ANALYZING MASK-BREAKING MECHANISMS: A METHODOLOGY FOR DEEP LEARNING SIDE-CHANNEL ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Side-channel analysis (SCA) of cryptographic implementations remains a persistent threat, necessitating the certification of new products by independent evaluation labs. In recent years, deep learning-based SCA (DLSCA) has become a standard approach for worst-case evaluations, due to its ability to defeat diverse countermeasures. While achieving state-of-the-art performance, this comes at the cost of the evaluator's understanding of the attacks themselves.

In this work, we propose a novel methodology based on mechanistic interpretability to understand how neural networks recombine shares to defeat the masking countermeasure. Concretely, to isolate the role of individual layers, we freeze the first few layers of a trained model and retrain the remainder while injecting ground-truth information about specific secret shares. This allows us to determine where exploitable share-related information is represented in the network and how it can be used.

To demonstrate our approach, we analyze EffCNN and EstraNet on ASCADv1f and ASCADv1r, and EffCNN on the non-shuffled ESHARD benchmark. We find that injecting clean information about one share reveals which information about the other share remains exploitable by the model, highlighting that the models represent substantially more information about individual shares than they originally exploit. Furthermore, our method confirms that the organization of share-related information within the models follows the physical leakage characteristics, without requiring the manual effort of previous approaches and thereby allowing interpretability analyses to scale to more realistic models. Overall, our work suggests that injecting ground-truth share information is a promising direction for interpreting DLSCA models, going beyond prior approaches by revealing not only where share information is represented but also how much of it remains unexploited.

1 INTRODUCTION.

Side-channel analysis (SCA) represents a critical threat to the security of cryptographic implementations. SCA consists of diverse methods characterized by the extraction of unintended (physical) leakage, e.g., power consumption or electromagnetic radiation, generated by an electronic device while operating Kocher (1996); Kocher et al. (1999). These signals can carry noisy information on the internally processed data. This poses a realistic threat to device security, as effective analysis of these signals could allow an attacker to recover sensitive information Roche et al. (2021).[1]

As such, certification schemes often require devices to be evaluated against SCA ISO/IEC 15408-1:2022. A common, worst-case approach to upper-bounding attacker capabilities is to assume an attacker has full access to a clone device Chari et al. (2003); Masure et al. (2023). This then allows the attacker to first 'profile' the leakage signatures on the copy device, thereby more efficiently extracting the secret information from the target. In this setting, DLSCA has received significant attention from both researchers Maghrebi et al. (2016); Zaid et al. (2020); Perin et al. (2022); Picek et al. (2023); Karayalcin et al. (2026) and regulatory bodies Federal Office for Information Security (BSI) (2024).

[1]For example, the private keys of an IoT device, neural network model weights, and, in general, any intellectual property.

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.

However, the success of DLSCA in attacking cryptographic implementations protected by masking and hiding countermeasures comes at the cost of a precise understanding of the attack mechanisms Picek et al. (2023); Karayalcin et al. (2026). A successful attack resulting in a failed evaluation is substantially harder to address without an explanation of how the implementation is exploited, i.e., which secret information is being extracted from where Gao & Oswald (2024). As such, several approaches have been proposed to explain neural network decisions, e.g., input attribution methods Masure et al. (2019); Yap et al. (2024); Gammell et al. (2026), alternative architectures Zaid et al. (2023); Yap et al. (2023); Yoshida et al. (2025), and recently, mechanistic interpretability Karayalçin et al. (2025); Perin et al. (2026). In particular, Karayalçin et al. (2025) provides a precise understanding of what specific leakage is being exploited, and this relies on manual analysis of small multilayer perceptrons (MLPs). Scaling similar analyses to larger models, such as those studied in Zaid et al. (2020); Hajra et al. (2023), remains challenging.

In this work, we aim to bridge the gap in understanding how a deep learning model used in SCA classifies sensitive information from a masked cryptographic implementation. More precisely, we target implementations protected by first-order Boolean masking and investigate how a DL model implicitly defeats the countermeasure to perform an attack. Our main contributions are as follows:

- We introduce the *Mask Injection Mechanism*, a novel analytical framework for investigating how semantic information about the mask is processed across network layers. By injecting ground-truth mask information into specific activation maps and retraining subsequent layers, we can trace the model's dependency on mask-specific features.
- We perform an extensive analysis using both convolutional and attention-based DL models across three benchmark datasets characterized by first-order Boolean masking. This provides a comprehensive view of how different DL-SCA models handle masking countermeasures under different leakage scenarios.
- We provide empirical evidence that the tested convolutional and transformer models contain more mask- and share-related information than the original models exploit. We further show that our injection methodology enables the downstream network to use this information to improve attacks. Our analysis reveals that, on ASCAD, the tested models learn to identify and use the most informative bits of both the mask and the masked intermediate value, whereas, on ESHARD, the convolutional model focuses on the Hamming weight of the mask. These findings are consistent with a staged organization in which early and intermediate representations encode share-related information that can subsequently be combined by downstream computation.

2 Background.

2.1 Deep Learning-based Side-channel Analysis.

Side-channel analysis (SCA) targets weaknesses in cryptographic implementations rather than in the underlying algorithms. More precisely, side-channel attacks are passive, non-invasive attacks that exploit unintended leakage from cryptographic implementations Kocher (1996). The principle behind these attacks is that secret-dependent operations in physical implementations of cryptographic algorithms influence measurable side channels (e.g., power consumption or electromagnetic emanations). In SCA, it is common to divide the attacks into direct attacks and profiling attacks.

Direct attacks consider only the device under attack and measure a large number of traces containing some intermediate operation that depends on a known input variable and (part of) the key. To extract the key, the attacker then enumerates the hypothetical intermediate values generated for each possible key candidate. The key candidate that best matches the measured leakage is then assumed to be the correct key. Profiling attacks assume that the attacker has access to a clone device. With that clone device, the attacker models the secret-dependent leakage in the profiling phase and then, in the attack stage, uses this model to obtain the secret key from the target Chari et al. (2003).

To assess the attack performance, common side-channel metrics are key rank, guessing entropy, and success rate Standaert et al. (2009). Key rank is simply the position of the correct key in the key guessing vector that is ordered from the most likely to the least likely key guess. The smaller the key rank, the better the attack is. Guessing entropy is the average key rank, where we average

performance over different sets of attack traces to reduce the variability stemming from selecting specific traces.

Deep learning-based SCA uses deep neural networks to build profiling models. More formally, DL-SCA learns parameters θ for a model $f_\theta : X \mapsto Y$ that extracts a probability distribution over the targeted intermediate value from the traces. These parameters are optimized on the training set, i.e., the measurements taken from the profiling device. To attack the target, the attacker obtains measurements from the target device and uses the trained model to extract the key by combining predictions across multiple traces into a key-guessing vector. For further details on DL-SCA, we refer readers to Picek et al. (2023); Karayalcin et al. (2025).

To protect implementations against side-channel attacks, countermeasures such as random shifting, random clock jitter, and masking can be implemented. These countermeasures aim to weaken the relationship between physical leakage and the data processed by the algorithm. Masking, on which we focus in this work, accomplishes this by splitting computations on sensitive values into d shares using internal randomness. This ensures that there is no direct computation on—and consequently no direct leakage of—the sensitive intermediate values.

2.2 Mechanistic Interpretability.

Mechanistic interpretability (MI) is a field that aims to reverse-engineer the internal computations performed by neural networks into human-understandable algorithms Olah et al. (2017; 2020). The underlying hypothesis is that neural networks internally represent *features* corresponding to input concepts as linear directions in high-dimensional activation spaces Park et al. (2024). These features then compose into *circuits*, i.e., algorithms implemented by the network Olah et al. (2020). Features in early layers are often low-level attributes of the input. In vision classification models, neurons in early convolutional layers have been shown to detect curves Cammarata et al. (2020) or high versus low frequencies Schubert et al. (2021). Neurons in later layers often correspond to more abstract concepts, such as a car-detector neuron Olah et al. (2020).

While features sometimes correspond to specific neurons, a more general view is that they correspond to linear directions in the activation space Park et al. (2024). Training linear classifiers (probes) on labeled activations is then a straightforward way to test whether models internally represent a feature Alain & Bengio (2017). Finally, model computations can be causally affected in predictable ways by directly intervening on directions corresponding to interpretable features Templeton et al. (2024); Arditi et al. (2024).

3 Mask Injection Mechanism.

When a deep learning model is used to attack traces from a masked implementation, it must indirectly infer information about the secret shares and recombine them to obtain the intermediate value Perin et al. (2026); Masure et al. (2023). Previous work indicates that the earliest layers first extract information about the shares, which is subsequently recombined across layers before the model refines its prediction Perin et al. (2026). While previous studies show which parts of a share are directly used by the model, they do not necessarily reveal whether additional potentially exploitable information about individual shares remains available in specific layers.

To test whether additional information is available, we define a methodology for studying how the model analyzes the secret shares while processing each trace. We inject clean information about the mask at each layer, freeze all preceding layers, and reinitialize and retrain the remainder of the network. Freezing preserves the features learned by the preceding layers, while reinitialization allows the downstream network to escape the previous minimum. To inject this information into the activation map ($x^{k,c,n}$, where k, c, and n denote the batch, feature, and sample dimensions, respectively), we add $\Delta x^{k,c,n}$ as follows:

$$\Delta x^{k,c,n} = f_{SiLU}\left(\frac{1}{\sqrt{|p|}}\sum_{p} Q^{c,n,p} K^{c,p,m_k}\right). \tag{1}$$

This follows a query-key scheme in which we define a matrix $K^{c,p}$ for each mask value m_k, similarly to word2vec (W2V), and a query tensor $Q^{c,n,p}$. Here, p is a latent dimension and $|p|$ is its cardinality.

We use f_{SiLU} Zhang et al. (2024) as the nonlinear function. Its continuous differentiability facilitates training in noisy scenarios Shah & Youngblood (2025). Applying the injection to an activation map causes the downstream network to receive $\chi^{k,c,n} = x^{k,c,n} + \Delta x^{k,c,n}$ as input. Thus, the following layers receive their original input together with additional semantic information about one of the secret shares. Although the injected mask values are exact, their mapping into the activation space is learned from randomly initialized parameters. Consequently, early in retraining, $\Delta x^{k,c,n}$ can act as a perturbation that degrades useful information already represented in $x^{k,c,n}$. The injection becomes beneficial only when its mask-dependent information outweighs this initial disruption; our experiments show that this generally occurs for informative injections at early and intermediate layers.

This mechanism creates a different K for each of the 256 possible values the secret share can take. The idea is then that any information the model has about the other share is now much easier to use. Consequently, the injections of the mask values become organized according to whatever is useful to exploit the other share, which allows us to infer how this other share is represented in the specific layer. However, learning 256 separate embeddings has a drawback. Each embedding is only updated by traces with the corresponding mask value, roughly $1/256$ of the training set, and at initialization the randomly initialized embeddings add substantial noise to the activations. We therefore also consider a simplified variant (1V), in which a single trainable key $K^{c,p}$ is shared across all mask values. This key defines one direction in the latent space, and the mask value only determines its magnitude through a scalar $g_{\text{mode}}(m_k)$:

$$\Delta x^{k,c,n} = f_{SiLU}\left(\frac{1}{\sqrt{|p|}}\sum_{p} Q^{c,n,p}K^{c,p} \cdot g_{\text{mode}}(m_k)\right). \tag{2}$$

For g_{mode}, we use the normalized identity (ID), Hamming weight (HW), or least significant bit (LSB) of the mask value, following common leakage models in SCA Picek et al. (2023). Unlike W2V, this variant imposes a leakage model but requires far fewer parameters to learn. Notably, if one choice of g_{mode} works substantially better than the others, this may indicate which representation is most closely aligned with the actual leakage model.

4 Related Work.

Early explainability work in DLSCA focused largely on input-attribution methods that highlight which parts of a trace contribute to network predictions Masure et al. (2019); Hettwer et al. (2020); Yap et al. (2024); Gammell et al. (2026). Another line of work focused on training inherently more interpretable architectures Yap et al. (2023); Zaid et al. (2023); Yoshida et al. (2025); Boussam et al. (2025) or layers that explicitly encode masking recombination Cao et al. (2022); Masure et al. (2023). Input-attribution methods do not explain how neural networks internally defeat countermeasures, while more interpretable architectures often require significant additional a priori assumptions and may reduce performance.

More recently, several works on mechanistic interpretability have emerged in SCA. First, Perin et al. Perin et al. (2026) showed that neural networks trained against masked implementations form an information bottleneck that compresses irrelevant information in early layers. Furthermore, they attempted to provide a precise understanding of how masks are recombined across layers using activation patches. In follow-up work, Karayalç et al. Karayalçin et al. (2025) showed that the discrete jumps in performance initially observed in Masure et al. (2023) correspond to the network learning to embed the leakage from secret shares. They then used this finding to extract the secret shares directly from the network activations.

These works improved our understanding of learning dynamics and feature learning in DLSCA, but precisely how secret shares are recombined remains an open question. While Perin et al. (2026); Karayalçin et al. (2025) provide a layer-wise understanding of where shares are embedded, they do not examine the specific mechanisms responsible for recombination. Understanding these mechanisms is not merely academic; it is crucial for mitigating DLSCA attacks and implementing countermeasures that target neural-network weaknesses. Our work contributes to this direction by considering not only the information exploited by the original networks, but also additional potentially exploitable information extracted in early layers but not directly used by the models.

5 EXPERIMENTAL RESULTS.

We analyzed the EffCNN architecture Zaid et al. (2020), as a standardized CNN commonly used in DLSCA, and EstraNet, another state-of-the-art DL model for SCA based on a transformer architecture.

The datasets used in this study are ASCADv1f, ASCADv1r and ESHARD (see Appendix A). We use Guessing Entropy (GE), a standard attack metric that indicates the rank of the correct key averaged over random samples of attack traces, to evaluate the model's effectiveness at each epoch. This metric allowed us to select the best model checkpoint, i.e., the one that obtained the lowest number of traces needed to reach GE equal to 0. We trained multiple independent runs for each model–dataset setting and retained successful runs according to the criterion defined in Appendix D.4; the number of retained runs varies across settings because of computational cost and training variability.

For all mask-injection experiments (Section 3), we set the latent dimensionality to $|p| = 30$. At initialization, the resulting W2V key vectors K^{c,p,m_k} for the 256 mask values are approximately equidistant in latent space. The training settings followed the baseline settings in the corresponding papers Zaid et al. (2020); Hajra et al. (2023). For each training run, we retained the checkpoint with the best attack performance and used these selected checkpoints in all subsequent comparisons.

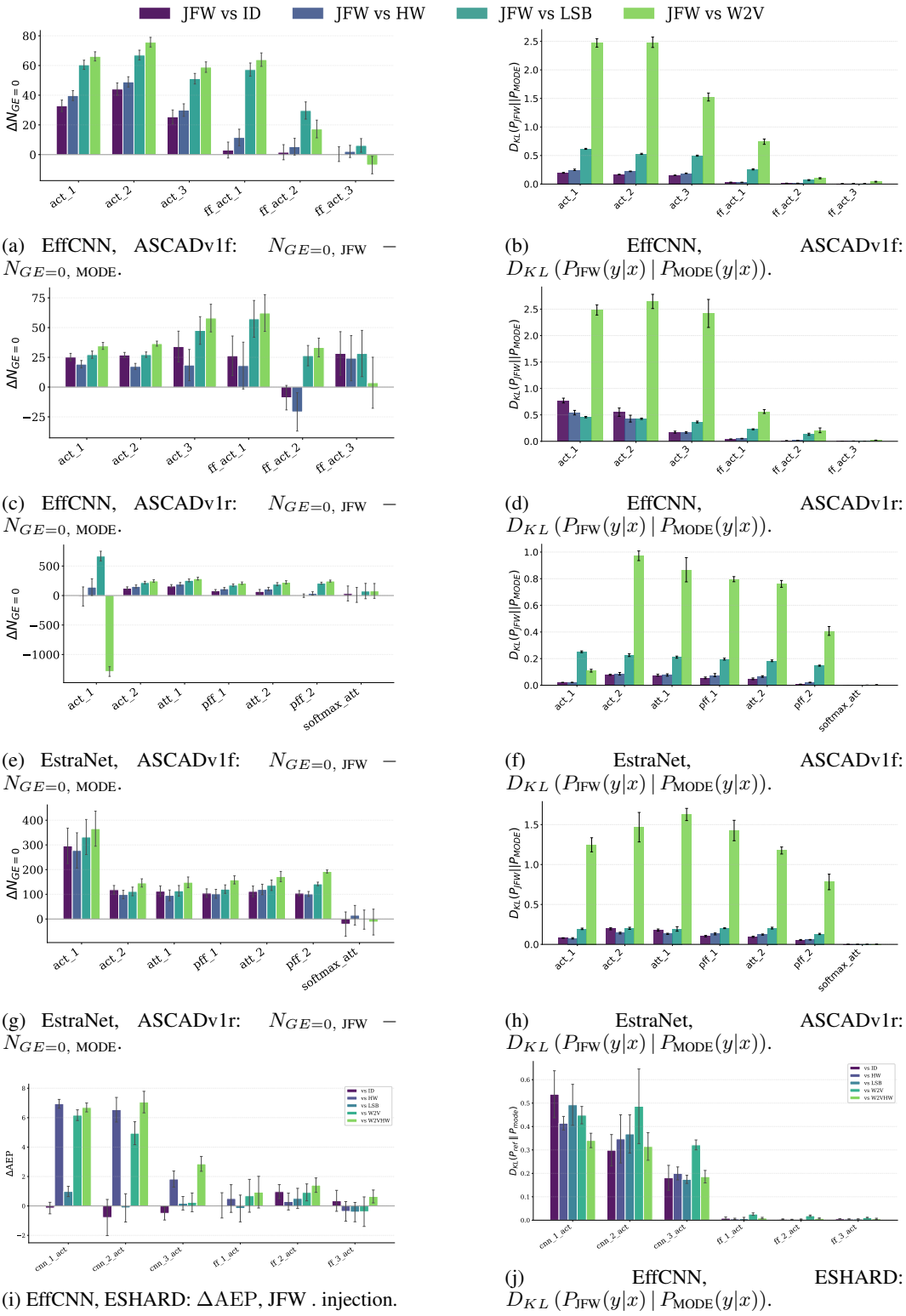
5.1 INJECTION CONSEQUENCES.

We analyzed reference models trained on the *ASCADv1f*, *ASCADv1r*, and *ESHARD* datasets and integrated mask information using our mechanism. We define JFW as the model that undergoes only freezing and reinitialization. For both *EffCNN* and *EstraNet*, each layer was selected as an injection point for the ID, HW, LSB, and W2V injection modes. Appendix D reports additional comparisons against RM (random-mask injection) and REF (the original model without injection). These baselines help distinguish improvements produced by informative mask injection from those caused by adding trainable parameters or retraining the downstream network. To quantify the effect of injection on attack performance, we calculated the difference in the number of traces required to perform the attack ($\Delta N_{GE=0} = N_{GE=0,\text{JFW}} - N_{GE=0,\text{MODE}}$), together with the Kullback–Leibler (KL) divergence between the output probability distributions of JFW and each injection mode (ID, HW, LSB, and W2V). The results for both architectures and all datasets are presented in Figure 1.

Layer-Wise Patterns. The clearest early-layer regime appears in EffCNN, for which the largest changes in both attack performance and D_{KL} occur in the convolutional activations (Figure 1). EstraNet exhibits a less localized pattern, with substantial injection effects extending from its convolutional layers into the attention and point-wise feed-forward blocks. At these injection points, explicit mask information remains useful to the retrained downstream network, consistent with the frozen representations retaining complementary information about the masked intermediate value.

A notable exception occurs in the first layers of EstraNet on ASCADv1f. The reference model is already a comparatively weak attacker, requiring nearly the complete 2,000-trace evaluation budget to reach $GE = 0$ (Appendix Figure 14). Although the injected mask values are correct, their mapping into the activation space is controlled by randomly initialized parameters; at the beginning of retraining, the injection therefore behaves as an additional noise-like perturbation. When introduced into these early representations, this perturbation appears to make optimization too difficult for most injection variants, preventing them from recovering a successful attack. LSB remains beneficial, and HW does so to a lesser extent. Both use the lower-capacity 1V parameterization, which shares a single key across mask values and may therefore introduce less initialization noise than W2V. The advantage of LSB is also consistent with the later clustering analysis, in which b_0 is among the bits most strongly associated with the learned embedding geometry (Section 5.2). We present these observations as a plausible explanation rather than as causal evidence for the optimization failure.

We hypothesize that the relatively homogeneous injection effects across EstraNet's attention block arise from its residual architecture. Each attention or point-wise feed-forward sublayer adds an update to a shared residual stream, $x' = x + f(x)$, rather than replacing its input representation. Information injected at any point in this sequence can therefore persist in the residual stream as subsequent updates are added, making the downstream effect less dependent on the precise injection point. The observed homogeneity continues until the softmax-attention layer, which marks the end of



(a) EffCNN, ASCADv1f: $N_{GE=0,\text{ JFW}} - N_{GE=0,\text{ MODE}}$.

(b) EffCNN, ASCADv1f: $D_{KL}\left(P_{\text{JFW}}(y|x) \mid P_{\text{MODE}}(y|x)\right)$.

(c) EffCNN, ASCADv1r: $N_{GE=0,\text{ JFW}} - N_{GE=0,\text{ MODE}}$.

(d) EffCNN, ASCADv1r: $D_{KL}\left(P_{\text{JFW}}(y|x) \mid P_{\text{MODE}}(y|x)\right)$.

(e) EstraNet, ASCADv1f: $N_{GE=0,\text{ JFW}} - N_{GE=0,\text{ MODE}}$.

(f) EstraNet, ASCADv1f: $D_{KL}\left(P_{\text{JFW}}(y|x) \mid P_{\text{MODE}}(y|x)\right)$.

(g) EstraNet, ASCADv1r: $N_{GE=0,\text{ JFW}} - N_{GE=0,\text{ MODE}}$.

(h) EstraNet, ASCADv1r: $D_{KL}\left(P_{\text{JFW}}(y|x) \mid P_{\text{MODE}}(y|x)\right)$.

(i) EffCNN, ESHARD: ΔAEP, JFW . injection.

(j) EffCNN, ESHARD: $D_{KL}\left(P_{\text{JFW}}(y|x) \mid P_{\text{MODE}}(y|x)\right)$.

Figure 1: Analysis of the consequences of mask injection on the attacking capabilities and output probability distributions of EffCNN and EstraNet on ASCADv1f and ASCADv1r, and of EffCNN on ESHARD. In each row, the change in attack performance (left), measured using $N_{GE=0}$ for ASCAD and AEP for ESHARD, is paired with its corresponding KL divergence (right).

this residual sequence. This provides a plausible architectural explanation for the pattern, although our experiments do not causally establish that the residual connections are responsible for it.

Fixed Leakage-Model Injections. The effectiveness of fixed leakage-model injections depends on how well the injected values match the physical leakage. On ASCAD, LSB injection generally yields larger improvements than HW and ID injection, consistent with the bitwise leakage indicated by the probes and correlation analyses in Appendix F. LSB also follows a layer-wise trend similar to W2V: its benefit is most apparent at early and intermediate injection points and decreases toward the output. This suggests that a low-complexity mask feature can provide much of the downstream utility of the more flexible W2V representation, provided that it is aligned with the leakage available in the frozen representation. Conversely, on ESHARD, HW injection clearly outperforms the other fixed leakage models and even W2V. This is surprising because W2V learns the HW structure (Figure 2) yet still underperforms direct HW injection. W2VHW closes this gap by constraining W2V to a single vector per HW value, thereby mitigating some of the optimization issues discussed above.

5.2 Mask Clustering.

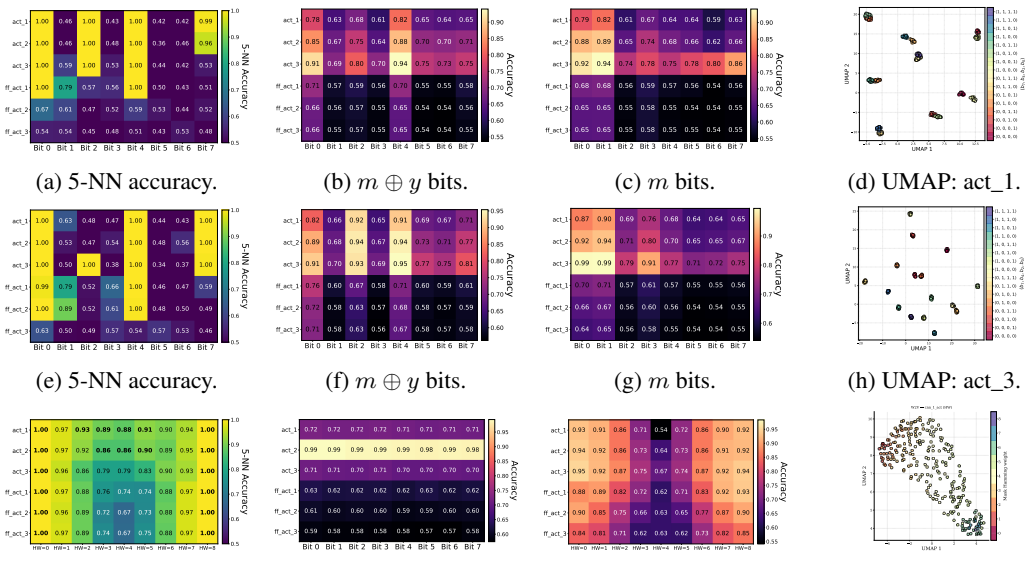
The W2V injection mode allows us not only to study the effects of injection but also to examine how the model dynamically relates mask values during trace processing. Under Eq. (1), relationships between masks can be reflected by assigning similar representations in $K^{c,p}$ to the corresponding mask values. Masks that encode similar information for the model will therefore have nearby vector representations. We visualize this structure using a two-dimensional nonlinear UMAP projection at each layer and observe clusters associated with individual bits of m_k.

To quantify the high-dimensional structure of the learned W2V representations, we applied a 5-nearest-neighbor (5-NN) classifier to the set of mask-specific key tensors $\{K^{c,p,m_k}\}_{m_k=0}^{255}$ at every injection point. An initial exploratory analysis using DBSCAN suggested that neighborhoods in this space were organized according to individual bits of the associated mask values. We therefore evaluated, for each bit b_i, how accurately its value could be predicted from the local neighborhoods of the learned key tensors. For ESHARD, we additionally evaluated whether these neighborhoods were organized according to the Hamming weight of the mask.

We compared this embedding-level analysis with linear probes trained on the corresponding network activations. At each injected layer, we trained binary logistic-regression probes to predict each bit of the mask m and of the masked intermediate value $m \oplus y$ from the trace representation. Thus, the 5-NN analysis measures how the injection mechanism organizes the 256 mask embeddings, whereas the probes measure how much information about the two shares is linearly decodable from the model representation at the same depth. We performed both analyses for all five model–dataset settings considered in this work: EffCNN and EstraNet on ASCADv1f and ASCADv1r, and EffCNN on non-shuffled ESHARD. The results are summarized in Figures 2 and 3.

When we cluster the W2V vectors, the bitwise 5-NN accuracies align much more closely with the probe accuracies for $m \oplus y$ than with those for m across both ASCAD settings in Figures 2 and 3. In particular, 5-NN accuracy shows almost no improvement for b_1, even though this bit is among the two most accurately predicted bits of m. For EffCNN, the 5-NN accuracy for b_1 increases in later layers, whereas this pattern does not appear for EstraNet. The UMAP projections likewise show clusters corresponding to combinations of individual bits.

For ESHARD, the individual-bit accuracies of both the probes and 5-NN show almost no variation across bits. This occurs because HW leakage does not identify the individual bits of a value, although it can improve over random guessing for particularly high or low Hamming weights. In contrast, the ESHARD probe and 5-NN results align much more closely with the HW of $m \oplus y$. The UMAP projection in Figure 2 similarly shows that the vectors are organized by increasing HW rather than by individual bits. Because the physical leakage of both the mask and the masked intermediate value closely follows the HW leakage model, the W2V vectors do not clearly indicate which share they match more closely in this case.



(a) 5-NN accuracy. (b) $m \oplus y$ bits. (c) m bits. (d) UMAP: act_1.

(e) 5-NN accuracy. (f) $m \oplus y$ bits. (g) m bits. (h) UMAP: act_3.

(i) 5-NN HW accuracy. (j) $m \oplus y$ bit probes. (k) $m \oplus y$ HW probe. (l) UMAP: cnn_act_1.

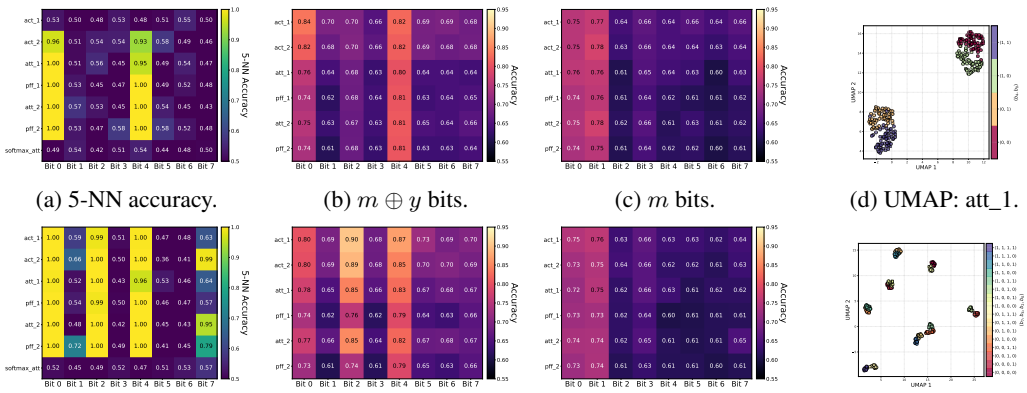
Figure 2: 5-NN, linear-probe, and W2V-embedding analyses for EffCNN. From top to bottom, the rows correspond to ASCADv1f, ASCADv1r, and non-shuffled ESHARD. The ASCAD rows report bitwise 5-NN accuracy, probes of $m \oplus y$ and m, and a representative UMAP. The ESHARD row reports bitwise probes of $m \oplus y$, Hamming-weight results for 5-NN and probing, and a W2V UMAP colored by the mask Hamming weight.

6 Discussion & Limitations.

The results in Section 5 suggest that models contain more information about the leakage of intermediate values than the original models exploit. Although it is unsurprising that providing cleaner information about a noisy mask improves attacks, the strong dependence of model performance on the form of the injection is noteworthy. In particular, the KL divergence indicates that W2V injection leads the retrained model toward substantially different solutions. Information about specific bits of the masked intermediate value that is present in the frozen early-layer representations, but not exploited by the original model, becomes useful when the model is provided with sufficiently clean mask information through the injection. Concretely, the groupings of the learned m-injection vectors visible in the 5-NN analyses and UMAP projections for ASCAD align more closely with bits that have higher probe accuracies for $m \oplus y$ than for m.

Furthermore, the performance of mask-injection mechanisms with different leakage models varies according to how well the injected values match the physical leakage. On ASCAD, LSB injection outperforms HW and ID injection, consistent with the bitwise leakage indicated by the probes and correlation analyses in Appendix F. Conversely, on ESHARD, HW injection substantially outperforms the other injection mechanisms, including W2V. This is surprising because W2V learns the HW structure, as shown in Figure 2, yet still underperforms direct HW injection. This observation motivates sweeping across leakage models to identify the closest match and better characterize the leakage.

More broadly, our overall results show that black-box evaluations may reveal only a limited subset of the leakage structure present in a model. Post hoc analysis can characterize the specific leakage function learned, but not necessarily all exploitable information available in the representations. Different objectives, regularization strategies, or masking distributions could plausibly steer the model toward exploiting different leakage structures, even when trained on the same underlying signal. This is also supported by recent work on adversarial training procedures, which shows that models can learn to use more trace points under sufficient optimization pressure (Gammell et al., 2026). This raises a broader question about the objective of DLSCA: should we focus on understanding the leakage that models naturally learn and correcting the corresponding flaws, as in Karayalçin et al. (2025), or should we design training procedures that encourage models to learn a



(a) 5-NN accuracy. (b) $m \oplus y$ bits. (c) m bits. (d) UMAP: att_1.

(e) 5-NN accuracy. (f) $m \oplus y$ bits. (g) m bits. (h) UMAP: att_2.

Figure 3: 5-NN, linear-probe, and W2V-embedding analyses for EstraNet. From top to bottom, the rows correspond to ASCADv1f and ASCADv1r. Each row reports bitwise 5-NN accuracy, probes of $m \oplus y$ and m, and a representative UMAP.

more exhaustive profile of the leakage, even if doing so reduces attack performance, as in Gammell et al. (2026)?

Limitations. Like many studies on interpreting DLSCA, our method assumes access to the internal masking randomness. Such access is not always possible in practice, even in evaluation settings (Masure et al., 2023). In addition, our method requires a model that can already recover the key; otherwise, there is no successful attack mechanism to analyze. Although unsuccessful models may be useful for identifying failure modes Perin et al. (2026), they do not provide information about successfully learned leakage because they have failed to exploit it.

Additionally, a limitation of this intervention-based analysis is that the noise introduced by the randomly initialized injection parameters can initially outweigh the benefits of the injected mask information, as observed in the early layers of EstraNet on ASCADv1f. Consequently, a weak injected attack may reflect optimization difficulties rather than the absence of useful mask features. The method therefore cannot guarantee that it reveals every mask characteristic exploited by the model, leaving room for complementary interpretability approaches.

7 Conclusions & Future Work.

In this work, we examined the interpretability challenge in DLSCA by investigating how a convolutional neural network (EffCNN) and a transformer (EstraNet) defeat Boolean masking on ASCADv1f and ASCADv1r, together with EffCNN on non-shuffled ESHARD.

By introducing the mask-injection mechanism, we established a methodology for tracing the semantic processing of mask information across network layers. We injected ground-truth mask features into specific activation maps using the ID, HW, LSB, and W2V modes and analyzed their effect on attack performance, thereby identifying where additional mask information can be exploited by the downstream network.

We apply our method to two models across three datasets, demonstrating its applicability to several challenging settings with larger models and different targets. Overall, we find that the method can reveal leakage present in model activations but not originally exploited by the model. Furthermore, our results support a layer-wise organization of mask-related processing in DLSCA models. For EffCNN, the strongest injection effects occur in the convolutional and intermediate representations, whereas EstraNet exhibits a less localized pattern extending through its attention and point-wise feed-forward blocks.

In future work, the methodology could be applied to additional checkpoints to understand how features evolve throughout training.

Reproducibility Statement

This work relies on three publicly available side-channel datasets: ASCADv1f, ASCADv1r, and the non-shuffled ESHARD dataset. We evaluate two publicly available model architectures, EffCNN and EstraNet, whose architectural configurations are described in the paper and appendix. The proposed injection mechanism is defined mathematically, including the concrete dimensions of the injected representations, and we report the hyperparameters used to train both the reference and injected models. Together, these public resources and methodological details provide the information required to fully reproduce the experiments presented in this work.

AI Use Statement

OpenAI's ChatGPT was used as a writing assistant to improve the clarity, grammar, and presentation of the manuscript. OpenAI Codex and Anthropic's Claude Code were used as programming assistants, primarily to support the development and refinement of plotting code and the visual presentation of the figures. All AI-assisted text, code, and visualizations were reviewed, edited, and validated by the authors, who take full responsibility for the contents of this work.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HJ4-rAVtl.

Andy Arditi, Oscar Balcells Obeso, Aaquib Syed, Daniel Paleka, Nina Rimsky, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=pH3XAQME6c.

Sana Boussam, Mathieu Carbone, Benoît Gérard, Guénaël Renault, and Gabriel Zaid. Optimal dimensionality reduction using conditional variational autoencoder. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(3):164–211, Jun. 2025. doi: 10.46586/tches.v2025.i3.164-211. URL https://tches.iacr.org/index.php/TCHES/article/view/12214.

Nick Cammarata, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah. Curve detectors. *Distill*, 2020. doi: 10.23915/distill.00024.003. https://distill.pub/2020/circuits/curve-detectors.

Pei Cao, Chi Zhang, Xiangjun Lu, Dawu Gu, and Sen Xu. Improving deep learning based second-order side-channel analysis with bilinear CNN. *IEEE Trans. Inf. Forensics Secur.*, 17:3863–3876, 2022. doi: 10.1109/TIFS.2022.3216959. URL https://doi.org/10.1109/TIFS.2022.3216959.

Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski, çetin K. Koç, and Christof Paar (eds.), *Cryptographic Hardware and Embedded Systems - CHES 2002*, pp. 13–28, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-36400-9.

Federal Office for Information Security (BSI). Guidelines for Evaluating Machine-Learning based Side-Channel Attack Resistance. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_46_AI_guide.pdf?__blob=publicationFile&v=6, 02 2024. Technical Report AIS 46.

Jimmy Gammell, Anand Raghunathan, Abolfazl Hashemi, and Kaushik Roy. Learning to localize leakage of cryptographic sensitive variables. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=9qxCSU8nDO.

Si Gao and Elisabeth Oswald. A novel framework for explainable leakage assessment. In Marc Joye and Gregor Leander (eds.), *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual*

International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part III, Lecture Notes in Computer Science, pp. 221–250. Springer, 2024. doi: 10.1007/978-3-031-58734-4_8. URL https://doi.org/10.1007/978-3-031-58734-4_8.

Suvadeep Hajra, Siddhartha Chowdhury, and Debdeep Mukhopadhyay. Estranet: An efficient shift-invariant transformer network for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(1):336–374, Dec. 2023. doi: 10.46586/tches.v2024.i1.336-374. URL https://tches.iacr.org/index.php/TCHES/article/view/11255.

Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. Deep neural network attribution methods for leakage analysis and symmetric key recovery. In Kenneth G. Paterson and Douglas Stebila (eds.), *Selected Areas in Cryptography – SAC 2019*, pp. 645–666, Cham, 2020. Springer International Publishing. ISBN 978-3-030-38471-5.

ISO/IEC 15408-1:2022. Information security, cybersecurity and privacy protection — Evaluation criteria for IT security. Standard, International Organization for Standardization, Geneva, CH, August 2022.

Sengim Karayalçin, Marina Krček, and Stjepan Picek. Interpreting emergent features in deep learning-based side-channel analysis. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=GjkuXI7hX7.

Sengim Karayalcin, Marina Krcek, and Stjepan Picek. A practical tutorial on deep learning-based side-channel analysis. Cryptology ePrint Archive, Paper 2025/471, 2025. URL https://eprint.iacr.org/2025/471.

Sengim Karayalcin, Marina Krček, and Stjepan Picek. SoK: Deep learning-based physical side-channel analysis. In *35th USENIX Security Symposium (USENIX Security 26)*, pp. 5231–5253, Baltimore, MD, August 2026. USENIX Association. ISBN 978-1-939133-58-8. URL https://www.usenix.org/conference/usenixsecurity26/presentation/karayalcin.

Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz (ed.), *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pp. 104–113. Springer, 1996. doi: 10.1007/3-540-68697-5_9. URL https://doi.org/10.1007/3-540-68697-5_9.

Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener (ed.), *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, Lecture Notes in Computer Science, pp. 388–397. Springer, 1999. doi: 10.1007/3-540-48405-1_25. URL https://doi.org/10.1007/3-540-48405-1_25.

Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat (eds.), *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*, pp. 3–26. Springer, 2016. doi: 10.1007/978-3-319-49445-6_1. URL https://doi.org/10.1007/978-3-319-49445-6__1.

Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Gradient visualization for general characterization in profiling attacks. In *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, pp. 145–167, 2019. doi: 10.1007/978-3-030-16350-1_9. URL https://doi.org/10.1007/978-3-030-16350-1_9.

Loïc Masure, Valence Cristiani, Maxime Lecomte, and François-Xavier Standaert. Don't learn what you already know scheme-aware modeling for profiling side-channel analysis against masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):32–59, 2023. doi: 10.46586/TCHES.V2023.I1.32-59. URL https://doi.org/10.46586/tches.v2023.i1.32-59.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=UGpGkLzwpP.

Guilherme Perin, Lichao Wu, and Stjepan Picek. Exploring feature selection scenarios for deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):828–861, Aug. 2022. doi: 10.46586/tches.v2022.i4.828-861. URL https://tches.iacr.org/index.php/TCHES/article/view/9842.

Guilherme Perin, Sengim Karayalçin, Lichao Wu, and Stjepan Picek. I know what your layers did: Layer-wise explainability of deep learning side-channel analysis. *IACR Communications in Cryptology*, 2(4), 2026. ISSN 3006-5496. doi: 10.62056/a0wahee-3.

Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55(11), February 2023. ISSN 0360-0300. doi: 10.1145/3569577. URL https://doi.org/10.1145/3569577.

Thomas Roche, Victor Lomné, Camille Mutschler, and Laurent Imbert. A side journey to titan. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 231–248. USENIX Association, August 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/roche.

Ludwig Schubert, Chelsea Voss, Nick Cammarata, Gabriel Goh, and Chris Olah. High-low frequency detectors. *Distill*, 2021. doi: 10.23915/distill.00024.005. https://distill.pub/2020/circuits/frequency-edges.

Vivswan Shah and Nathan Youngblood. Leveraging continuously differentiable activation for learning in analog and quantized noisy environments. *IEEE Journal of Selected Topics in Quantum Electronics*, 31(3: AI/ML Integrated Opto-electronics):1–9, 2025. doi: 10.1109/JSTQE.2025.3534636.

François-Xavier Standaert, Tal G. Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux (ed.), *Advances in Cryptology - EUROCRYPT 2009*, pp. 443–461, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-01001-9.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Aurélien Vasselle, Hugues Thiebeauld, and Philippe Maurine. Spatial dependency analysis to extract information from side-channel mixtures: extended version. *Journal of Cryptographic Engineering*, 13(4):409–425, January 2023. ISSN 2190-8516. doi: 10.1007/s13389-022-00307-9. URL http://dx.doi.org/10.1007/s13389-022-00307-9.

Lichao Wu, Amir Alipour, Azade Rezaeezade, Guilherme Perin, and Stjepan Picek. Breaking free: Leakage model-free deep learning-based side-channel analysis. *IACR Cryptol. ePrint Arch.*, pp. 1110, 2023. URL https://eprint.iacr.org/2023/1110.

Trevor Yap, Adrien Benamira, Shivam Bhasin, and Thomas Peyrin. Peek into the black-box: Interpretable neural network using SAT equations in side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):24–53, 2023. doi: 10.46586/TCHES.V2023.I2.24-53. URL https://doi.org/10.46586/tches.v2023.i2.24-53.

Trevor Yap, Stjepan Picek, and Shivam Bhasin. Occpois: Points of interest based on neural network's key recovery in side-channel analysis through occlusion. In Sourav Mukhopadhyay and Pantelimon Stanica (eds.), *Progress in Cryptology - INDOCRYPT 2024 - 25th International Conference on Cryptology in India, Chennai, India, December 18-21, 2024, Proceedings, Part II*, volume 15496 of *Lecture Notes in Computer Science*, pp. 3–28. Springer, 2024. doi: 10.1007/978-3-031-80311-6_1. URL https://doi.org/10.1007/978-3-031-80311-6_1.

Kota Yoshida, Sengim Karayalcin, and Stjepan Picek. Can kans do it? toward interpretable deep learning-based side-channel analysis. In Mark Manulis (ed.), *Applied Cryptography and Network Security Workshops - ACNS 2025 Satellite Workshops: AIHWS, AIoTS, QSHC, SCI, PrivCrypt, SPIQE, SiMLA, and CIMSS 2025, Munich, Germany, June 23-26, 2025, Revised Selected Papers, Part I*, volume 15653 of *Lecture Notes in Computer Science*, pp. 40–57. Springer, 2025. doi: 10.1007/978-3-032-01799-4_3. URL https://doi.org/10.1007/978-3-032-01799-4_3.

Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient CNN architectures in profiling attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):1–36, 2020. doi: 10.13154/TCHES.V2020.I1.1-36. URL https://doi.org/10.13154/tches.v2020.i1.1-36.

Gabriel Zaid, Lilian Bossuet, Mathieu Carbone, Amaury Habrard, and Alexandre Venelli. Conditional variational autoencoder based on stochastic attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):310–357, 2023. doi: 10.46586/TCHES.V2023.I2.310-357. URL https://doi.org/10.46586/tches.v2023.i2.310-357.

Shijun Zhang, Jianfeng Lu, and Hongkai Zhao. Deep network approximation: Beyond relu to diverse activation functions. *J. Mach. Learn. Res.*, 25:35:1–35:39, 2024. URL https://jmlr.org/papers/v25/23-0912.html.

A Description of the Datasets

ASCADv1. ASCAD datasets represent the most common target for evaluating the performance of DLSCA. The first edition of the dataset (ASCADv1) contains two versions: one with a fixed key for the profiling set and the other with random keys for the profiling set. We denote the former as ASCADv1f and the latter as ASCADv1r. The ASCADv1f dataset[2] contains measurements from an 8-bit AVR microcontroller. The AES-128 implementation is protected with the first-order Boolean masking, except for the first two bytes, which are left unprotected for testing purposes. The acquisition window covers only the first round of AES, and each trace consists of $100\,000$ sample points. The dataset contains $60\,000$ traces overall, which are split into $50\,000$ for the profiling phase and $10\,000$ for the attack. All encryption operations in both the profiling and attack groups are performed using the same fixed key. Following the recommendations from the datasets' authors, we consider the preselected interval of 700 features for our experiments. Note that this is the OPOI cases as discussed in Perin et al. (2022).

The ASCADv1r dataset[3] has a similar setup to ASCADv1f concerning the measurements source. The main difference is that the traces in the ASCADv1r profiling group use random keys for encryption rather than a fixed key. The traces are longer as they consist of $250\,000$ samples. In total, there are $200\,000$ traces for the profiling phase and $100\,000$ for the attack phase. The encryption for the attack group is performed using a fixed key. We again work with a preselected window of features, which for this dataset contains $1\,400$ features. For both ASCAD datasets, we consider the Identity leakage model, and we attack the third key byte, as this is the first masked key byte.

ESHARD. The ESHARD dataset[4] contains EM emanations of a first-order Boolean masked AES-128 implementation implemented on an STM32F4 microcontroller (Cortex-M4) and was introduced by Vasselle et al. Vasselle et al. (2023). The dataset includes $100\,000$ traces with a fixed key, where we take $90\,000$ traces as the profiling set and $10\,000$ traces as the attack set. Each trace contains a window of $1\,400$ samples per trace, consisting of the loading of the mask and the 16 masked S-box operations. Note that we consider the non-shuffled version of the traces. For attacks on ESHARD, we use the Hamming weight of the first byte of the intermediate value as the leakage model, following Wu et al. (2023).

B EffCNN's Architecture

The architecture studied for EffCNN Zaid et al. (2020) is shown in Table 1. We replaced the last average pooling layer of the convolutional block with an adaptive average pooling layer. This always gives an output with 128 characteristics and 7 samples. This allows us to use it not only with ASCADv1f traces, as it was originally tested, but also with ASCADv1r traces, which contain 1400 samples. Table 1 summarizes the architectural parameters, where c_{out} denotes the number of output channels, ks represents the kernel size, and p is the padding. Furthermore, out_size indicates the output dimension of the adaptive average pooling layer, while L refers to the input signal length. All convolutional layers employ the Scaled Exponential Linear Unit (SELU) as the activation function.

C EstraNet's Architecture

The architecture implemented for EstraNet Hajra et al. (2023) is presented in Table 2. This table summarizes the key architectural hyperparameters, where c_{in} and c_{out} denote the number of input and output channels, respectively, and ks represents the kernel size for the convolutional layers. For the attention and feed-forward mechanisms, d_{model} defines the embedding dimension, n_{head} indicates the number of attention heads, d_{head} represents the dimension of each head, d_{inner} specifies the inner projection dimension of the pointwise feed-forward networks, and p corresponds to the dropout probability. Furthermore, L denotes the initial input signal length, with L_1 and L_2 denoting

[2] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_fixed_key

[3] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_variable_key

[4] https://gitlab.com/eshard/nucleo_sw_aes_masked_shuffled/-/blob/main/Nucleo_AES_masked_non_shuffled.ets

Table 1: EffCNN Architecture.

Block	Layer Type	Hyperparameters	Output Shape
Convolutional	Conv1D	$c_{out} = 32, ks = 1, p = 0$	$(32, L)$
	BatchNorm1D	–	$(32, L)$
	SELU	–	$(32, L)$
	AvgPool1D	$ks = 2$	$(32, L/2)$
	Conv1D	$c_{out} = 64, ks = 25, p = 12$	$(64, L/2)$
	BatchNorm1D	–	$(64, L/2)$
	SELU	–	$(64, L/2)$
	AvgPool1D	$ks = 25$	$(64, L/50)$
	Conv1D	$c_{out} = 128, ks = 3, p = 1$	$(128, L/50)$
	BatchNorm1D	–	$(128, L/50)$
	SELU	–	$(128, L/50)$
	AdaptiveAvgPool1D	$out_size = 7$	$(128, 7)$
Fully Connected	Flatten	–	896
	Dense	$c_{out} = 15$, SELU	15
	Dense	$c_{out} = 15$, SELU	15
	Dense	$c_{out} = 15$, SELU	15
Output	Prediction (Softmax)	$c_{out} = 256$	256

the progressively reduced sequence lengths resulting from the average pooling operations. All convolutional layers in this architecture employ the Rectified Linear Unit (ReLU) as their activation function.

Table 2: EstraNet Architecture.

Block	Layer Type	Hyperparameters	Output Shape
Convolutional	Conv1D	$c_{in} = 1, c_{out} = 8, ks = 11$	$(8, L)$
	ReLU	–	$(8, L)$
	AvgPool1D	$ks = 10$	$(8, L_1)$
	Conv1D	$c_{in} = 8, c_{out} = 128, ks = 3$	$(128, L_1)$
	ReLU	–	$(128, L_1)$
	AvgPool1D	$ks = 10$	$(128, L_2)$
	Permute	–	$(L_2, 128)$
Attention	Positional Encoding	–	$(L_2, 128)$
	LayerCentering	–	$(L_2, 128)$
	SelfAttention	$d_{model} = 128, n_{head} = 8, d_{head} = 32, p = 0.05$	$(L_2, 128)$
	LayerCentering	–	$(L_2, 128)$
	PointwiseFeedForward	$d_{inner} = 256, p = 0.05$	$(L_2, 128)$
	LayerCentering	–	$(L_2, 128)$
	SelfAttention	$d_{model} = 128, n_{head} = 8, d_{head} = 32, p = 0.05$	$(L_2, 128)$
	LayerCentering	–	$(L_2, 128)$
	PointwiseFeedForward	$d_{inner} = 256, p = 0.05$	$(L_2, 128)$
	Dropout	$p = 0.05$	$(L_2, 128)$
Output	SoftmaxAttention	$d_{model} = 128, n_{head} = 8, d_{head} = 16$	$(L_2, 128)$
	Mean (dim=1)	–	128
	Prediction (Linear)	$c_{out} = 256$	256

D Additional Analyses and Controls

This section reports supplementary experiments and control analyses. We use REF for the original reference model, JFW for retraining without informative injection, and RM for random-mask injection. ID, HW, LSB, and W2V denote identity, Hamming-weight, least-significant-bit, and trainable-embedding mask representations, respectively.

D.1 Injection-Mode Differences

Figure 4 compares the injected models against REF and RM across the five available model–dataset settings. For ASCAD, the plots report differences in $N_{GE=0}$; for ESHARD, they report differences in AEP.

Figure 5 presents the corresponding KL divergences using the same row and column organization.

Comparisons against REF exhibit the same qualitative layer-wise trends as those obtained against JFW, indicating that our conclusions are not an artifact of selecting JFW as the comparator. Moreover, RM behaves similarly to JFW despite using the same injection architecture and additional trainable parameters as W2V. Because the masks supplied to RM are independent of the true sample masks, this result indicates that the added capacity alone does not explain the improvements obtained with informative injection.

D.2 Direct Intervention on the Original Model

We directly intervened on the trained reference models by removing activation-space directions associated with individual mask bits. As a control, we removed random directions with the same dimensionality. Tables 3 and 4 report the resulting change in attack performance. Most bit-direction interventions have little or no effect and are generally comparable to the random-direction control, although isolated larger changes occur in later EffCNN layers and in the EstraNet attention blocks.

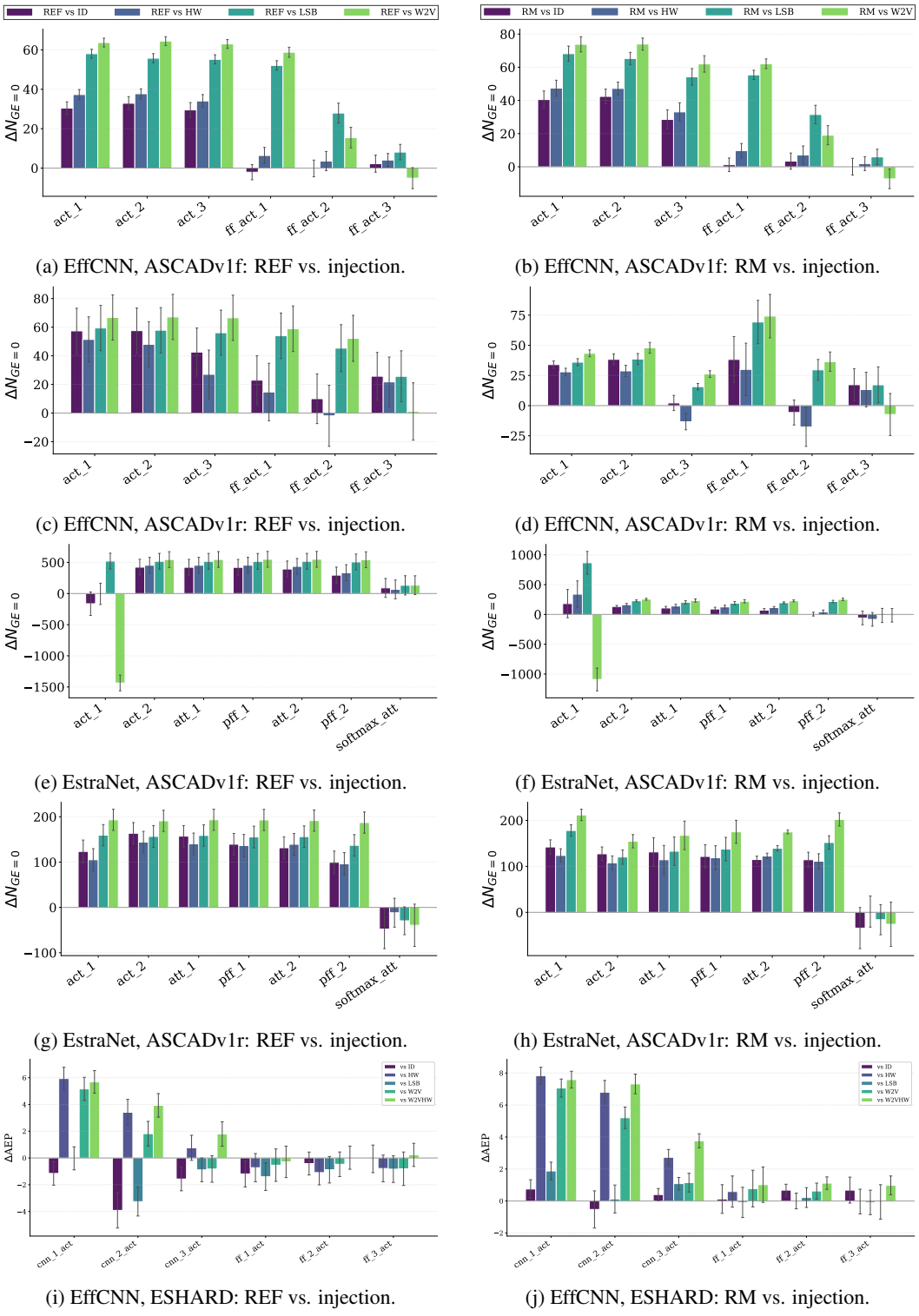
The absence of a systematic effect in this direct ablation contrasts with the clear layer- and encoding-dependent patterns revealed by mask injection. This suggests that removing a single probe-derived linear direction is insufficient to characterize distributed or nonlinear mask representations, whereas mask injection tests whether explicit mask semantics can be functionally exploited under downstream adaptation.

D.3 ESHARD with EstraNet

We report the EstraNet experiments on ESHARD for completeness. Applying our injection methodology requires a reference model that can first perform a successful attack. However, EstraNet did not attack ESHARD reliably: among 36 tested training configurations, only one produced a successful reference model, requiring 4486 traces and therefore substantially more attack traces than the other settings considered in this work. Obtaining this model also required extending evaluation during training to a much larger trace budget, which substantially increased the computational cost of model selection. We nevertheless applied the injection methodology to this single successful instance and report the resulting exploratory analysis in Table 5. None of the retrained or injected variants achieved a successful attack. Accordingly, REF is reported using $N_{GE=0}$, whereas the remaining models are reported using their residual guessing entropy. These quantities are not directly comparable; lower residual GE indicates better performance, but all values remain above one. Because these results are derived from a single successful reference model, they should be treated as illustrative rather than as conclusive evidence about EstraNet on ESHARD.

D.4 Number of Retained Runs

Table 6 reports the number of independent runs retained for computing the means and standard deviations presented above. The maximum number of runs differs across settings primarily because of their different computational costs. In addition, DLSCA training exhibits substantial variability across random initializations, and some runs do not reach the predefined attack-success criterion. We classify a run as successful when, within the 2,000 evaluated attack traces, the correct key reaches rank 1 before the complete trace budget is exhausted; under the zero-based rank convention used throughout this paper, this corresponds to reaching $GE = 0$ for some $N_{GE=0} < 2{,}000$. Because our methodology is intended to characterize how mask information can be exploited by a functional attack, the primary analysis is conditioned on runs satisfying this criterion; all other runs are excluded independently of their relative performance. This conditioning may introduce selection bias, so we report the retained sample size for every setting and explicitly identify all configurations with fewer runs than the stated maximum. Whenever computationally feasible, we use multiple independent runs to reduce dependence on any single initialization.



(a) EffCNN, ASCADv1f: REF vs. injection.

(b) EffCNN, ASCADv1f: RM vs. injection.

(c) EffCNN, ASCADv1r: REF vs. injection.

(d) EffCNN, ASCADv1r: RM vs. injection.

(e) EstraNet, ASCADv1f: REF vs. injection.

(f) EstraNet, ASCADv1f: RM vs. injection.

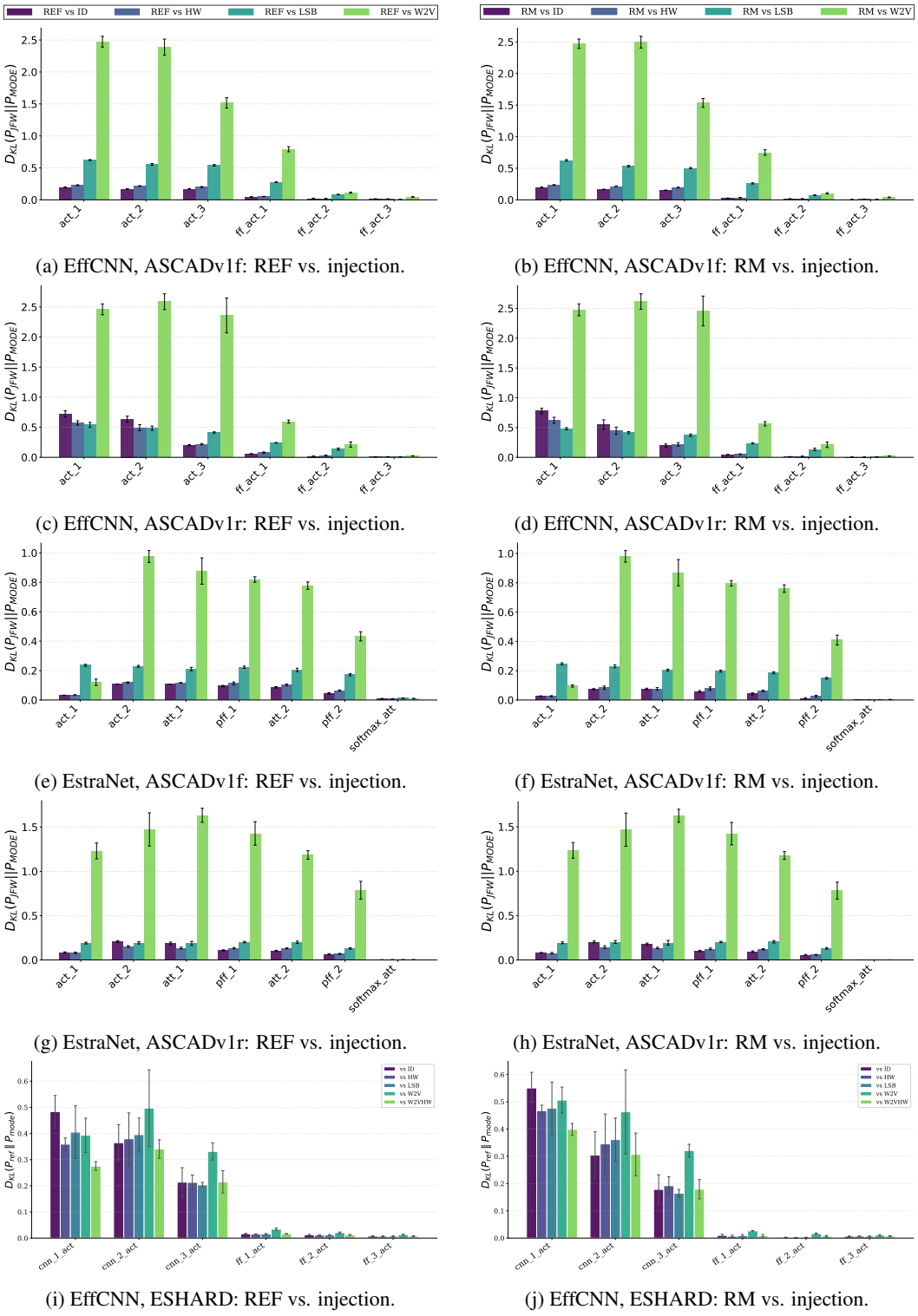
(g) EstraNet, ASCADv1r: REF vs. injection.

(h) EstraNet, ASCADv1r: RM vs. injection.

(i) EffCNN, ESHARD: REF vs. injection.

(j) EffCNN, ESHARD: RM vs. injection.

Figure 4: Differences between the injected models and the REF (left column) or RM (right column) comparators. From top to bottom, the rows correspond to EffCNN on ASCADv1f, EffCNN on ASCADv1r, EstraNet on ASCADv1f, EstraNet on ASCADv1r, and EffCNN on non-shuffled ESHARD. ASCAD results use $\Delta N_{GE=0}$, whereas ESHARD uses ΔAEP.



(a) EffCNN, ASCADv1f: REF vs. injection.

(b) EffCNN, ASCADv1f: RM vs. injection.

(c) EffCNN, ASCADv1r: REF vs. injection.

(d) EffCNN, ASCADv1r: RM vs. injection.

(e) EstraNet, ASCADv1f: REF vs. injection.

(f) EstraNet, ASCADv1f: RM vs. injection.

(g) EstraNet, ASCADv1r: REF vs. injection.

(h) EstraNet, ASCADv1r: RM vs. injection.

(i) EffCNN, ESHARD: REF vs. injection.

(j) EffCNN, ESHARD: RM vs. injection.

Figure 5: KL divergence between the injected models and the REF (left column) or RM (right column) comparators. From top to bottom, the rows correspond to EffCNN on ASCADv1f, EffCNN on ASCADv1r, EstraNet on ASCADv1f, EstraNet on ASCADv1r, and EffCNN on non-shuffled ESHARD.

Table 3: Direct intervention results for EffCNN. "Random" reports mean ± standard deviation across random directions.

Dataset	Layer	Dim.	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	Random
ASCADv1f	act_1	22400	0	0	0	0	0	0	0	0	-0.4 ± 0.8
ASCADv1f	act_2	22400	0	0	0	0	0	0	0	0	-0.4 ± 0.8
ASCADv1f	act_3	1792	0	0	0	0	0	0	0	0	-0.4 ± 0.8
ASCADv1f	ff_act_1	15	1	0	0	0	0	4	3	0	14.6 ± 35.8
ASCADv1f	ff_act_2	15	3	2	3	1	1	1	1	1	6.0 ± 17.8
ASCADv1f	ff_act_3	15	0	0	0	0	0	0	0	0	2.2 ± 14.2
ASCADv1r	act_1	44800	2	1	1	0	1	0	0	1	-2.5 ± 4.3
ASCADv1r	act_2	44800	0	0	0	0	0	0	1	0	-1.6 ± 3.6
ASCADv1r	act_3	3584	1	1	1	1	0	0	0	0	-1.2 ± 4.0
ASCADv1r	ff_act_1	15	83	11	-8	18	-8	-2	0	-2	141.5 ± 221.2
ASCADv1r	ff_act_2	15	24	117	26	37	26	5	26	-9	373.0 ± 472.2
ASCADv1r	ff_act_3	15	1	1	-2	-2	-2	0	0	1	8.1 ± 9.2

Table 4: Direct intervention results for EstraNet. "Random" reports mean ± standard deviation across random directions.

Dataset	Layer	Dim.	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	Random
ASCADv1f	act_1	5520	0	0	0	0	0	0	0	0	-0.1 ± 1.1
ASCADv1f	act_2	8576	0	0	0	0	0	0	0	0	-0.2 ± 1.2
ASCADv1f	att_1	768	0	1	0	0	0	0	0	0	1.4 ± 8.8
ASCADv1f	pff_1	768	0	1	0	0	0	0	0	0	0.0 ± 0.6
ASCADv1f	att_2	768	1	54	0	0	0	0	0	0	0.0 ± 0.8
ASCADv1f	pff_2	768	0	1	0	0	0	0	0	0	-0.2 ± 1.0
ASCADv1f	softmax_att	768	0	0	0	0	0	0	0	0	-0.4 ± 1.5
ASCADv1r	act_1	11120	0	0	0	0	0	0	0	0	0.7 ± 6.6
ASCADv1r	act_2	17536	0	0	0	0	0	0	0	0	-0.7 ± 3.4
ASCADv1r	att_1	1664	38	19	0	14	0	-1	-1	-1	0.2 ± 4.8
ASCADv1r	pff_1	1664	0	0	-1	0	0	0	-1	-1	-0.3 ± 2.1
ASCADv1r	att_2	1664	0	0	-1	0	-1	0	0	0	-0.5 ± 2.1
ASCADv1r	pff_2	1664	-1	-1	0	-2	0	-1	0	0	2.3 ± 6.7
ASCADv1r	softmax_att	1664	0	0	0	0	0	0	0	0	-0.8 ± 0.8

Table 5: EstraNet on ESHARD. REF reports $N_{GE=0}$, whereas all other columns report residual guessing entropy.

Injection point	REF: $N_{GE=0}$	JFW: GE	RM: GE	W2V: GE	ID: GE	HW: GE	LSB: GE
act_1	4486.00 ± 0.00	87.23 ± 0.00	30.07 ± 0.00	74.90 ± 0.00	72.70 ± 0.00	61.83 ± 0.00	31.23 ± 0.00
act_2	4486.00 ± 0.00	83.83 ± 0.00	14.30 ± 0.00	73.80 ± 0.00	57.80 ± 0.00	27.97 ± 0.00	19.03 ± 0.00
att_1	4486.00 ± 0.00	72.20 ± 0.00	52.03 ± 0.00	25.67 ± 0.00	43.43 ± 0.00	19.93 ± 0.00	10.73 ± 0.00
pff_1	4486.00 ± 0.00	101.60 ± 0.00	96.27 ± 0.00	42.40 ± 0.00	44.80 ± 0.00	44.20 ± 0.00	62.67 ± 0.00
att_2	4486.00 ± 0.00	80.47 ± 0.00	47.70 ± 0.00	41.17 ± 0.00	23.23 ± 0.00	4.97 ± 0.00	36.13 ± 0.00
pff_2	4486.00 ± 0.00	56.33 ± 0.00	29.83 ± 0.00	15.97 ± 0.00	12.27 ± 0.00	43.33 ± 0.00	4.67 ± 0.00
softmax_att	4486.00 ± 0.00	15.03 ± 0.00	3.67 ± 0.00	8.43 ± 0.00	4.90 ± 0.00	14.10 ± 0.00	3.57 ± 0.00

E UMAP PROJECTIONS OF ACTIVATION LAYERS

This section provides the UMAP projections of the learned W2V representations at every analyzed activation layer. For EstraNet, we additionally include the final softmax-attention layer. The ASCAD projections are colored according to the mask bits identified in each layer, whereas the ESHARD projections are colored according to the Hamming weight of the mask.

F PEARSON CORRELATION ANALYSIS

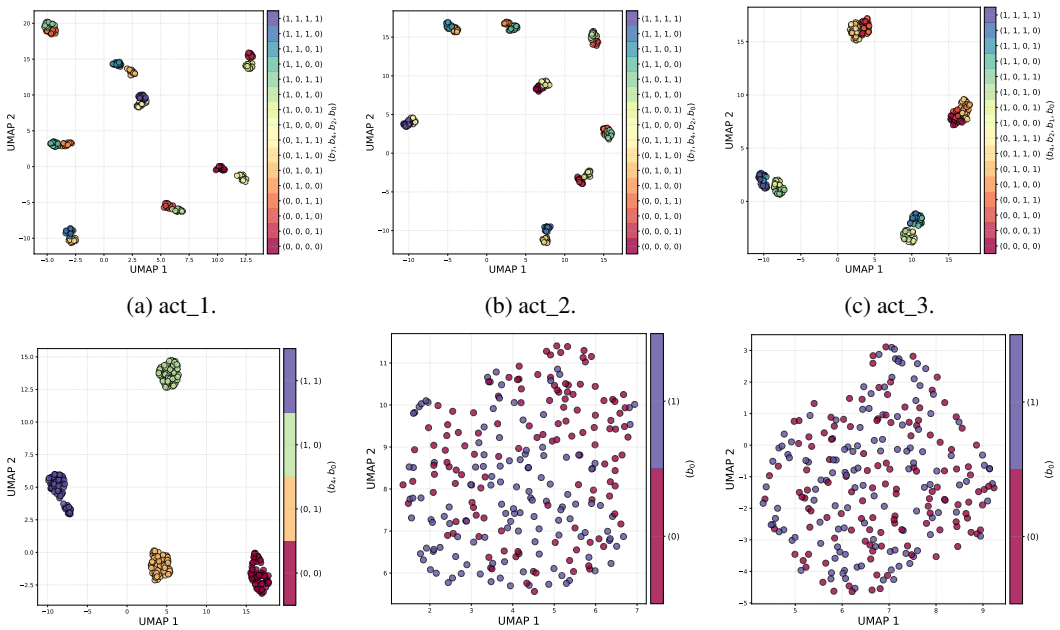
Figure 12 reports the Pearson correlation between the trace samples and the bits of the mask m or the masked intermediate value $m \oplus y$ for the ASCAD datasets. For ASCAD, the bits that leak the most information about the masked intermediate value (b_4, b_0, and b_2) are also those that best explain the clusters observed in the UMAP projections and the 5-NN accuracies.

G GUESSING ENTROPY

Here, we report the guessing-entropy results for EffCNN and EstraNet on ASCADv1f and ASCADv1r, together with the activation-layer results for EffCNN on ESHARD. Each subfigure includes W2V, ID, HW, LSB, and JFW, together with REF (the original trained model used as the reference) and RM (a W2V-like control that receives random masks and isolates the effect of the additional trainable injection parameters). The results are shown in Figures 13, 15, 14, 16, and 17.

Table 6: Maximum numbers of retained runs and configurations with fewer retained runs for each setting.

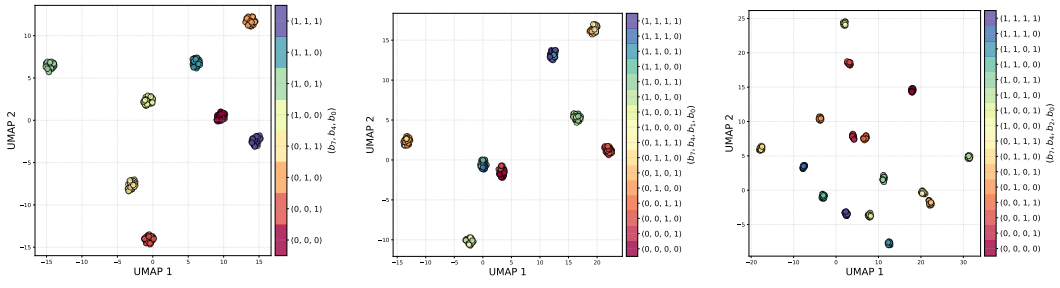
Setting	Maximum	Exceptions
ASCADv1f, EffCNN	20	ff_act_1/W2V: 19; ff_act_2/LSB: 19
ASCADv1f, EstraNet	6	cnn_1/JFW: 5; cnn_1/RM: 5; cnn_act_1/JFW: 5; cnn_act_1/ID: 5; cnn_act_1/HW: 5; cnn_poo_1/RM: 5; cnn_poo_1/ID: 5; cnn_2/W2V: 5; cnn_act_2/W2V: 5; cnn_poo_2/W2V: 5
ASCADv1r, EffCNN	10	Several exclusions from cnn_3 onward; minimum: cnn_act_3/RM: 4; cnn_poo_3/ID: 5; cnn_poo_3/JFW: 6
ASCADv1r, EstraNet	4	cnn_1/W2V: 3
ESHARD, EffCNN	5	None; all configurations retain all five seeds



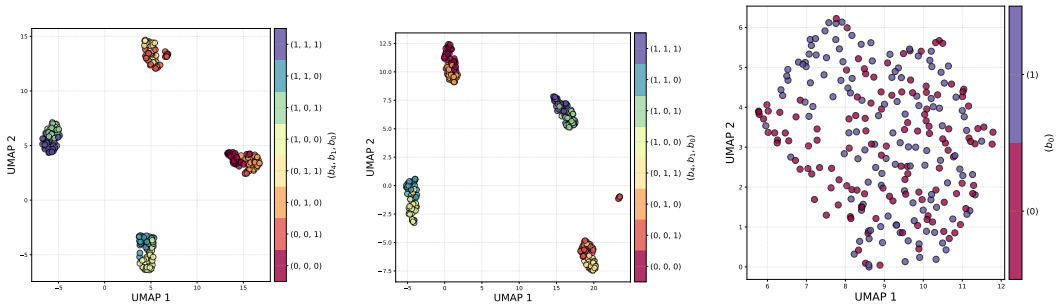
(a) act_1. (b) act_2. (c) act_3.

(d) ff_act_1. (e) ff_act_2. (f) ff_act_3.

Figure 6: UMAP projections of the W2V representations at the activation layers of EffCNN trained on ASCADv1f.

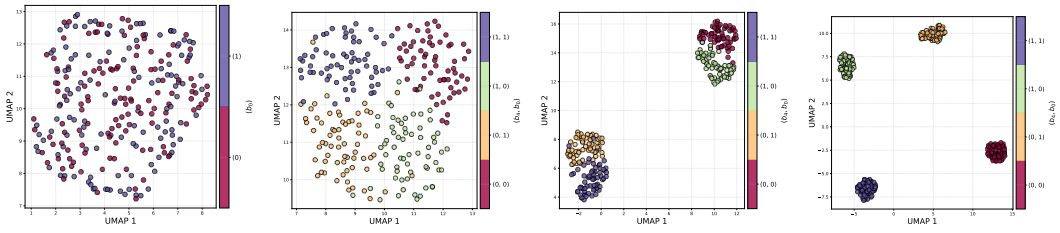


(a) act_1. (b) act_2. (c) act_3.

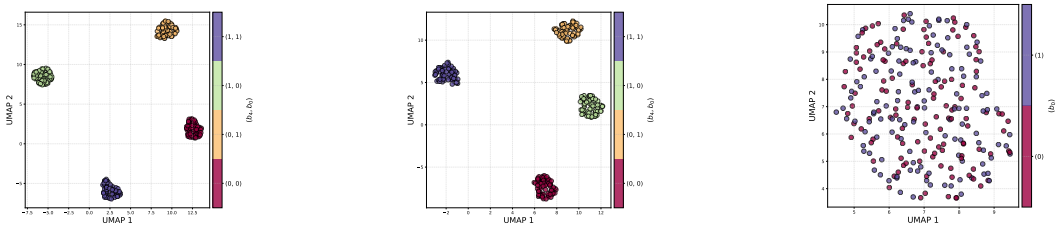


(d) ff_act_1. (e) ff_act_2. (f) ff_act_3.

Figure 7: UMAP projections of the W2V representations at the activation layers of EffCNN trained on ASCADv1r.

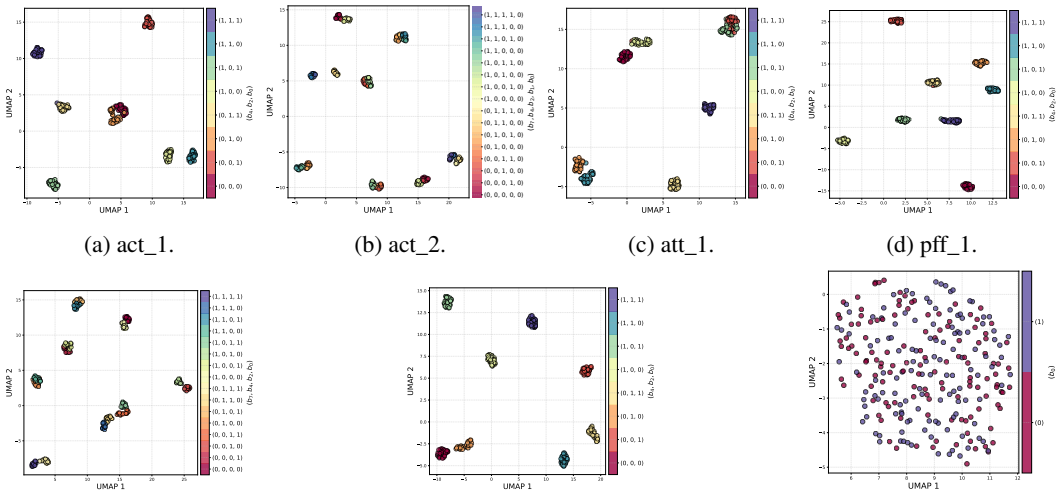


(a) act_1. (b) act_2. (c) att_1. (d) pff_1.



(e) att_2. (f) pff_2. (g) softmax_att.

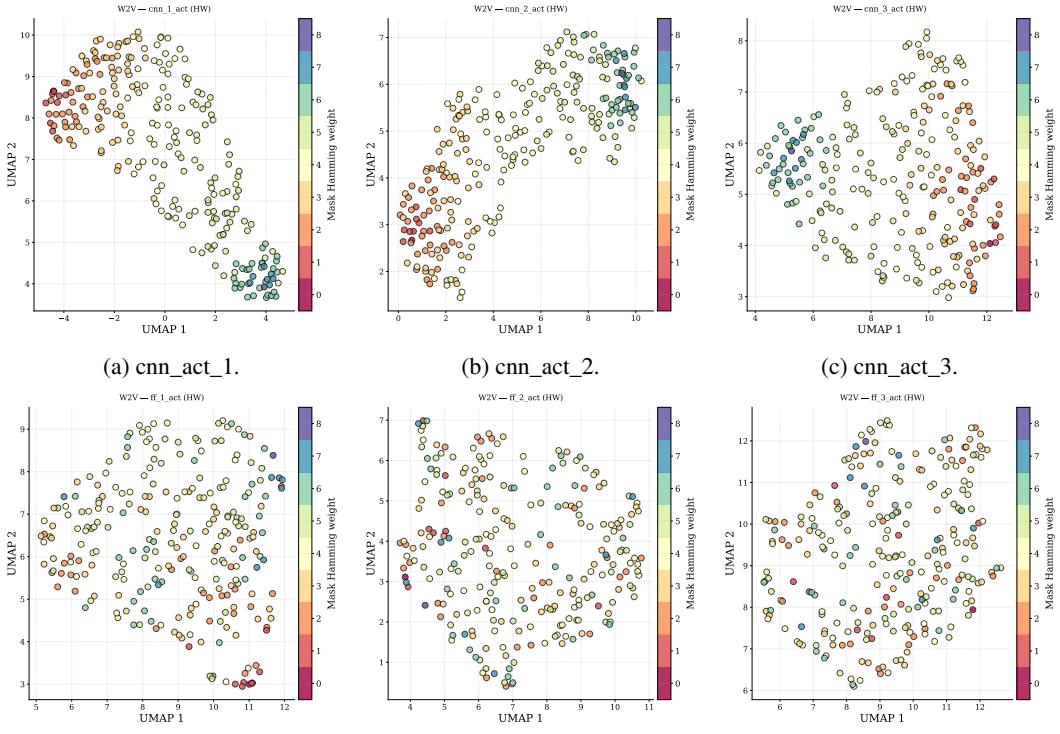
Figure 8: UMAP projections of the W2V representations at the activation and softmax-attention layers of EstraNet trained on ASCADv1f.



(a) act_1. (b) act_2. (c) att_1. (d) pff_1.

(e) att_2. (f) pff_2. (g) softmax_att.

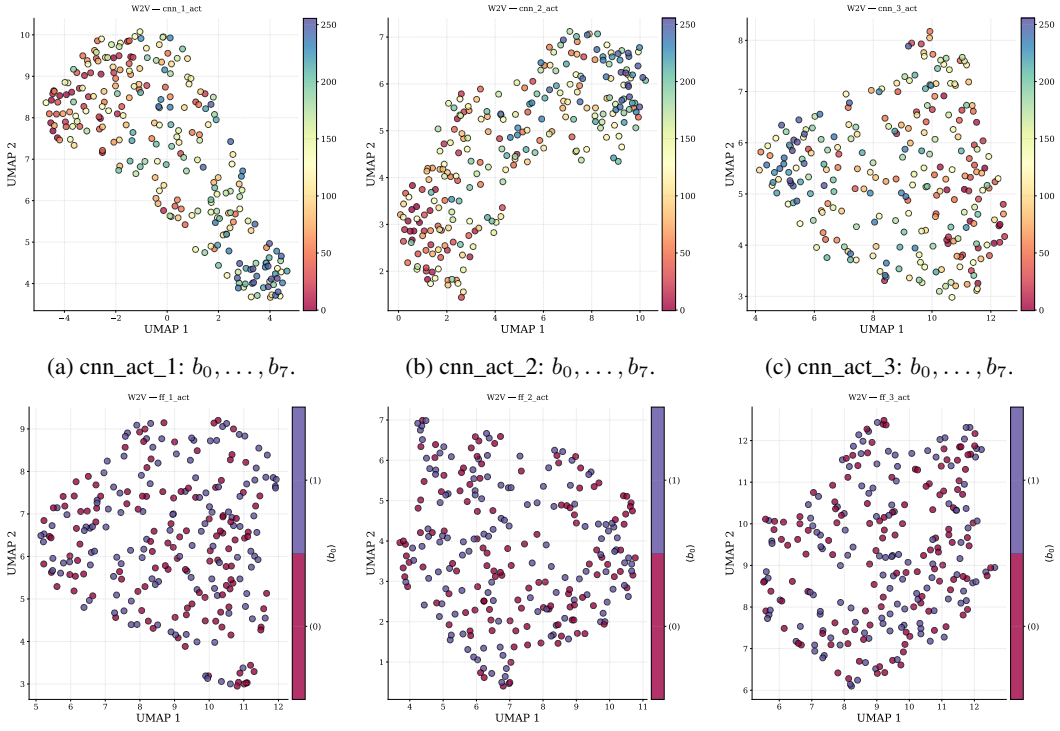
Figure 9: UMAP projections of the W2V representations at the activation and softmax-attention layers of EstraNet trained on ASCADv1r.



(a) cnn_act_1. (b) cnn_act_2. (c) cnn_act_3.

(d) ff_act_1. (e) ff_act_2. (f) ff_act_3.

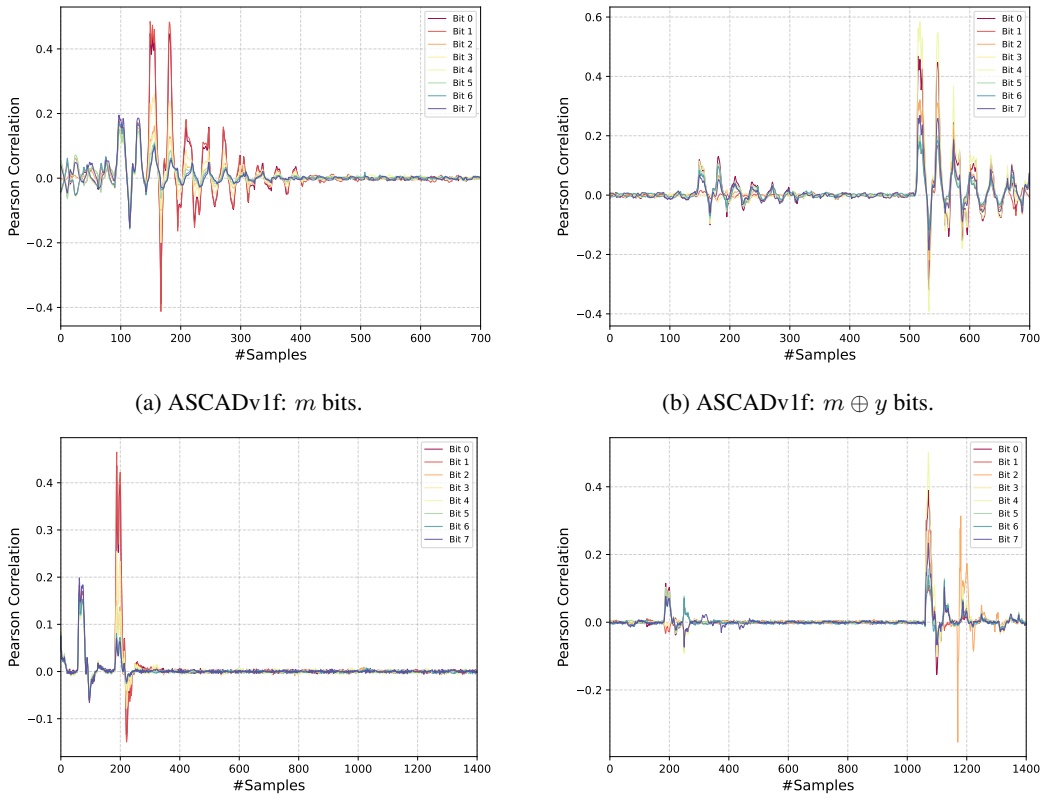
Figure 10: UMAP projections of the W2V representations at the activation layers of EffCNN trained on non-shuffled ESHARD, colored according to the Hamming weight of the mask.



(a) cnn_act_1: $b_0, \ldots, b_7$. (b) cnn_act_2: $b_0, \ldots, b_7$. (c) cnn_act_3: $b_0, \ldots, b_7$.

(d) ff_act_1: b_0. (e) ff_act_2: b_0. (f) ff_act_3: b_0.

Figure 11: UMAP projections of the W2V representations at the activation layers of EffCNN trained on non-shuffled ESHARD, colored according to the available mask-bit configurations.



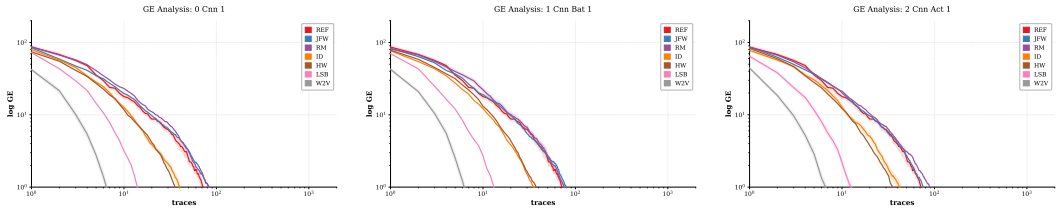
(a) ASCADv1f: m bits.

(b) ASCADv1f: $m \oplus y$ bits.

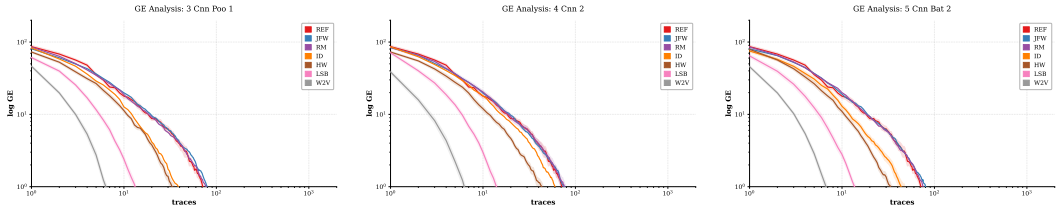
(c) ASCADv1r: m bits.

(d) ASCADv1r: $m \oplus y$ bits.

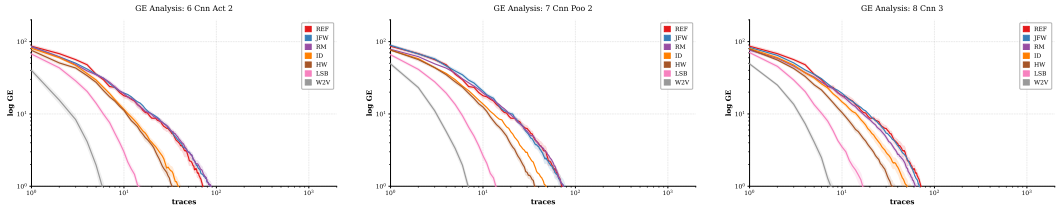
Figure 12: Pearson correlation of the mask and masked intermediate value bits with the traces of ASCADv1f and ASCADv1r.



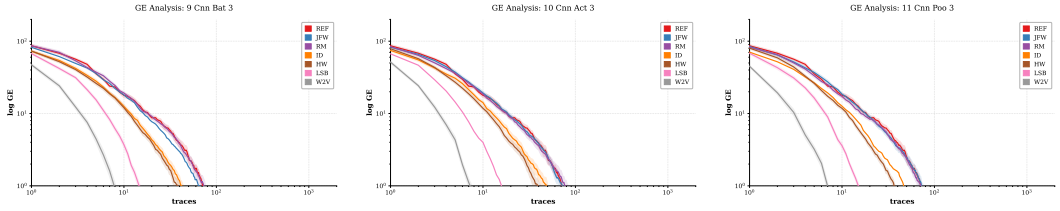
(a) L0: cnn_1 (b) L1: cnn_bat_1 (c) L2: cnn_act_1



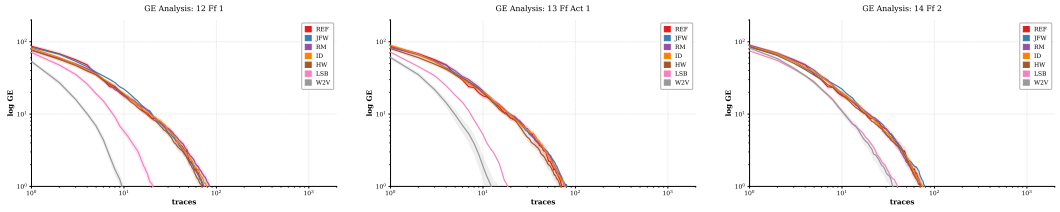
(d) L3: cnn_poo_1 (e) L4: cnn_2 (f) L5: cnn_bat_2



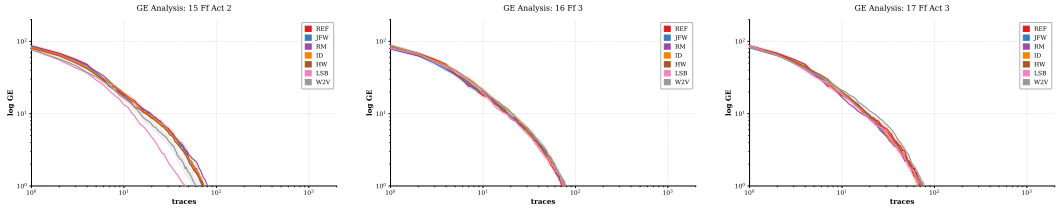
(g) L6: cnn_act_2 (h) L7: cnn_poo_2 (i) L8: cnn_3



(j) L9: cnn_bat_3 (k) L10: cnn_act_3 (l) L11: cnn_poo_3

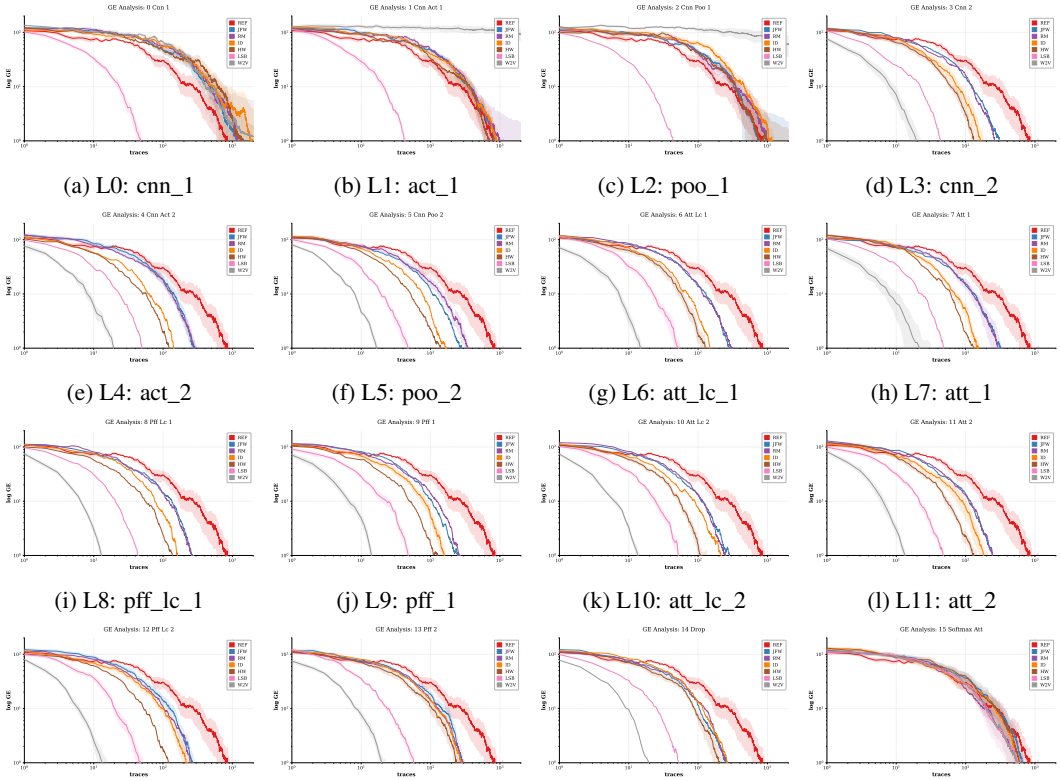


(m) L12: ff_1 (n) L13: ff_act_1 (o) L14: ff_2



(p) L15: ff_act_2 (q) L16: ff_3 (r) L17: ff_act_3

Figure 13: Guessing Entropy across all layers of the EffCNN architecture on the ASCADv1f dataset.



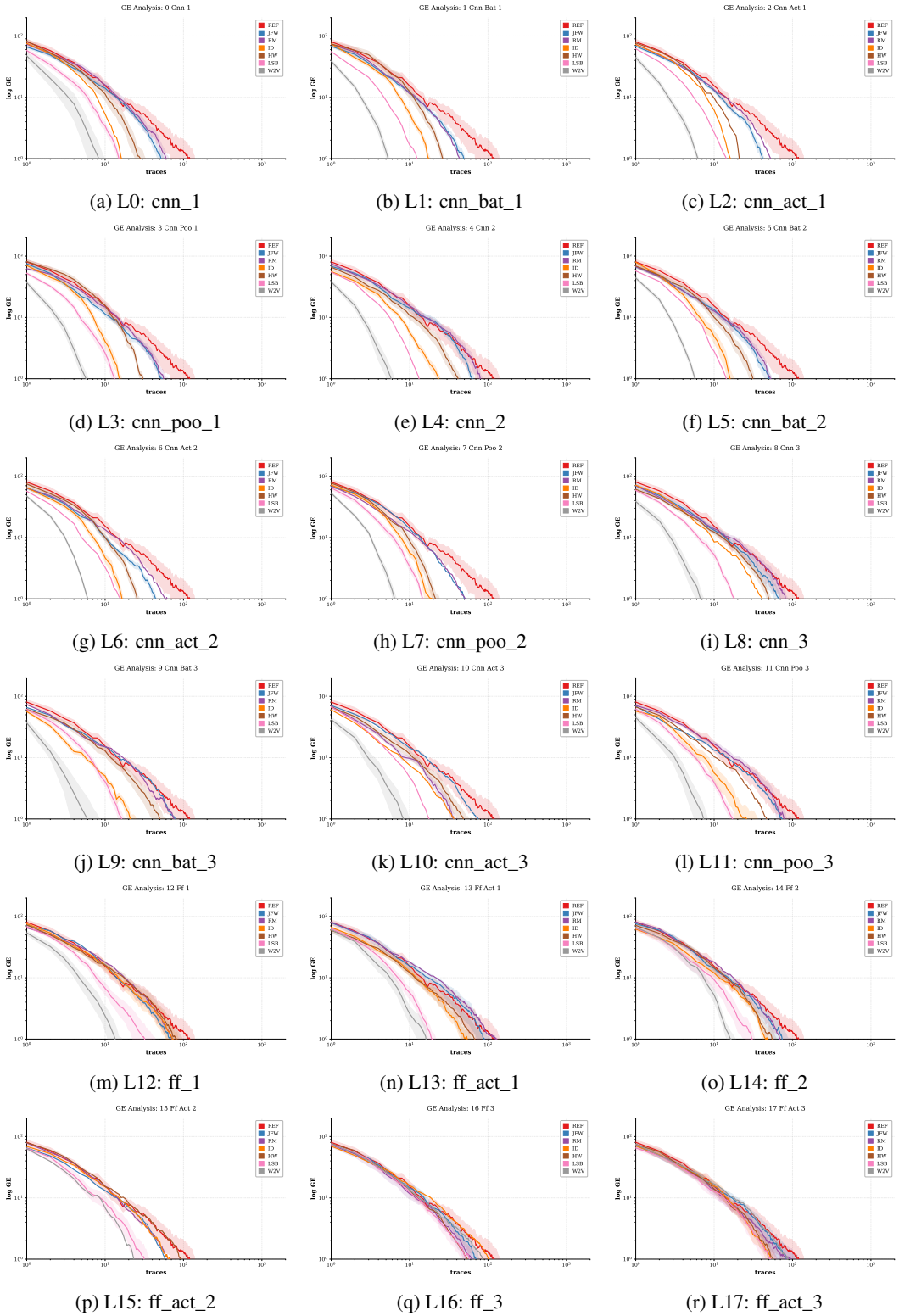
(a) L0: cnn_1 (b) L1: act_1 (c) L2: poo_1 (d) L3: cnn_2

(e) L4: act_2 (f) L5: poo_2 (g) L6: att_lc_1 (h) L7: att_1

(i) L8: pff_lc_1 (j) L9: pff_1 (k) L10: att_lc_2 (l) L11: att_2

(m) L12: pff_lc_2 (n) L13: pff_2 (o) L14: drop (p) L15: sftmx_att

Figure 14: Guessing Entropy across all layers of the EstraNet architecture on the ASCADv1f dataset.



(a) L0: cnn_1 (b) L1: cnn_bat_1 (c) L2: cnn_act_1

(d) L3: cnn_poo_1 (e) L4: cnn_2 (f) L5: cnn_bat_2

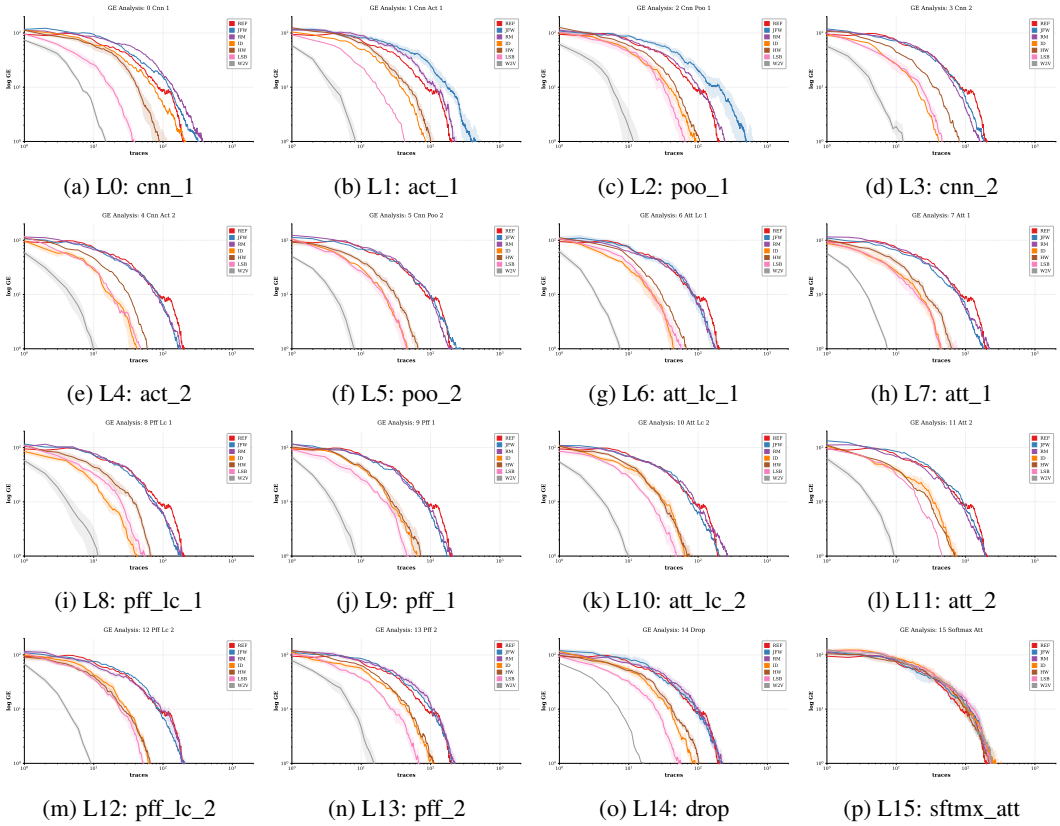
(g) L6: cnn_act_2 (h) L7: cnn_poo_2 (i) L8: cnn_3

(j) L9: cnn_bat_3 (k) L10: cnn_act_3 (l) L11: cnn_poo_3

(m) L12: ff_1 (n) L13: ff_act_1 (o) L14: ff_2

(p) L15: ff_act_2 (q) L16: ff_3 (r) L17: ff_act_3

Figure 15: Guessing Entropy across all layers of the EffCNN architecture on the ASCADv1r dataset.



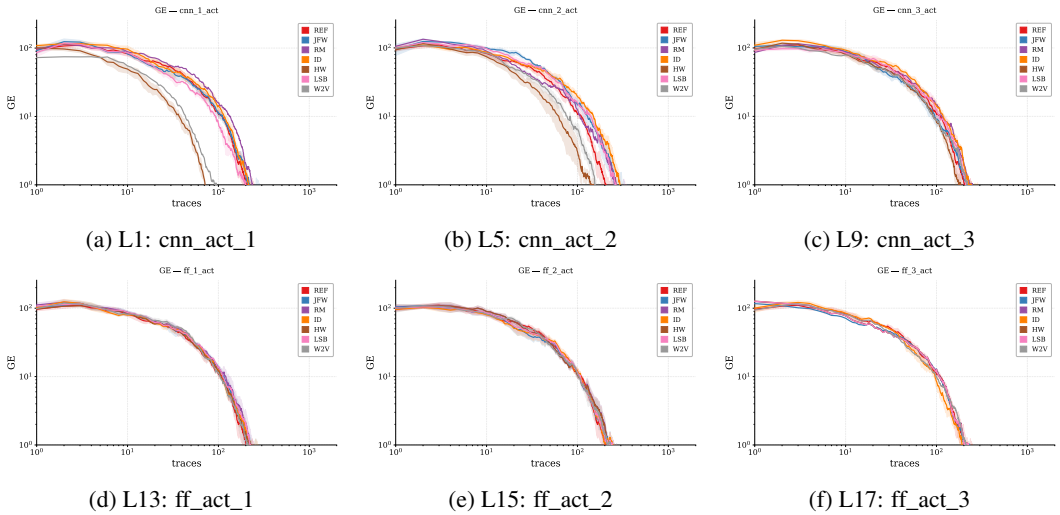
(a) L0: cnn_1 (b) L1: act_1 (c) L2: poo_1 (d) L3: cnn_2

(e) L4: act_2 (f) L5: poo_2 (g) L6: att_lc_1 (h) L7: att_1

(i) L8: pff_lc_1 (j) L9: pff_1 (k) L10: att_lc_2 (l) L11: att_2

(m) L12: pff_lc_2 (n) L13: pff_2 (o) L14: drop (p) L15: sftmx_att

Figure 16: Guessing Entropy across all layers of the EstraNet architecture on the ASCADv1r dataset.



(a) L1: cnn_act_1 (b) L5: cnn_act_2 (c) L9: cnn_act_3

(d) L13: ff_act_1 (e) L15: ff_act_2 (f) L17: ff_act_3

Figure 17: Guessing Entropy across the activation layers of the EffCNN architecture on the non-shuffled ESHARD dataset.