ($\sum_j \delta_j = 0$) that keeps the t-test numerator constant. Therefore, the aggregate input variation simplifies to an approximately constant baseline, σ_{input}^2.

Between-Weight Variation: This term captures the variance of the conditional means μ_j across the parameter ensemble. Applying Lemma 2, we express this as:

$$\mathrm{Var}_W[\mathbb{E}_{x_r}[L|W]] = \frac{1}{V^2}\sum_{i<j}(\mu_i - \mu_j)^2 \tag{39}$$

Applying the same linear approximation and Cauchy-Schwarz bounding used for the fixed set, each pairwise difference satisfies $(\mu_i - \mu_j)^2 \propto d_{ij}^2$. Letting β' represent the proportionality bound defined by γ, the sum evaluates to $\frac{V-1}{2V} \cdot \beta' \cdot \bar{D}^2$.

Combining the within-weight variation (σ^2_{input}), the between-weight variation ($\frac{V-1}{2V} \cdot \beta' \cdot \bar{D}^2$), and the independent measurement noise (σ^2_ϵ) completes the proof. □

A.2.5 SUBSTITUTION

Having derived the generalized components, we now synthesize them into a single, closed-form equation. To evaluate the robustness of the defense, we are interested in the worst-case security scenario (the highest possible leakage detectability). This occurs when the parameter versions are packed as closely together as possible. Therefore, we lower-bound the average squared distance with the minimum pairwise distance: $\bar{D}^2 \geq d^2_{\min}$. This substitution minimizes the variance denominator, providing an upper bound on the resulting t-score.

Theorem 7 (Generalized T-Score Upper Bound)**.** *Assuming a balanced trace collection where $N_f = N_r = N/2$, and substituting the worst-case distance bound $\bar{D}^2 \geq d^2_{\min}$, the Welch's t-test statistic for a V-version parameter ensemble is bounded by:*

$$\boxed{t(V, d_{\min}) \leq \frac{K}{\sqrt{A \cdot \frac{V-1}{V} \cdot d^2_{\min} + B}}} \tag{40}$$

where the constants, which are fundamentally independent of both the version count V and the Euclidean distance $d_{\min}$, are defined as:

$$K = \mu_f - \mu_r \qquad \textit{(baseline signal strength)} \tag{41}$$

$$A = \frac{\beta + \beta'}{N} \qquad \textit{(structural variance / total traces)} \tag{42}$$

$$B = \frac{2(2\sigma^2_\epsilon + \sigma^2_{\text{input}})}{N} \qquad \textit{(aggregate noise / total traces)} \tag{43}$$

Proof. We construct the generalized t-statistic by substituting the results of Theorems 4, 5, and 6 into the standard Welch's t-test equation equation 1. Using the balanced collection assumption ($N_f = N_r = N/2$), the total variance denominator becomes:

$$t = \frac{K}{\sqrt{\frac{2}{N}(\sigma^2_f + \sigma^2_r)}} \tag{44}$$

$$= \frac{K}{\sqrt{\frac{2}{N}\left[\left(\frac{V-1}{2V} \cdot \beta \cdot \bar{D}^2 + \sigma^2_\epsilon\right) + \left(\sigma^2_{\text{input}} + \frac{V-1}{2V} \cdot \beta' \cdot \bar{D}^2 + \sigma^2_\epsilon\right)\right]}} \tag{45}$$

Grouping the structurally induced variance terms and isolating the inherent system noise yields:

$$t = \frac{K}{\sqrt{\frac{\beta+\beta'}{N} \cdot \frac{V-1}{V} \cdot \bar{D}^2 + \frac{2(2\sigma^2_\epsilon + \sigma^2_{\text{input}})}{N}}} \tag{46}$$

Applying the worst-case assumption ($\bar{D}^2 \geq d^2_{\min}$) bounds the expression. Finally, substituting the defined constants A and B resolves the master formula:

$$t(V, d_{\min}) \leq \frac{K}{\sqrt{A \cdot \frac{V-1}{V} \cdot d^2_{\min} + B}} \tag{47}$$

□

Remark 8 (Consistency with Section A.1)**.** *This generalized formula reduces to the specific two-version derivation established in Section A.1. For two versions ($V = 2$), the version-scaling*

coefficient $\frac{V-1}{V}$ evaluates to $1/2$. Consequently, the variance term becomes $A \cdot \frac{1}{2} \cdot d^2 = \frac{\beta+\beta'}{2N} d^2$. Assuming $d = \|\Delta W\|$, this algebraically recovers the exact $C\|\Delta W\|^2$ baseline proportionality derived in equation 27.

A.3 Predicting TVLA Security

While the generalized master formula establishes the theoretical bounds of the defense, directly computing the system-dependent constants A, B, and K analytically is often impractical due to complex physical noise profiles. To bridge this gap, we introduce a one-time two-point empirical calibration method. By conducting just two profiling measurements, a system designer can bypass these constants and predict the worst-case side-channel leakage for any arbitrary defense configuration.

Theorem 9 (Two-Point Calibration)**.** *Let t_1 denote the empirical t-score of the unprotected baseline network ($V = 1$), and let t_2 denote the empirical t-score of a protected network profiling run using two versions ($V = 2$) separated by a known Euclidean distance d. Using only these two hardware measurements, the upper-bound t-score for any arbitrary configuration of V versions with a minimum distance $d_{\min}$ can be predicted as:*

$$\boxed{t(V, d_{\min}) \leq \frac{t_1}{\sqrt{\frac{V-1}{V} \cdot \left(\frac{A}{B}\right) \cdot d_{\min}^2 + 1}}} \tag{48}$$

where the environment-specific structural-to-baseline noise ratio, $\frac{A}{B}$, is empirically derived as:

$$\frac{A}{B} = \frac{2}{d^2} \left[\left(\frac{t_1}{t_2} \right)^2 - 1 \right] \tag{49}$$

Proof. We begin by evaluating the generalized final formula equation 40 for the unprotected baseline. For a single static weight ($V = 1$), the distance-scaling variance term nullifies, leaving only the baseline signal and noise:

$$t_1 = \frac{K}{\sqrt{B}} \qquad \Longrightarrow \qquad K = t_1 \sqrt{B} \tag{50}$$

Next, we evaluate the formula for the two-version profiling run ($V = 2$) operating at a known distance $d_{\min} = d$. Substituting these parameters, we obtain:

$$t_2 = \frac{K}{\sqrt{A \cdot \frac{1}{2} \cdot d^2 + B}} \tag{51}$$

Substituting $K = t_1 \sqrt{B}$ into equation 51 allows us to link the two empirical measurements. Factoring out the baseline noise B in the denominator isolates the ratio $\frac{A}{B}$:

$$t_2 = \frac{t_1 \sqrt{B}}{\sqrt{B\left(\frac{A}{B} \cdot \frac{d^2}{2} + 1\right)}} = \frac{t_1}{\sqrt{\frac{A}{B} \cdot \frac{d^2}{2} + 1}} \tag{52}$$

Squaring both sides and algebraically isolating the ratio yields the calibration factor equation 49. Finally, substituting the baseline substitution $K = t_1 \sqrt{B}$ into the full master formula equation 40 and canceling B resolves the general predictive equation equation 48. □

A.4 Additional Empirical Evaluation Details

A.4.1 Experimental Setup Details

Target Device: We perform all side-channel measurements on a Google Coral Dev Board featuring an Edge TPU System-on-Module (SoM) (25). The board runs Mendel Linux, and our Python inference scripts execute compiled TensorFlow Lite models using the Edge TPU runtime through the TensorFlow Lite interpreter (26). A host PC communicates with the board through USB-C and UART interfaces to control inference execution and coordinate trace acquisition.

Measurement Equipment: Figure 4 shows the full measurement setup, and Figure 5 shows the probe-positioning procedure. We acquire EM emanations using a Keysight[3] EM measurement setup consisting of a high-sensitivity EM probe, probe station, and motorized XYZ table (38). To improve EM trace SNR while maintaining cooling, we remove the SoM cooling fan and use an external fan during acquisition. We capture traces using a PicoScope 6000E oscilloscope at 1.25 GS/s (51). The EM probe is connected to channel A of the oscilloscope, while the GPIO trigger is connected to channel B. Because the Edge TPU floorplan is not public, we use the motorized XYZ table to scan the chip package during continuous inference. We first identify the dominant operating-frequency region around 500 MHz, then use the spatial EM heat map to locate the coordinate with maximum TPU activity. We fix the probe at this hotspot for all subsequent acquisitions.

Acquisition and Post-processing Compute: Trace acquisition and post-processing run on a separate workstation with an Intel Core i7-9700K CPU at 3.60 GHz and 64 GB RAM.

Trace Acquisition: We configure an Edge TPU GPIO pin as a hardware trigger. For each trace, the host PC initiates an inference, the Edge TPU asserts the GPIO trigger, and the oscilloscope captures the EM waveform. After each inference, the Edge TPU deasserts the trigger. We collect fixed-input and random-input trace sets using the trace counts reported for each experiment (54).

Trace Post-processing and Alignment: We use Keysight Inspector SCA 2025.12 through its Python API for alignment, TVLA computation, trace averaging, and plotting. OS scheduling, memory activity, and interleaved CPU/TPU operations introduce system-level timing variation, so a single static alignment point does not align the full inference trace. We therefore apply static alignment over layer-specific regions piecewise before computing TVLA. To identify these regions, we first locate macro-layer computations using heuristics from prior work on neural-network hyperparameter extraction (41). We then generate templates for predicted layer operations using single-layer sub-models and match them against runtime measurement profiles. These matched templates define the layer-specific windows used for alignment and TVLA evaluation.

A.4.2 Training Details for Composability and Accuracy Evaluation

Training Hardware: We train all models on a workstation with an AMD Ryzen 9 7950X 16-Core Processor at 4.50 GHz, 128 GB RAM, and an NVIDIA GeForce RTX 4090 GPU with 24 GB VRAM.

Datasets and Preprocessing: We use MNIST and CIFAR-10 for model training and accuracy evaluation (42; 39). All task-model inputs are normalized to $[0, 1]$. MLP inputs are flattened, while CNN inputs remain in spatial form. We split the original training set into 90% training and 10% validation, and use the standard held-out test set for final accuracy evaluation. The protected-model training script also generates random selector arrays for the training, validation, and test splits to evaluate stochastic version selections after training.

Unprotected Model Training: The unprotected models use the same task architectures as the protected models, but with one parameter set per layer. They do not use stacked weights, selector inputs, gradient masking, or diversity loss. The MLP unprotected model uses fully connected layers. The CNN unprotected model uses the same convolutional, pooling, flattening, and dense sequence as the protected CNN. We train the unprotected models with SGD, learning rate 0.005, batch size 64, cross-entropy loss, and 100,000 optimization steps.

Protected Model Training: The protected models use the stochastic layer-wise training procedure from Section 3. Each defended trainable layer stores V stacked parameter versions. At each mini-batch, the model samples one version per defended layer and computes task loss on the sampled path. The total loss combines cross-entropy with a diversity regularizer weighted by λ_{div}, which penalizes similarity between versions of the same layer. Gradient masking updates only the sampled version of each defended layer. We train the protected models with SGD, learning rate 0.05, batch size 64, and 50 epochs.

Model Architecture: For MNIST (42), we use a 784–100–10 MLP and protect both dense layers. We evaluate configurations with $V = 2$ and $\lambda_{\text{div}} = 0.3$, $V = 8$ and $\lambda_{\text{div}} = 12{,}500$, and a sweep from $V = 2$ to $V = 8$ with $\lambda_{\text{div}} = 0.3$. We use this MLP for the theoretical-to-empirical TVLA

[3]We use Keysight in the text to reflect the current vendor name after Keysight's acquisition of Riscure. The figures retain Riscure labels because the equipment and software interfaces used during acquisition display that name.

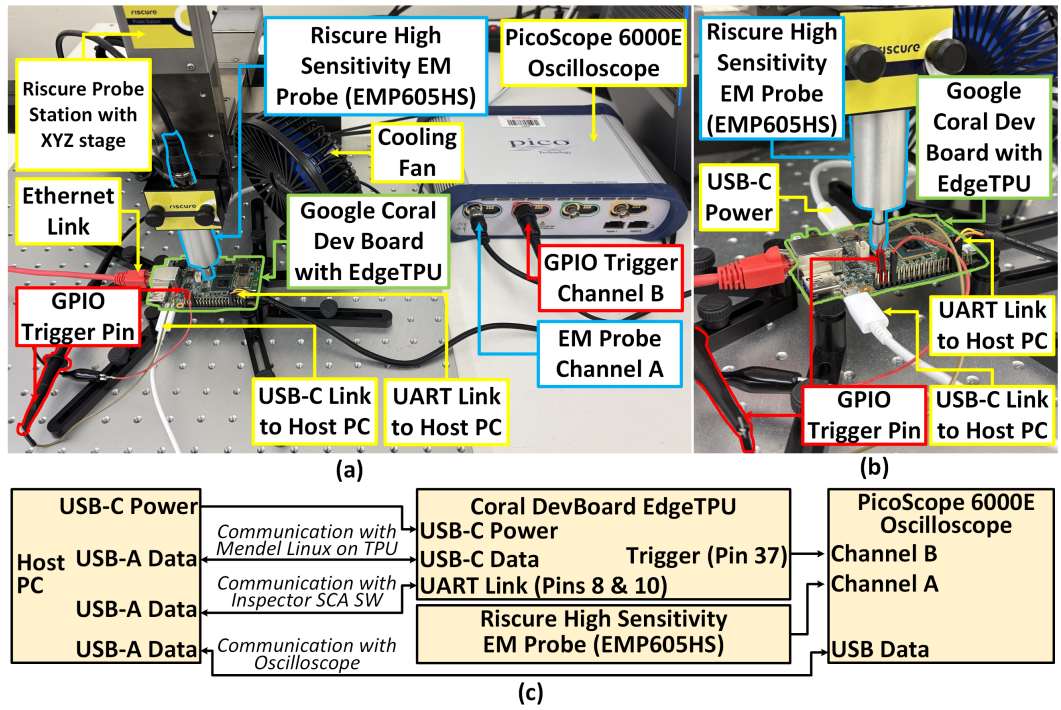


Figure 4: EM side-channel measurement setup for the Google Coral Dev Board with Edge TPU. (a) Full measurement setup showing the Riscure probe station, high-sensitivity EM probe, Google Coral Dev Board, PicoScope 6000E oscilloscope, GPIO trigger connection, and host-PC links. (b) Close-up view of the EM probe positioned over the Edge TPU SoM. (c) Schematic diagram depicting the connections between the host PC, Coral Dev Board, EM probe, and oscilloscope.

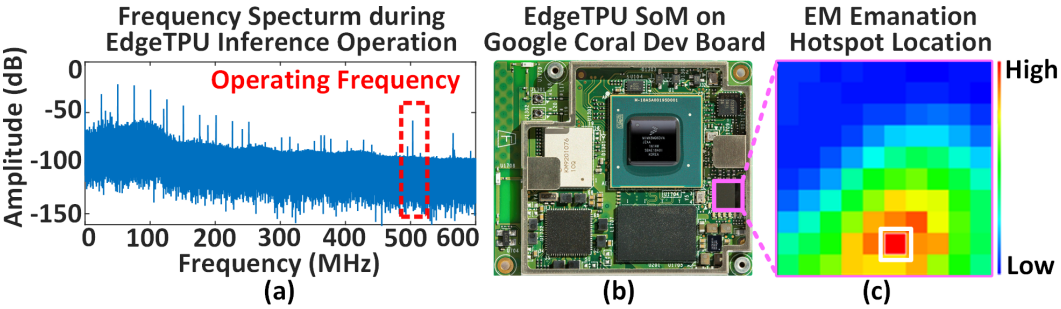


Figure 5: Operating-frequency extraction and EM hotspot identification for the Edge TPU. (a) Frequency spectrum measured during Edge TPU inference, showing the dominant operating-frequency region around 500 MHz. (b) Edge TPU SoM mounted on the Google Coral Dev Board, with the scanned package region highlighted. (c) EM heat map obtained from the spatial scan, where warmer colors indicate stronger EM activity. The hotspot marked in red is used as the fixed probe location for subsequent side-channel acquisitions.

validation, trace-count evolution, and defense-aware clustering analyses. For CIFAR-10 (39), we use a CNN with four 3×3 convolutional layers containing 32, 32, 64, and 64 filters, followed by 2×2 max-pooling, flattening, and two dense layers with 256 and 10 neurons. We protect all convolutional and dense layers. We evaluate a configuration with $V = 2$ for all protected layers and $\lambda_{\text{div}} = 60$, and a higher-diversity configuration with $V = 2$ for the convolutional layers, $V = 8$ for the dense layers, and $\lambda_{\text{div}} = 100$. Pooling and flattening layers are not protected because they contain no trainable parameters. For ImageNet-1K (15; 53), we evaluate EfficientNet-EdgeTPU-S (60; 28). We protect the first two vulnerable layers identified in prior work (32) using $V = 3$ and $\lambda_{\text{div}} = 0.3$.

Composability Evaluation: We evaluate functional composability by testing whether layer-version selections preserve task accuracy after training. For each protected configuration, we evaluate

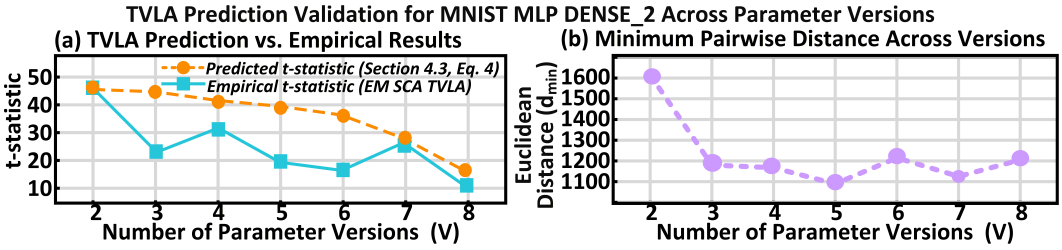


Figure 6: Theoretical-to-empirical TVLA prediction for DENSE_2 of the MNIST MLP with $\lambda_{\text{div}} = 0.3$ and V varied from 2 to 8. (a) Predicted and empirical peak TVLA t-statistics. (b) Minimum pairwise Euclidean distance $d_{\min}$ between parameter versions for each V.

stochastic inference using sampled layer-version selections and compare test accuracy against the corresponding unprotected model. These results show that the protected models used for TVLA remain task-accurate under random parameter selection.

Quantization and Edge TPU compilation: After training, we export each model to TensorFlow Lite, quantize it with representative training samples, and compile the quantized model with the Edge TPU compiler. Protected models include the input tensor and one selector tensor per defended layer. Unprotected models take only the input tensor. We evaluate final accuracy on both floating-point and quantized TensorFlow Lite models before using the compiled model for Edge TPU side-channel measurements.

A.5 TVLA Prediction and Layer-Dependent Leakage Characterization

This section provides the detailed analysis supporting Section 5.2. We first validate the calibrated theoretical TVLA model across different parameter-version counts and examine how inter-version separation affects the predicted leakage reduction. We then study how the empirical TVLA t-statistic evolves with increasing trace count and compare the effect of parameter shuffling across the two dense layers of the MNIST MLP.

A.5.1 Theoretical-to-Empirical TVLA Prediction

Our theoretical TVLA model estimates the measured t-statistic across defense configurations, enabling design-space exploration without empirical side-channel measurements for every candidate configuration. We calibrate the model using the $V = 2$ configuration and then evaluate its predictions on new configurations by varying V from 3 to 8 with $\lambda_{\text{div}} = 0.3$. For each configuration, we collect 20K total fixed-vs-random traces and compare predicted and empirical TVLA t-statistics for DENSE_1 and DENSE_2. For DENSE_2, Figure 6(a) shows that the theoretical estimates remain above the empirical measurements for $V = 3$ to $V = 8$, with a mean absolute difference of 12.64 t-statistic units. Figure 7(a) shows similar agreement for DENSE_1, with a mean absolute difference of 4.92 t-statistic units over the same range.

Figure 6(b) and Figure 7(b) also report the minimum pairwise Euclidean distance $d_{\min}$ between parameter versions. For DENSE_2, $d_{\min}$ decreases as V increases, consistent with additional versions having less freedom to remain separated within its smaller parameter space (100×10) while preserving model accuracy. In contrast, DENSE_1, with its larger parameter space (784×100), does not show the same decrease in $d_{\min}$. This behavior is consistent with the larger parameter space providing greater freedom for additional versions to remain separated. These results show that increasing V alone does not guarantee proportional leakage reduction: the benefit of additional version-induced variance depends on how well the parameter versions remain separated. Because the calibrated model captures both version count and inter-version separation, it can account for this behavior and support exploration of candidate V and λ_{div} configurations without requiring empirical side-channel measurements for every candidate.

A.5.2 Layer-Dependent Leakage Characterization

We next examine how the empirical TVLA t-statistic evolves as the measurement budget increases. For the MNIST MLP, we compare the unprotected model with protected configurations using $V = 2$ and $V = 8$ with $\lambda_{\text{div}} = 0.3$, and a higher-diversity configuration with $V = 8$ and $\lambda_{\text{div}} = 12{,}500$. We acquire up to 100K total fixed-vs-random traces. Figure 8 reports the peak TVLA t-statistic against the effective per-set trace count for DENSE_1 and DENSE_2.

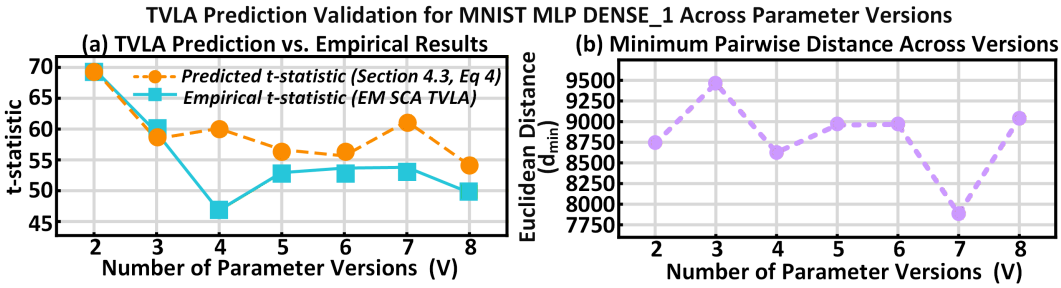


Figure 7: Theoretical-to-empirical TVLA prediction for DENSE_1 of the MNIST MLP with $\lambda_{\text{div}} = 0.3$ and V varied from 2 to 8. (a) Predicted and empirical peak TVLA t-statistics. (b) Minimum pairwise Euclidean distance $d_{\min}$ between parameter versions for each V.

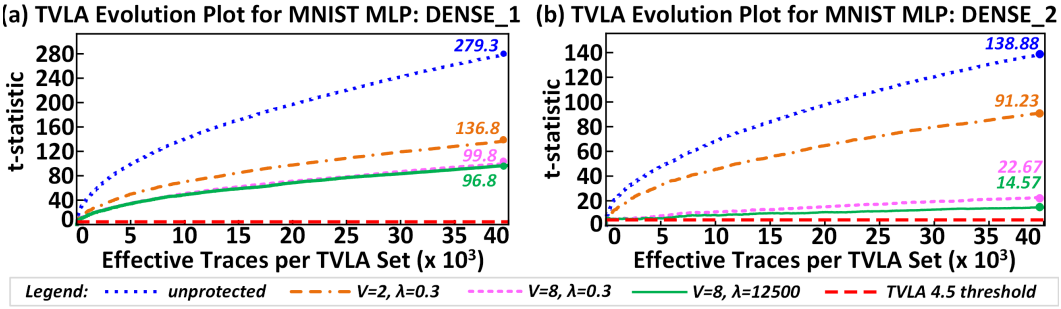


Figure 8: Evolution of peak TVLA t-statistic with increasing trace count for the MNIST MLP. The protected configurations show slower growth in the peak t-statistic as the trace count increases, with the strongest suppression in DENSE_2.

The results reveal a layer-dependent effect of parameter shuffling. For both layers, the peak t-statistic increases as additional traces are included, reflecting the increased statistical power of the TVLA test. However, the protected configurations exhibit slower t-statistic growth than the unprotected model, with a substantially larger reduction in DENSE_2. In contrast, DENSE_1 remains comparatively high despite protection. This weaker reduction is consistent with input-dependent leakage in the first layer, where the directly processed input remains correlated with the measured computation (18). The stronger suppression in DENSE_2 is consistent with deeper layers benefiting from both their own parameter shuffling and randomized intermediate activations propagated from earlier protected layers. Thus, the effectiveness of parameter shuffling is layer-dependent, with substantially stronger suppression observed in the deeper DENSE_2 layer. Reducing first-layer leakage further may therefore require stronger or layer-specific diversity regularization rather than a shared protection configuration across layers.

A.6 Defense-Aware Adaptive Clustering Analysis

We analyze whether the hidden parameter-version selector values leave separable signatures in the measured EM side-channel traces. If such signatures exist, an attacker could group traces that use the same parameter version and analyze each group separately for parameter extraction, weakening the protection provided by dynamic parameter shuffling. Since parameter-extraction attacks are typically performed layer-wise (32), we focus on DENSE_1 of the MNIST MLP as a representative case. We assume that the attacker can collect traces under fixed and random or otherwise known inputs, localize the DENSE_1 execution region using prior hyperparameter-extraction heuristics (41), and knows the number of parameter versions V, but not the selector value r_1. The attacker uses the fixed-vs-random TVLA trace only to identify an attacker-visible high-leakage region for subsequent unsupervised clustering.

We use the attacker-visible TVLA-peak region within DENSE_1 as the feature region. For the MNIST MLP models with $V = 3$ and $V = 8$, both using $\lambda_{\text{div}} = 0.3$, we form balanced evaluation sets of 2,000 fixed-input traces per selector value r_1. We fit principal component analysis (PCA) (37) to the selected trace region without using selector labels and project the traces onto PC1–PC2. We standardize the PC1–PC2 scores to zero mean and unit variance before applying K-means (48) and

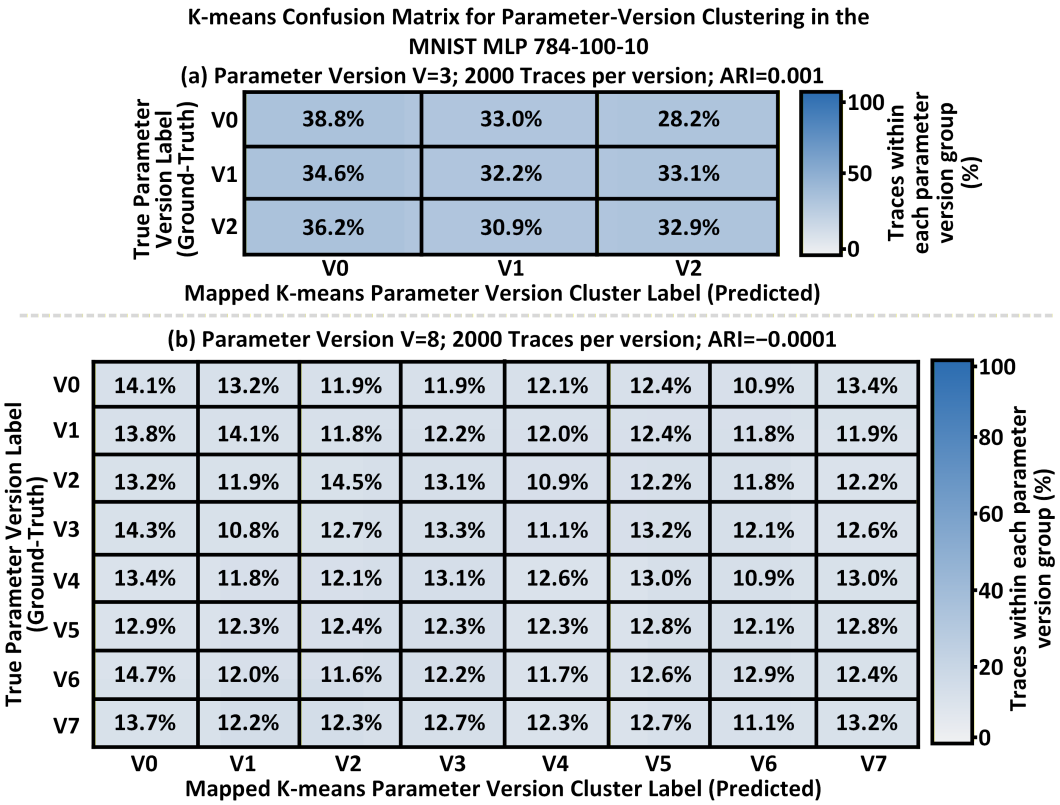


Figure 9: K-means parameter-version clustering for the protected MNIST MLP with $V = 3$ and $V = 8$, using 2,000 traces per version. Best-mapped accuracies remain close to chance at 34.62% versus 33.3% for $V = 3$ and 13.43% versus 12.5% for $V = 8$, with adjusted Rand index (ARI) values near zero.

Gaussian mixture model (GMM) clustering (4). We use $K = V$ clusters for K-means and V mixture components for GMM. The trace metadata records r_1 for each inference; we use these labels only to construct the balanced evaluation sets, visualize the selector groups, and evaluate the resulting clusters. Because cluster indices are arbitrary, we compute best-mapped accuracy after clustering using a post-hoc one-to-one assignment between cluster IDs and recorded selector labels. Selector labels do not affect PCA, PC1–PC2 selection, feature standardization, or clustering.

The trace groups corresponding to different parameter versions fail to form distinct clusters under the evaluated clustering attack. As shown in Figure 3, the PC1–PC2 projections for $V = 3$ and $V = 8$ exhibit substantial overlap among traces generated using different parameter versions. Figure 9 provides the corresponding K-means clustering results. For $V = 3$, K-means achieves a best-mapped accuracy of 34.62%, close to the 33.3% chance level, with an adjusted Rand index (ARI) of 0.0010. For $V = 8$, the best-mapped accuracy is 13.43%, close to the 12.5% chance level, with an ARI of -0.0001. GMM gives similar results, with best-mapped accuracies of 35.08% and 13.59% and ARI values of 0.0012 and 0.0001 for $V = 3$ and $V = 8$, respectively. These near-chance clustering accuracies and near-zero ARI values show that the evaluated PC1–PC2 attack does not reliably group traces by the parameter version used during inference.