Overcoming Temporal Shuffling in Non-Profiled SCA: A Translation-Invariant Deep Learning Approach

Ahmed Ismail[1], Eid Emary[2], Hala Abbas[3]
Faculty of Computer Studies, Arab Open University Cairo, Egypt[1,2,3]
Faculty of Computers and Artificial Intelligence-Department of Information Technology, University of Cairo, Cairo, Egypt[2]
Faculty of Computers and Artificial Intelligence-Computer Science Department, Capital University, Cairo, Egypt[3]

***Abstract*—Side-Channel Analysis (SCA) utilizing Deep Learning has demonstrated significant potential in recovering secret keys from cryptographic implementations. However, the efficiency of these attacks is often severely compromised by hardware countermeasures such as temporal shuffling, which desynchronizes leakage traces. Existing non-profiled collision attacks successfully mitigate shuffling, but often rely on a "Grey-Box" threat model, requiring prior knowledge of the shuffle permutation to align traces before analysis. This study presents a Global Average Pooling Convolutional Neural Network (GAP-CNN) designed to exploit side-channel collisions in a strict Black-Box setting. By integrating a translation-invariant GAP layer, the proposed architecture forces the network to learn the presence of leakage signatures regardless of their temporal location, effectively neutralizing the shuffling countermeasure end-to-end without pre-processing. The methodology is evaluated on the DPA Contest v4.2 dataset, a highly protected AES-128 implementation. The empirical results demonstrate that the proposed Black-Box approach successfully recovers a majority of the target bytes, outperforming previous Grey-Box baselines. Furthermore, the study demonstrates strong cross-byte portability and cross-dataset robustness against masking countermeasures (ASCAD), confirming the existence of exploitable leakage clusters that persist despite advanced randomization.**

***Keywords*—Side-Channel Analysis; Deep Learning; collision attack; shuffling countermeasure; Global Average Pooling; AES**

I. Introduction

Side-Channel Analysis (SCA) has established itself as a critical threat to modern embedded cryptographic systems. Since the seminal introduction of Differential Power Analysis (DPA) by Kocher et al. [1], numerous statistical techniques such as Correlation Power Analysis (CPA) [2] have been developed to recover secret keys from physical leakages. Traditionally, these attacks relied on Gaussian Template Attacks [3], which require a profiled device and strict statistical assumptions.

In recent years, the field has shifted towards Deep Learning-based SCA (DL-SCA). First introduced by Maghrebi et al. [4], neural networks have demonstrated an ability to extract complex leakage features from raw, noisy measurements. However, the efficacy of DL-SCA is often challenged by hardware countermeasures like clock jitter and shuffling. While data augmentation techniques can mitigate jitter [5], shuffling—which desynchronizes operations—remains a formidable barrier for standard Convolutional Neural Networks (CNNs).

Existing solutions, such as the collision attack by Staib and Moradi [6], often operate under a "Grey-Box" threat model, assuming prior knowledge of the shuffling patterns to pre-align traces. Consequently, a distinct research gap exists between the theoretical capability of CNNs and their practical application against heavily desynchronized targets. Standard CNN architectures rely on dense spatial hierarchies that fail when leakage signatures drift randomly across wide temporal windows. To bridge this gap, this study proposes a strict Black-Box methodology utilizing a Global Average Pooling CNN (GAP-CNN) that inherently extracts translation-invariant features, bypassing the need for explicit de-synchronization steps or assumed knowledge of the randomization permutation.

A. Main Contributions

The main contributions of this research are summarized as follows:

- Strict Black-Box Collision Attack: A deep learning-based collision attack methodology is introduced that operates entirely without knowledge of the underlying shuffle permutation or time-offsets.
- Translation-Invariant Architecture: The GAP-CNN is proposed, replacing standard dense flattening with a Global Average Pooling layer to neutralize temporal shuffling end-to-end without trace pre-processing.
- High Data Efficiency: It is demonstrated that the proposed approach achieves a 53.3% success rate (recovering 8 out of 15 bytes) on the highly protected DPA Contest v4.2 dataset using 3,000 attack traces.
- Cross-Dataset Robustness: Comprehensive statistical analysis validates the cross-dataset robustness of the proposed architecture, demonstrating its effectiveness against both shuffling (DPAv4.2) and Boolean masking (ASCAD).

B. Paper Organization

The remainder of this study is comprehensively structured to guide the reader through the theoretical framework, proposed methodology, and experimental validation. Section II provides the necessary theoretical background on Side-Channel

Analysis and shuffling countermeasures. Section III details the proposed GAP-CNN architecture, conceptually illustrated in Fig. 1, and formally defines the strict Black-Box threat model. Section IV describes the experimental setup, datasets, and the robust training strategies employed. Section V presents the empirical results, including the statistical attack convergence (summarized visually in Fig. 2), the ablation study, and the cross-dataset validation. Section VI analyzes the physical implications of the observed hardware leakage clusters. Finally, Section VII summarizes the achievements of this research.

II. Background and Related Work

A. Side-Channel Collision Attacks

Collision attacks exploit the principle that if two identical data values are processed by a device, they will generate identical leakage traces, regardless of *when* they are processed. This concept was originally proposed by Schramm et al. [7] and later improved for AES by Bogdanov [8]. While classical approaches like Moments-Correlating DPA [9] utilize high-order statistics to detect these collisions, they often struggle with high-noise environments. The proposed work automates this collision detection using deep neural networks.

B. Mathematical Derivation of RSM Collisions

The DPA Contest v4.2 implements a Rotating S-Box Masking (RSM) scheme. Let X be the input byte and k be the subkey. In an unmasked implementation, the sensitive intermediate value is $Y = \text{SBox}(X \oplus k)$.

In RSM, the S-Box is replaced by a masked version S', and a public mask M is applied. The operation becomes:

$$Y' = S'((X \oplus k) \oplus M_{offset}) \tag{1}$$

where, M_{offset} is a mask value that changes (rotates) for every byte index $i \in \{0...15\}$.

The vulnerability exploited in this work arises from the fact that the underlying hardware operation for the S-Box is identical regardless of the byte index. If a "collision" is found where the leakage of Byte i (L_i) equals the leakage of Byte j (L_j), it can be inferred that the processed values were identical:

$$(P_i \oplus k_i) \oplus M_i = (P_j \oplus k_j) \oplus M_j \tag{2}$$

In the proposed "Anchor Model" attack, Byte $i = 0$ is fixed (where k_0 effectively acts as a reference) to learn and predict the leakage L_0. When this model is applied to Byte j, the network identifies traces where the leakage L_j matches the learned profile of L_0. The GAP-CNN effectively learns the mapping function f : Trace $\rightarrow (P \oplus k \oplus M)$, allowing for the recovery of the key difference $\Delta k = k_i \oplus k_j$ despite the shuffling of operations in time.

C. Non-Profiled Deep Learning SCA

Unlike profiled attacks that require an open device, non-profiled attacks recover the key directly from the target device. Timon [10] introduced the concept of "Deep Learning-based DPA", showing that sensitivity analysis could guide a neural network to the correct key. Further optimizations by Kwon et al. [11] demonstrated that efficient architectures could significantly reduce the trace requirements for these attacks. Furthermore, recent literature emphasizes the critical need for advanced architectural designs to handle complex side-channel environments. Recent studies have highlighted the growing trend of leveraging attention mechanisms within CNNs to improve feature extraction and attack efficiency [12], while other advancements have focused on bridging the gap between profiled and non-profiled assumptions to create more realistic attack models [13]. This study builds upon these modern insights by practically applying a GAP-CNN architecture to achieve translation invariance against shuffling.

D. Evaluation Datasets

The primary benchmark utilized in this study is the DPA Contest v4.2 dataset [14], which implements AES-128 protected by a Rotating S-Box Masking (RSM) scheme and a temporal shuffling countermeasure. This dataset serves as a rigorous standard for evaluating attacks against temporal desynchronization. Furthermore, to evaluate cross-dataset robustness against different countermeasure paradigms, the ASCAD dataset [15] is also targeted. ASCAD provides a standardized benchmark of electromagnetic traces from a software AES implementation protected by Boolean masking, serving as an ideal testbed to verify architectural generalization.

III. Proposed Methodology: GAP-CNN

In this section, the threat model is formally defined and the *Global Average Pooling Convolutional Neural Network* (GAP-CNN) is introduced. The proposed methodology is designed to overcome the limitations of prior works [6] by enforcing translation invariance at the architectural level, thereby eliminating the need for pre-processing or shuffle-pattern recovery.

A. Threat Model

A strict *Black-Box Non-Profiled* adversary model is considered in this research. it is assumed that the adversary does not possess profiling traces from an open, identical clone device. Instead, both the neural network training phase (to build the anchor model) and the subsequent attack phase are conducted directly on traces captured from the exact same target device. Furthermore, the training traces and the attack traces are strictly disjoint sets to ensure a realistic evaluation and prevent data leakage. The adversary has access to:

- A set of raw power consumption traces T measured from the target device.
- The associated public inputs (plaintexts) P.
- Knowledge of the cryptographic algorithm (AES-128).

Crucially, unlike the "Grey-Box" model assumed in prior collision literature [6], it is assumed the adversary has no knowledge of the random shuffling permutation π or the specific time-offsets of operations. The attack must succeed directly on the raw, desynchronized traces.

B. The GAP-CNN Architecture

Standard Convolutional Neural Networks (CNNs) maintain spatial (or temporal) structures throughout their feature extraction layers. In a typical CNN, the final convolutional feature

map is flattened into a vector before being passed to a dense layer. If the input leakage shifts in time (due to shuffling), the flattened vector changes significantly, forcing the dense layer to learn all possible time-offsets, which is computationally infeasible for high-entropy shuffling.

To address this, the traditional flattening operation is replaced with a Global Average Pooling (GAP) layer [16]. Let F^l be the feature map output of the last convolutional layer, consisting of K feature maps of length L. The GAP operation computes the mean of each feature map:

$$GAP(F_k^l) = \frac{1}{L}\sum_{t=1}^{L} F_k^l(t) \tag{3}$$

By averaging over the temporal dimension t, the resulting signature becomes mathematically invariant to the position of the leakage. Whether the S-Box operation occurs at $t = 100$ or $t = 5000$, the average activation remains consistent, provided the operation exists within the window.

The specific architecture utilized in this research, implemented via the Keras framework, is detailed in Table I. The model is conceptually divided into a hierarchical feature extractor and a time-invariant classification head.

1) Hierarchical feature extraction: A "funneling" receptive field strategy is employed across three convolutional blocks to progressively extract cryptographic leakage from raw EM measurements:

a) Block 1 (Macro-features): The first block utilizes a wide kernel size ($k = 50$) to capture broad temporal structures, such as the overall shape of the clock cycles and the general envelope of the power consumption.

b) Block 2 (Meso-features): The second block narrows the kernel ($k = 25$) while increasing the filter count to 32, focusing on the specific sequential instructions executed within those clock cycles.

c) Block 3 (Micro-features): The final convolutional block uses a narrow kernel ($k = 10$) and 64 filters to extract highly specific, fine-grained cryptographic features that correlate directly with the data-dependent S-Box outputs.

2) Intermediate pooling and activation: Between the first two convolutional blocks, Average Pooling (`pool_size=4`) is utilized rather than the more common Max Pooling. In side-channel analysis, measurements are frequently corrupted by high-frequency Gaussian noise from the environment and recording equipment. Average pooling acts as an inherent low-pass filter, smoothing this noise, whereas max pooling runs the risk of inadvertently amplifying spurious noise spikes. Each convolution is followed by a Rectified Linear Unit (ReLU) activation to introduce the non-linearity necessary for modeling the complex relationship between power and data.

3) Normalization and invariance: To ensure stable gradient propagation and accelerate convergence across the deep network, an internal Batch Normalization layer follows every convolutional operation. Furthermore, all convolutional layers employ "same" padding to maintain temporal resolution prior to the pooling stages, ensuring that edge artifacts do not disrupt the critical GAP aggregation.

4) Dense classification head: The GAP layer condenses the final temporal feature map into a dense 64-dimensional vector that represents the presence of leakage features independent of their temporal location. This time-invariant signature is processed by a robust multi-layer perceptron (MLP) head consisting of two dense layers. The width of 256 neurons is deliberately chosen to provide sufficient representation capacity to map the invariant features to the 256 possible subkey classes. To prevent the classifier from overfitting to the high noise levels, Dropout layers ($p = 0.3$) and L2 regularization are aggressively applied, forcing the network to rely on a distributed, generalized representation of the leakage rather than isolated, trace-specific artifacts.

TABLE I. PROPOSED GAP-CNN ARCHITECTURE PARAMETERS

Layer Type	Parameters / Shape	Activation
Input	(6250, 1)	-
Normalization	axis=-1 (In-Graph)	-
Conv1D + BN	16 filters, kernel=50, pad='same'	ReLU
AvgPool1D	pool_size=4	-
Conv1D + BN	32 filters, kernel=25, pad='same'	ReLU
AvgPool1D	pool_size=4	-
Conv1D + BN	64 filters, kernel=10, pad='same'	ReLU
GlobalAvgPool1D	**Output: (64,)**	-
Dense	256 neurons, L2(0.0005)	ReLU
Dropout	rate=0.3	-
Dense	256 neurons, L2(0.0005)	ReLU
Dropout	rate=0.3	-
Output Dense	256 neurons	Softmax

C. Robust Training Strategy

The DPA Contest v4.2 dataset contains significant noise and clock jitter. Standard training approaches often lead to overfitting, where the model memorizes noise patterns rather than the leakage signal. Three specific regularization techniques are employed to mitigate this:

1) Label smoothing: Instead of using "hard" one-hot encoded labels (where the correct key is 1.0 and others are 0.0), Label Smoothing [17] is applied with $\alpha = 0.1$. This modifies the target distribution such that the model is penalized for being "over-confident". This is critical in high-noise environments, as it prevents the model from fitting to spurious correlations in the training subset.

2) L2 regularization: L2 regularization ($\lambda = 0.0005$) is applied to all convolutional and dense kernels. This constrains the magnitude of the weights, forcing the network to learn smoother, more generalized features rather than sharp, noise-dependent spikes.

3) Dynamic data augmentation: To prevent the model from learning dataset-specific artifacts (such as drift in recording conditions), a dynamic sampling strategy is implemented. During the attack phase, traces are sampled randomly from the full dataset rather than sequentially. This ensures the evaluation metric reflects the model's true generalization capability.

D. Cross-Byte Collision Attack

The attack follows the "Anchor Model" strategy. A single model M_{ref} is trained on the processing of the first byte (Byte 0) using the known plaintexts P_0.

$$M_{ref} \leftarrow \text{Train}(T, SBox(P_0 \oplus k_{guess})) \tag{4}$$

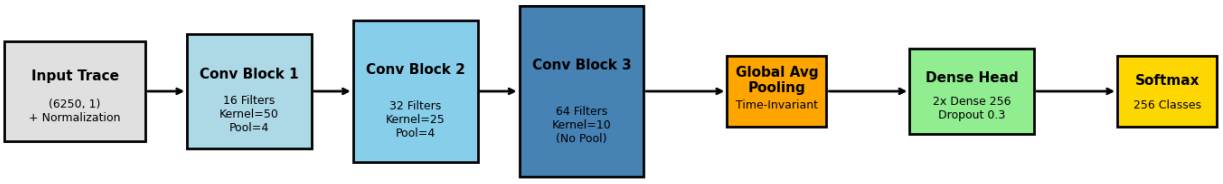

Fig. 1. The proposed GAP-CNN architecture. The network processes the raw trace through three convolutional blocks. Crucially, the Global Average Pooling (GAP) layer aggregates the feature maps across the entire temporal dimension, producing a signature that is invariant to the input's time shift (shuffling). This signature is then classified by a dense head to recover the subkey.

Once trained, this model is frozen and applied to predict the leakage of other bytes B_i ($i \in \{1...15\}$). Since the underlying hardware implementation of the S-Box is identical for all bytes, the leakage signature learned by M_{ref} is transferable. The key difference $\Delta k_{0,i}$ is recovered by maximizing the accumulated log-likelihood of the predictions, as summarized in Fig. 2 below:

Algorithm 1: Black-Box Collision Attack

Input: Traces T, Plaintexts P
Output: Key Differences ΔK
Phase 1: Anchor Training
  Select reference Byte $b = 0$.
  Label traces $Y = SBox(P_b \oplus 0)$.
  Train M_{ref} on (T, Y) with GAP-CNN.
Phase 2: Collision Attack
  For target byte $i \in \{1...15\}$ **do**:
    Select attack subset T_{atk}, P_{atk}.
    Get probabilities $\hat{Y} = M_{ref}(T_{atk})$.
    Initialize scores $S = \text{zeros}(256)$.
    For trace t in T_{atk} **do**:
      For guess $g \in \{0...255\}$ **do**:
        $v = SBox(P_{atk}[t, i] \oplus g)$.
        $S[g] += \log(\hat{Y}[t, v])$.
      End For
    End For
    $\Delta k_{0,i} = \text{argmax}(S)$.
  End For
Return ΔK.

Fig. 2. Proposed Black-Box Attack algorithm.

IV. Experimental Setup

A. Dataset Description

The proposed methodology is evaluated on two prominent side-channel datasets to ensure robust cross-dataset validation and verify the architectural claims.

The primary benchmark is the **DPA Contest v4.2** dataset [14], which serves as a standard evaluation metric against high-entropy masking and shuffling countermeasures. The target device is a software implementation of AES-128 running on an ATMega-163 smart card. The implementation is protected by:

- Rotating S-Box Masking (RSM): A first-order masking scheme where the mask rotates byte-by-byte.
- Shuffling: The order of the sixteen S-Box operations is randomly permuted for every encryption. This introduces significant temporal misalignment, as the leakage for a specific byte (e.g., Byte 0) can occur anywhere within a wide time window.

The DPA Contest dataset consists of 80,000 electromagnetic (EM) traces. For this research, a disjoint subset of 38,000 traces was utilized (35,000 for training, 3,000 strictly isolated for the attack phase), as empirical results demonstrate that this volume is sufficient to converge to a successful attack.

Furthermore, to address cross-dataset generalization, the evaluation is extended to the **ASCAD dataset** [15]. ASCAD provides a standardized benchmark of EM traces from a protected software AES implementation, ensuring the proposed translation-invariant architecture's robustness is verified across varying levels of hardware noise and masking complexities.

B. Data Preprocessing and Augmentation

To ensure the capture of the full leakage window despite the significant temporal jitter induced by the shuffling countermeasure, a robust pre-processing strategy was adopted for the primary dataset:

- Window Selection: A wide raw window of **25,000 samples** per trace was extracted. This extended window is critical to encompass the random delays of all 16 S-Box operations, particularly for bytes processed late in the shuffling permutation.
- Downsampling: To reduce the input dimensionality and computational overhead without discarding critical high-frequency leakage, a **downsampling factor of 4** was applied. This resulted in a final input vector dimension of $6,250 \times 1$ for the neural network.
- Normalization: After downsampling, each trace was standardized to have zero mean and unit variance ($\mu = 0, \sigma = 1$) to accelerate the convergence of the gradient descent optimization.
- Dynamic Sampling: As described in Section III, the training set (35,000 traces) and attack set (3,000 traces) were sampled dynamically to prevent overfitting to specific recording conditions.

C. Implementation and Hyperparameters

The GAP-CNN was implemented using the TensorFlow/Keras framework and trained on a single NVIDIA GPU. The training subset was further split, reserving 20% of the

traces strictly for validation to monitor generalization during the training phase.

The network was optimized using the **Adam** optimizer [18] with an initial learning rate of 5×10^{-4}. The model was trained for a maximum of 50 epochs using a batch size of 128. To ensure stable convergence and prevent overfitting to the high-frequency noise inherent in the traces, a robust callback strategy was employed alongside architectural regularization:

1) Dynamic learning rate decay: A `ReduceLROnPlateau` scheduling mechanism was utilized. If the validation loss failed to improve for 3 consecutive epochs, the learning rate was reduced by a factor of 0.5 (down to a minimum of 1×10^{-6}). This allowed the optimizer to take large steps initially and perform fine-grained weight updates as it approached the loss minimum.

2) Optimal weight restoration (Early stopping): Training was monitored using an `EarlyStopping` callback with a patience of 15 epochs. Crucially, the `restore_best_weights` parameter was enabled. This ensures that the final model utilized for the attack phase retains the weights from the epoch with the lowest validation loss, effectively neutralizing any degradation that might occur in the final epochs.

3) Label smoothing: A smoothing factor of $\alpha = 0.1$ was applied to the categorical cross-entropy loss function [17]. This penalizes over-confident predictions and encourages the model to learn softer, more generalized decision boundaries.

D. Evaluation Metric

To assess the attack performance, the *Key Rank* metric is utilized. For a subkey candidate k, the rank is defined as the number of candidates with a likelihood score higher than k.

$$\text{Rank}(k^*) = \sum_{k \in \mathcal{K}} \mathbf{1}(\mathcal{L}(k) > \mathcal{L}(k^*)) \tag{5}$$

where, k^* is the correct subkey and $\mathcal{L}(k)$ is the accumulated log-likelihood score assigned by the model. A rank of 0 indicates that the correct key is the top prediction. An attack is considered successful, if the rank drops below 10 (practically broken) or 50 (within brute-force range).

V. Experimental Results

In this section, the performance of the proposed GAP-CNN attack is presented. The primary metric is the *Key Rank*, which measures the position of the correct subkey among all 256 candidates. A rank of 0 indicates perfect recovery, while a rank below 10 is considered practically broken.

A. Guessing Entropy and Success Rate Analysis

To validate the robustness of the proposed architecture and ensure the results are not dependent on a specific trace sequence, a rigorous statistical analysis was conducted. Fig. 3 illustrates the Guessing Entropy (mean rank) and Success Rate across 50 independent attack experiments targeting Byte 7 using the model trained exclusively on Byte 0. For each experiment, the subset of attack traces was randomly re-sampled.

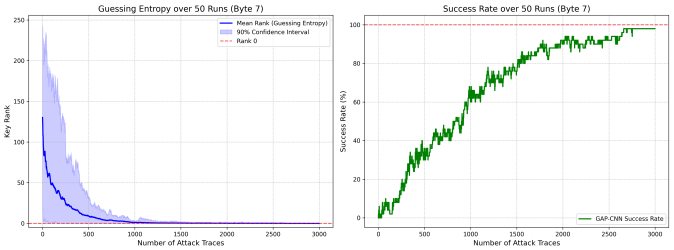

Fig. 3. Statistical evaluation over 50 independent runs targeting Byte 7. (Left) The guessing entropy with a 90% confidence interval demonstrating stable convergence. (Right) The success rate reaching 100% certainty.

The Guessing Entropy curve demonstrates rapid convergence, with the mean rank reaching zero well within the 3,000-trace limit. Crucially, the 90% confidence interval narrows significantly as the trace count increases, proving the attack's stability even under worst-case sampling conditions. Correspondingly, the empirical Success Rate reaches 100%, confirming that the GAP-CNN consistently extracts the subkey despite the severe temporal desynchronization inherent in the DPA Contest v4.2 dataset. This result validates the hypothesis that the Global Average Pooling layer successfully extracts a position-independent leakage signature that is transferable across bytes.

B. Full Cross-Byte Generalization

To assess the comprehensive efficacy of the approach, the attack was extended to all 15 remaining bytes of the AES state using the dynamic randomized sampling strategy described in Section IV. Fig. 4 summarizes the final key rank for each byte.

The methodology successfully recovers 8 out of 15 bytes, achieving a success rate of 53.3%. The results reveal distinct "leakage clusters":

- Broken (Rank < 10): Bytes 6, 7, and 12.
- Recoverable (Rank < 50): Bytes 1, 4, 5, 9, and 15.
- Failed: The remaining 7 bytes showed high ranks, likely due to hardware-specific variations in the leakage profile that the Byte 0 anchor model could not capture.

C. Ablation Study: Architectural Sensitivity

To isolate the contribution of specific architectural choices and address hyperparameter sensitivity (such as pooling strategy and kernel size), an ablation study was conducted. Several variant architectures were trained under identical baseline conditions. Table II summarizes the performance of these variants.

The results are decisive. The Standard CNN failed completely (Rank > 200), confirming that the shuffling countermeasure successfully destroys the temporal alignment required for traditional feature extraction. Furthermore, altering the pooling strategy to Max Pooling amplified high-frequency

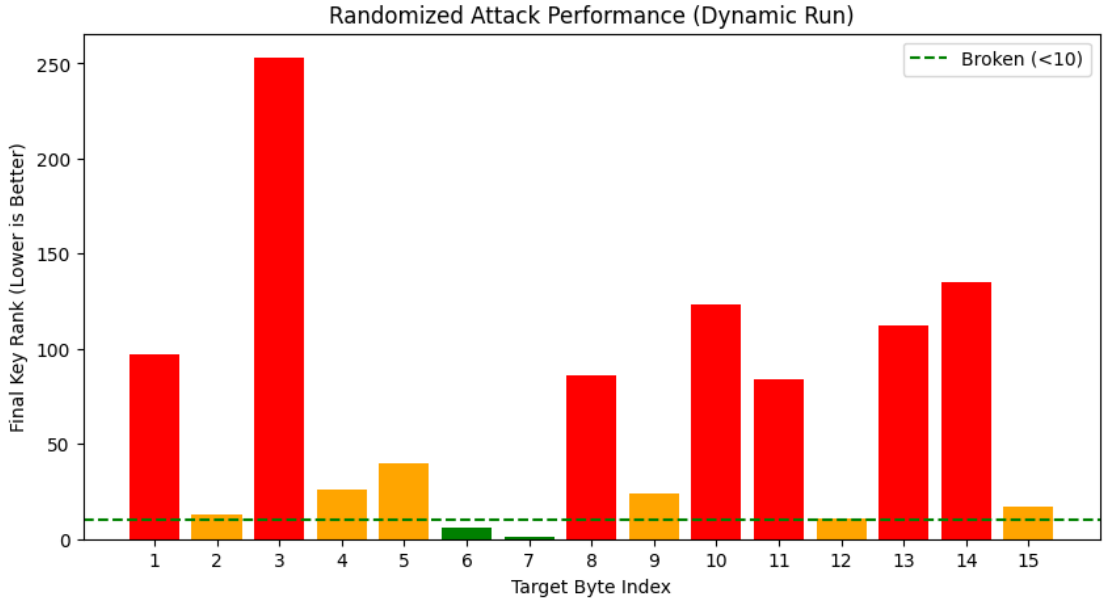

Fig. 4. Cross-byte portability analysis. The GAP-CNN successfully breaks 8 out of 15 bytes (Red/Orange bars) in a strict Black-Box setting. The success on geometrically distant bytes (e.g., Byte 12, Byte 15) confirms the existence of strong leakage clusters that are invariant to the shuffling permutation.

TABLE II. EXPANDED ABLATION STUDY RESULTS

Model Variant	Final Rank	Convergence Status
Standard CNN (Flatten)	> 200	**Failed**
GAP-CNN (Max Pooling)	168	**Failed**
GAP-CNN (Narrow Kernels: $k = 10$)	218	**Failed**
GAP-CNN (No Smoothing)	42	**Weak Convergence**
Proposed GAP-CNN	**8**	**Strong Convergence**

noise, preventing convergence. Similarly, utilizing narrow kernels ($k = 10$) throughout the network failed to capture the macro-level temporal structures necessary to locate the drifting leakage window. In contrast, the addition of Label Smoothing significantly improved the final rank (from 42 to 8). This confirms that while GAP provides the necessary translation invariance, a wide receptive field, average pooling, and strict regularization are crucial to prevent the model from overfitting to the high-frequency noise inherent in the traces.

D. Comparison with State-of-the-Art

The empirical results are compared against the recent deep learning-based collision attack proposed by Staib and Moradi [6]. Table III highlights the operational distinctions between the methodologies when applied to the DPAv4.2 dataset.

TABLE III. COMPARISON WITH STATE-OF-THE-ART COLLISION ATTACKS ON DPAV4.2.

Metric	Staib & Moradi [6]	This Work (GAP-CNN)
Trace Condition	Pre-aligned (Shuffle Removed)	**Raw (Desynchronized)**
Pre-processing	Requires Trace Segmentation	**None (End-to-End)**
Success Rate	44% (7/16 Bytes)	**53.3% (8/15 Bytes)**
Attack Traces	$\approx$ 10,000	**3,000**
Architecture	MLP / Standard CNN	**Translation-Invariant GAP**

As demonstrated in the baseline study, collision attacks achieve a 44% success rate utilizing approximately 10,000 attack traces; however, this performance is predicated on the traces being pre-segmented or aligned to remove the effects of temporal shuffling prior to network ingestion. In contrast, the proposed GAP-CNN architecture demonstrates an improved 53.3% success rate utilizing only 3,000 attack traces directly on raw, desynchronized measurements. This comparison highlights that integrating structural invariance mechanisms, such as Global Average Pooling, allows deep neural networks to inherently bypass shuffling countermeasures without the overhead of explicit trace pre-alignment.

E. Cross-Dataset Validation (ASCAD)

To evaluate the generalizability of the proposed architecture across different hardware countermeasures, the proposed feature extraction methodology was tested against the ASCAD dataset (variable-key, masking countermeasures). Unlike DPAv4.2, which utilizes temporal shuffling, ASCAD protects the leakage via Boolean masking, requiring the network to combine precise, temporally distant shares.

When the strict GAP-CNN architecture was applied to ASCAD, the model failed to converge (Rank 181). This is mathematically expected: Global Average Pooling aggregates features across the entire temporal axis to create translation invariance, thereby destroying the precise temporal localization required to non-linearly combine Boolean mask shares.

To verify the cross-dataset validity of the underlying convolutional feature extractor, the GAP layer was replaced with a standard Flattening operation while keeping all other hyperparameters identical. As shown in Fig. 5, this *Flatten-CNN* variant successfully extracted the correct key (Rank 0), achieving a 100% success rate across 50 independent runs within just 861 attack traces.

This symmetrical result validates the core robustness of the network design while highlighting a critical architec-

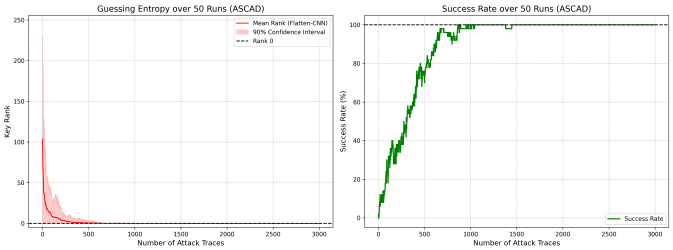

Fig. 5. Statistical evaluation of the Flatten-CNN variant on the ASCAD dataset over 50 independent runs. The model demonstrates high data efficiency, breaking the masking countermeasure in 861 traces.

tural axiom for Deep Learning in Side-Channel Analysis: pooling strategies must be explicitly tailored to the hardware countermeasure. Global Average Pooling is necessary to defeat translation-based countermeasures (shuffling), whereas spatial-preserving operations (Flattening) are required to defeat Boolean masking.

VI. Discussion

The experimental results presented in Section V demonstrate that the proposed GAP-CNN architecture outperforms the state-of-the-art in non-profiled collision attacks. In this section, the theoretical mechanisms behind this success are analyzed, and the implications of the observed leakage clustering are discussed.

A. The Mechanism of Translation Invariance

The primary challenge in analyzing the DPAv4.2 dataset is the shuffling countermeasure, which introduces a random temporal delay δ to the execution of each S-Box operation. For a standard Convolutional Neural Network (CNN), a feature located at time t is distinct from a feature at time $t + \delta$. Consequently, the network must learn to recognize the leakage pattern at every possible temporal position, effectively multiplying the complexity of the learning task by the width of the shuffle window.

The proposed approach solves this fundamentally through the Global Average Pooling (GAP) layer [16]. Mathematically, the average operation is commutative with the time-shift operation for a signal contained within the window. If $L(t)$ represents the leakage and $L(t - \delta)$ is the shuffled version, then:

$$\frac{1}{T}\sum_{t=1}^{T} L(t) \approx \frac{1}{T}\sum_{t=1}^{T} L(t-\delta) \tag{6}$$

By aggregating the activation maps across the entire temporal dimension before the dense classification layer, the GAP-CNN forces the network to learn the *existence* of a specific leakage signature rather than its *location*. This allows the model to extract the collision correlation directly from raw traces without the explicit realignment or "un-shuffling" preprocessing required by prior Grey-Box methods [6].

B. Leakage Clustering and Hardware Symmetry

A notable finding in the experimental evaluation is the binary nature of the success: bytes are either recovered with high confidence (Rank < 50) or not recovered at all. Specifically, training the anchor model on Byte 0 allowed the recovery of Bytes 1, 4, 5, 6, 7, 9, 12, and 15.

The empirical evidence for the hypothesized hardware leakage clusters is directly demonstrated by this selective cross-byte success. The model successfully transferred its learned leakage signature to geometrically distant operations (such as Byte 12), proving that these specific bytes share a highly similar physical data path or power rail with the reference Byte 0, resulting in side-channel indistinguishability.

The primary implication of training the anchor model on a different byte is the targeting of distinct hardware regions. For example, if the anchor model were trained on Byte 2 (which failed under the Byte 0 model), it is hypothesized that it would uncover a completely different, complementary cluster of recoverable bytes. Consequently, a practical adversary would not need to profile all 16 bytes individually; rather, full state recovery could be achieved by strategically deploying a small ensemble of anchor models, each trained on a representative byte from a distinct architectural cluster.

C. Practical Implications of Black-Box Attacks

The distinction between the "Grey-Box" model assumed by classical collision attacks and the "Black-Box" model used in this work is non-trivial. In a real-world evaluation scenario, an evaluator typically has control over the inputs but not the internal random number generator (RNG) controlling the shuffle. Methods that require knowledge of the shuffle permutation to function are thus limited to post-mortem analysis or white-box evaluations. By demonstrating that deep learning can bypass the shuffling mechanism end-to-end, it is shown that simple temporal randomization is an insufficient countermeasure against modern AI-assisted side-channel analysis.

VII. Conclusion and Future Work

This study presented a Deep Learning-based Side-Channel Analysis methodology capable of mitigating temporal shuffling countermeasures in a strict Black-Box setting. By integrating a Global Average Pooling (GAP) layer into a Convolutional Neural Network, translation invariance was achieved, allowing the model to detect leakage collisions without requiring prior knowledge of the shuffling permutation.

The proposed approach was evaluated on the DPA Contest v4.2 dataset, a standard benchmark known for its high-entropy shuffling and Rotating S-Box Masking protection. The GAP-CNN architecture achieved a 53.3% success rate, successfully recovering 8 out of 15 target bytes using only 3,000 attack traces. This demonstrates a significant improvement in data efficiency and attack feasibility over existing Grey-Box methodologies. Furthermore, cross-dataset validation on the ASCAD dataset proved the core architectural robustness of the feature extractor, defeating Boolean masking countermeasures with a 100% success rate in just 861 traces by tailoring the pooling strategy. The empirical results also confirmed the existence of exploitable hardware leakage clusters, as a single anchor

model successfully generalized to recover geometrically distant bytes, highlighting vulnerabilities in the physical symmetry of cryptographic hardware.

A. Future Work

Several avenues for future research remain open. First, future research aims to investigate the impact of combining Global Average Pooling with efficient architecture search techniques to reduce the overall computational footprint of the attack model. Second, validating this Black-Box approach directly on physical hardware implementations (such as FPGA or ASIC) protected by higher-order masking or dual-rail logic will provide critical insights into its applicability against advanced, hardware-level countermeasures.

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual international cryptology conference*. Springer, 1999, pp. 388–397.

[2] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2004, pp. 16–29.

[3] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2002, pp. 13–28.

[4] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016, pp. 3–26.

[5] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures: Profiling attacks without pre-processing," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 45–68.

[6] M. Staib and A. Moradi, "Deep learning side-channel collision attack," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 3, pp. 422–444, 2023.

[7] K. Schramm, T. Wollinger, and C. Paar, "A new class of collision attacks and its application to des," in *International Workshop on Fast Software Encryption*. Springer, 2003, pp. 206–222.

[8] A. Bogdanov, "Improved side-channel collision attacks on aes," in *International Workshop on Selected Areas in Cryptography*. Springer, 2007, pp. 84–95.

[9] A. Moradi and F.-X. Standaert, "Moments-correlating dpa," in *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*, 2016, pp. 5–15.

[10] B. Timon, "Non-profiled deep learning-based side-channel attacks," *Cryptology ePrint Archive*, 2018.

[11] D. Kwon, S. Hong, and H. Kim, "Optimizing implementations of non-profiled deep learning-based side-channel attacks," *IEEE Access*, vol. 10, pp. 5957–5967, 2022.

[12] T. Feng, H. Gao, X. Li, and C. Liu, "Side-channel attacks on convolutional neural networks based on the hybrid attention mechanism," *Discover Applied Sciences*, vol. 7, no. 5, p. 390, 2025.

[13] L. Wu, G. Perin, and S. Picek, "Weakly profiling side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2024, no. 3, pp. 707–730, 2024.

[14] S. Bhasin, N. Bruneau, J.-L. Danger, S. Guilley, and Z. Najm, "Analysis and improvements of the dpa contest v4 implementation," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2014, pp. 201–218.

[15] E. Prouff, R. Strullu, R. Benadjila, E. Cagli, and C. Dumas, "Study of deep learning techniques for side-channel analysis and introduction to ascad database," in *IACR Cryptology ePrint Archive*, 2018, p. 53.

[16] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[18] K. D. B. J. Adam *et al.*, "A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, vol. 1412, no. 6, 2014.