Shape and Substance: Dual-Layer Side-Channel Attacks on Local Vision-Language Models

Eyal Hadad[1,2(✉)][0009−0008−0672−2078] and Mordechai Guri[2][0000−0003−1806−8858]

[1] Achva Academic College, Israel
[2] Ben-Gurion University of the Negev, Israel
eyalhad@post.bgu.ac.il

Abstract. On-device Vision-Language Models (VLMs) promise data privacy via local execution. However, we show that the architectural shift toward Dynamic High-Resolution preprocessing (e.g., AnyRes) introduces an inherent algorithmic side-channel. Unlike static models, dynamic preprocessing decomposes images into a variable number of patches based on their aspect ratio, creating workload-dependent inputs. We demonstrate a dual-layer attack framework against local VLMs. In Tier 1, an unprivileged attacker can exploit significant execution-time variations using standard unprivileged OS metrics to reliably fingerprint the input's geometry. In Tier 2, by profiling Last-Level Cache (LLC) contention, the attacker can resolve structural ambiguity within identical geometries, distinguishing between visually dense (e.g., medical X-rays) and sparse (e.g., text documents) content. By evaluating state-of-the-art models such as LLaVA-NeXT and Qwen2-VL, we show that combining these signals enables the estimation of visual density, which in constrained domains can expose privacy-sensitive contexts. Finally, we analyze the security engineering trade-offs of mitigating this vulnerability, reveal substantial performance overhead with constant-work padding, and propose practical design recommendations for secure Edge AI deployments.

Keywords: Side-channel attacks · Vision-Language Models · Edge AI · Hardware performance counters · Microarchitectural security.

1 Introduction

The deployment of Vision-Language Models (VLMs) is experiencing a significant shift toward local, on-device execution in Edge AI environments [48,23]. The prevailing assumption is that local processing guarantees data privacy. However, if another unprivileged process runs on the same machine, it may still infer sensitive properties of the input data by observing shared microarchitectural behavior [33].

One might ask why an attacker who is already co-located on the victim's device would rely on side channels instead of direct screen capture or memory access. The answer lies in modern operating system security models. Contemporary systems enforce strong address-space isolation and memory randomization

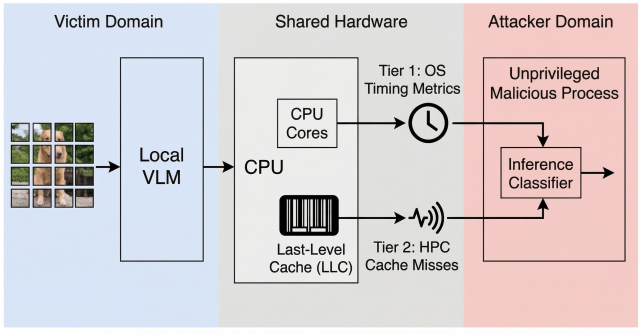


Fig. 1. Dual-layer threat model overview. A co-located, unprivileged process observes shared hardware behavior while a local VLM processes an image with dynamic resolution. The input-dependent workload creates two distinct side-channels: Tier 1 leverages OS-level timing metrics from CPU cores to infer the input's geometric grid (aspect ratio). In contrast, Tier 2 exploits HPC from the LLC to infer the input's visual semantic density.

[36], which prevents malicious applications from directly reading another process's memory or intercepting its input buffer.

However, these systems often expose low-level CPU telemetry, specifically Hardware Performance Counters (HPCs), to user-space processes for debugging and profiling. As a result, while direct access to the input data is blocked, the computational behavior induced by processing that data remains observable through shared hardware resources. This class of leakage has been widely exploited in cryptographic and microarchitectural attacks [13,20,44]. This threat is further amplified by the growing use of unified memory architectures, in which the CPU, GPU, and other accelerators share the same physical memory hierarchy [2]. In such systems, offloading computation to an accelerator does not eliminate side-channel exposure, as memory access patterns remain observable through shared caches and memory resources [29]. Consequently, microarchitectural side channels, including those exploiting DRAM addressing [30], persist even in modern heterogeneous computing platforms.

Figure 1 illustrates our threat model and the attacker's two observation channels.

To understand the threat surface, it is essential to examine the evolution of VLMs. Earlier generations, such as LLaVA v1.5 [27], favored simple and uniform preprocessing by resizing all images to a fixed square resolution (e.g., 336×336) [9]. While computationally wasteful [35], this design normalized the computational workload across inputs.

In contrast, recent models such as LLaVA-NeXT [24] and Qwen2-VL [42] adopt dynamic high-resolution preprocessing to better preserve visual structure. These mechanisms maintain the original aspect ratio by decomposing an image into a variable number of patches (e.g., 1×2 for portrait images and 2×2 for

square images). We argue that this shift fundamentally changes image preprocessing from a fixed, input-agnostic operation into an input-dependent control flow, expanding the observable side-channel surface.

We frame this vulnerability not as a specific implementation bug, but as a broader algorithmic side-channel. This class of leakage is inherent to input-dependent preprocessing pipelines in Edge AI. As models optimize for efficiency by adapting to the input data, they inevitably create data-dependent execution paths. This fundamental trade-off requires a fresh risk assessment for local AI deployments.

Specifically, we show that this architectural choice introduces a dual-layer vulnerability. First, at the control-flow level, execution time directly reflects the number of image patches. This allows an attacker to easily infer the underlying grid configuration and the image's aspect ratio. Second, at the memory-access level, the vision encoder's attention operations [40] create input-dependent cache activity. Images with high visual complexity cause significantly more cache contention than sparse, text-like documents [26]. By combining timing information with cache behavior, an attacker can resolve the ambiguity of a single metric and extract both geometric and semantic information.

Our evaluation demonstrates that this dynamic preprocessing design creates a practical security risk, enabling detailed context inference:

- **Aspect Ratio Inference:** Execution-time variations in dynamic models, such as Qwen2-VL, allow reliable inference of image dimensions due to substantial latency differences across grid configurations.
- **Visual Density Estimation:** By jointly analyzing timing and cache activity, the attack distinguishes between structurally dense and sparse scenarios. Rather than extracting full semantic context, we show that cache contention acts as a probabilistic heuristic for visual complexity, which can identify sensitive content in constrained workflows.
- **Structural vs. Hardware-Dependent Leakage:** We show that geometric leakage arises from the dynamic preprocessing design itself, while semantic leakage depends on hardware characteristics and is mitigated on systems with larger LLC [13].

2 Background

2.1 Local Vision-Language Models

To contextualize the vulnerabilities discussed in this paper, we first outline the architecture of local VLMs. A VLM is composed of a vision encoder (e.g., CLIP [31,39]) with a Vision Transformer backbone [9], a projector [25,3], and an LLM [27]. Unlike cloud deployments (e.g., GPT-4V [1]) where components execute on isolated hardware, local inference setups (e.g., `llama.cpp` [14]) execute the entire pipeline on a single physical device. Consequently, all components share the same memory hierarchy, including the Last-Level Cache (LLC). This shared microarchitectural resource inadvertently exposes the vision encoder's input-dependent execution behavior to co-located processes.

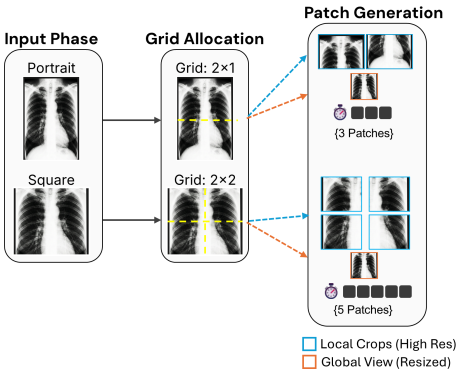


Fig. 2. AnyRes dynamic preprocessing. The model tiles the image into an $(m \times n)$ grid based on aspect ratio and adds a global view, yielding $(m \times n)+1$ patches. Different grids (e.g., 1×2 vs. 2×2) produce different amounts of work, creating a timing signal.

2.2 Dynamic Resolution (AnyRes)

To understand the root cause of the side-channel, it is necessary to examine how modern VLMs process image inputs. Earlier models, such as LLaVA v1.5 [27], relied on static preprocessing, where all images were resized or padded to a fixed square resolution (e.g., 336×336), following standard computer vision practices [16]. This approach resulted in largely input-agnostic computational workloads.

In contrast, recent architectures such as LLaVA-NeXT [24] and Qwen2-VL [42] adopt dynamic high-resolution preprocessing strategies, commonly referred to as AnyRes. Inspired by architectures like NaViT [7], rather than enforcing a fixed input shape, these models adapt to the image's original aspect ratio using a grid-based decomposition. Figure 2 illustrates how AnyRes maps aspect ratios to different grid configurations and patch counts.

To preserve global context that may be lost when processing high-resolution local crops (e.g., observing only part of an object), AnyRes pipelines typically include an additional downsampled global view alongside the local patches. For a given input image, the model selects a grid configuration $(m \times n)$ that minimizes aspect ratio distortion, resulting in a total number of processed patches given by

$$N_{\text{total}} = (m \times n) + 1.$$

For example, a vertical smartphone image commonly triggers a 1×2 grid, yielding three patches (two local crops and one global view), whereas a square document triggers a 2×2 grid, yielding five patches. Since the vision encoder must process every patch, square inputs (yielding five patches) inherently incur a higher computational workload than portrait inputs (yielding three patches). This quantitative difference effectively creates an exploitable algorithmic complexity gap [5].

2.3 Hardware Performance Counters

To bridge the gap between algorithmic complexity and physical execution, we rely on HPCs. While modern operating systems strongly isolate process memory, they often expose low-level CPU telemetry to unprivileged user-space applications for profiling purposes (e.g., via default Linux configurations where `perf_event_paranoid` ≤ 2) [44,21]. Our threat model assumes a co-located, unprivileged attacker exploiting this telemetry. Specifically, the attacker passively monitors hardware events, such as execution time and LLC activity, to measure the distinct microarchitectural signals induced by the workload variation of dynamic image preprocessing [43].

3 Threat Model

Building on these concepts, we now define the formal threat model for our attack, detailing the attacker's capabilities, the system architecture, and the assumptions about the victim's environment.

3.1 System Model and Edge AI Architecture

While traditional evaluations of side-channel vulnerabilities often focus on standard consumer workstations, the rapid deployment of local Vision-Language Models is increasingly driven by dedicated Edge AI systems. To accurately reflect modern deployment scenarios, our system model encompasses both standard heterogeneous architectures (i.e., discrete CPU and GPU) and specialized edge hardware.

In particular, dedicated edge modules (such as NVIDIA Jetson platforms) and emerging Data Processing Units (DPUs) heavily rely on Unified Memory Architectures (UMA) [18]. In these environments, the CPU and the hardware accelerator (GPU or NPU) share not only the same physical memory but often the same memory controllers and LLC hierarchy [10].

Although the computationally intensive matrix multiplications of the VLM are offloaded to the accelerator, the host CPU remains responsible for critical orchestration tasks [28]. Specifically, the dynamic high-resolution preprocessing (e.g., AnyRes patch decomposition), input-dependent control flow decisions, and memory allocation are executed by the OS on the CPU. Because these tasks are fundamentally input-dependent, the CPU's memory access patterns directly reflect the visual complexity and geometric structure of the input image.

Consequently, even in highly optimized Edge AI environments, the shared microarchitectural resources remain a potent source of information leakage. An unprivileged malicious process co-located on the same device can observe this behavior through shared cache contention, effectively bypassing application-level isolation.

3.2 Attacker Capabilities (Adversary Model)

We define the attacker as a malicious native application running in user space on the victim's device. To present a realistic scenario, we assume the attacker runs under the same user account (UID) as the victim but operates in a separate process. The operating system enforces standard address-space isolation, meaning the attacker cannot directly access the victim's memory, read the victim's input images, or capture the victim's screen. Furthermore, the attacker is unprivileged and lacks root (administrator) access.

Because the attacker shares the same physical hardware as the VLM process, we structure our threat model into two distinct tiers of observation capabilities:

1. **Tier 1: Coarse Timing Observation (Zero-Privilege):** The leakage in our system causes large differences in inference execution time, often on the order of tens of seconds. As a result, an attacker does not need access to hardware performance counters. Instead, they can simply measure the inference phase duration using standard OS-level metrics, such as CPU utilization or wall-clock time (e.g., via `/proc/stat` on Linux). This attack requires no special privileges and remains possible even if access to performance telemetry is restricted.
2. **Tier 2: Microarchitectural Profiling (perf):** To infer semantic density, the attacker monitors LLC cache contention. In environments where `perf_event_paranoid` ≤ 2 (a common default), an unprivileged attacker cannot directly monitor the victim's process. Instead, they can employ contention-based techniques (e.g., Prime+Probe or Evict+Time), measuring their *own* LLC misses while the victim induces memory pressure on the shared cache. For the purpose of our experimental evaluation, we configured `perf_event_paranoid = -1` strictly to collect clean "ground truth" measurements and isolate the algorithmic signal. However, the underlying physical contention remains exploitable under standard unprivileged conditions.

Core Isolation (Cross-Core Attack): We assume the attacker and victim are scheduled on separate physical CPU cores. While co-locating them on the same core (e.g., via SMT/Hyperthreading) would make the attack trivial due to shared L1/L2 caches, demonstrating the attack across separate cores proves that the vulnerability survives modern OS core-pinning defenses by exploiting the shared LLC.

The attacker's combined goal is to use Tier 1 (timing) to determine the image's geometry and Tier 2 (cache contention) to resolve its sensitive semantic context.

3.3 Victim Assumptions

The victim is a user or organization utilizing a VLM locally to process sensitive data. The primary motivation for local deployment is privacy, ensuring that data does not leave the physical device.

We assume the following about the victim's environment:

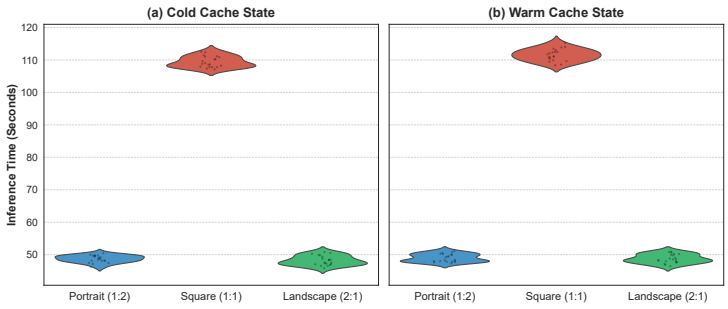


Fig. 3. Deterministic Geometric Leakage. Distribution of inference time across aspect ratios. Square inputs (1:1) incur a $\sim 2.3\times$ latency penalty compared to rectangular inputs, enabling 100% geometric classification regardless of cache state.

- **Software Stack:** The victim uses widely adopted inference frameworks, such as open-source implementations based on `llama.cpp` or PyTorch, to run the VLM.
- **Hardware:** The victim operates on standard consumer or workstation hardware (e.g., Intel or AMD CPUs). While the system may include a GPU, the CPU remains actively involved in input preprocessing, memory management, and control flow decisions of dynamic resolution mechanisms. As a result, shared microarchitectural resources remain a source of information leakage [44].
- **Data Sensitivity and Geometric Correlation:** We assume the VLM is deployed in a domain-specific environment (e.g., an enterprise or medical triage system) where input geometry inherently correlates with sensitive data types. For instance, standard scanned referral letters and patient records are consistently processed in a portrait (A4) aspect ratio, while targeted medical imaging crops (e.g., Chest X-rays) or profile ID photos are frequently cropped to a square ratio. In such constrained workflows, the attacker does not need to guess the format *a priori*; the system's operational constraints dictate these geometric correlations, allowing the attacker to deduce the context of the operation solely from the bounding box.

4 Experimental Evaluation

We evaluate the proposed dual-layer side-channel by first validating the deterministic timing signal (Tier 1) and subsequently analyzing the probabilistic microarchitectural leakage (Tier 2). Finally, we demonstrate how combining these signals enables reliable context inference.

4.1 Experimental Setup & Threat Translation

Evaluations were conducted on an Intel Core i7-13700 (16 cores, 30MB LLC) as the primary platform, and an AMD Ryzen 9 7950X (16 cores, 64MB LLC) for

cross-architecture validation. Both platforms run Ubuntu 24.04 LTS. Our reference implementation utilizes the `llama.cpp` inference engine [14]. To ensure a consistent memory layout and stable execution paths across runs, all experiments were conducted using a fixed commit (`b3456...`) and the `Q4_K_M` quantization scheme, which is the default configuration for high-performance local inference.

Victim & Attacker Isolation: To reduce scheduling-induced variance and enable reproducible attribution of observed leakage to shared microarchitectural resources, we pinned the victim and attacker processes to specific CPU cores. Specifically, the victim process (VLM) was pinned to physical core 4 (`taskset -c 4`), and the attacker process (Profiler) was pinned to physical core 5. This setup isolates the microarchitectural signal from OS scheduler noise, establishing a precise baseline for algorithmic leakage via the shared LLC.

Kernel Configuration: The attack relies solely on unprivileged hardware events (e.g., LLC misses) accessible under the default restrictive Linux configuration (`perf_event_paranoid` ≤ 2). For the purpose of constructing a clean and reproducible ground-truth dataset, we temporarily configured `kernel.perf_event_paranoid = -1` to enable unrestricted access to performance counters for measurement validation.

To isolate the dual-layer leakage, we constructed a synthetic dataset ($N = 250$ per category). It is important to note that this dataset was intentionally designed as an idealized, worst-case scenario to cleanly isolate the two physical variables: execution time (geometry) and cache contention (visual density). To contextualize the attack within realistic environments, we translated base visual categories into high-risk assets: (1) **Medical Report (Portrait 1:2)** representing a sparse text function with low cache contention; (2) **Chest X-Ray (Portrait 1:2)** representing a dense structural function with high cache contention; (3) **Encrypted Data (Square 1:1)** representing a random noise function with low cache contention; and (4) **Technical Schematic (Square 1:1)** representing a dense structural function with high cache contention.

While real-world inputs may exhibit more geometric variance, this controlled design demonstrates how Tier 1 acts as a deterministic filter for the image's geometric family, while Tier 2 acts as a probabilistic heuristic to differentiate visual density within identical geometries.

4.2 Tier 1: Timing Leakage

As established in our threat model, the dynamic grid allocation causes significant variations in the computational workload. As shown in Figure 3, inference time exhibits a clear, deterministic separation across aspect ratios. Square inputs (1:1) incur substantially higher latency than rectangular inputs (portrait or landscape), resulting in non-overlapping timing clusters.

Specifically, square inputs require approximately 110 seconds on average, corresponding to the processing of five patches (four local patches and one global view). In contrast, rectangular inputs complete in approximately 48 seconds, reflecting the processing of only three patches. This yields a latency ratio of

approximately 2.3×. It is important to clarify that the attacker does not mathematically deduce the exact number of patches. Instead, the attacker simply classifies the input into one of the known geometric families (e.g., portrait vs. square) based on which execution-time cluster the inference falls into.

Importantly, this separation persists across cache states. Although absolute execution times are slightly lower in the warm-cache setting, the relative gap and non-overlapping clusters remain unchanged, indicating that caching effects do not mitigate this architectural side-channel.

4.3 Tier 2: Cache Leakage

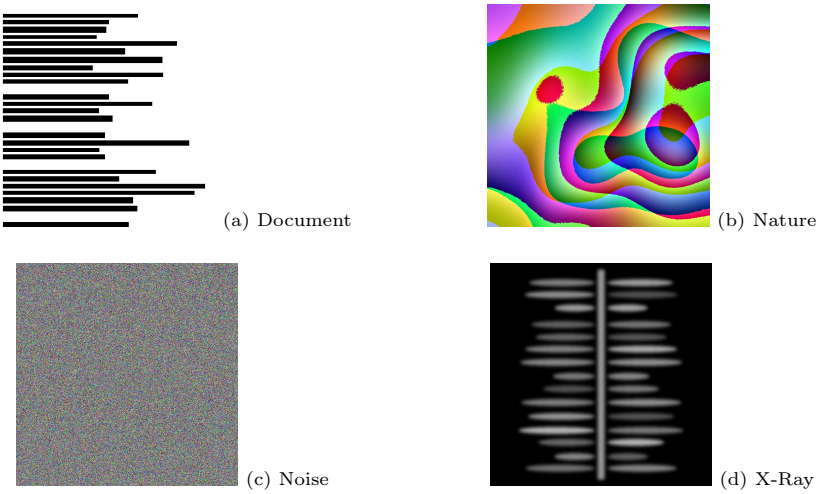
While timing reliably leaks geometry, modern vision encoders also exhibit memory access patterns dependent on visual complexity [6]. Dense visual structures (e.g., X-rays) force the attention mechanism to extract more features, increasing memory pressure compared to sparse documents.

To evaluate whether this semantic information is observable independently of geometry, we analyzed raw hardware metrics for images of identical dimensions (672×672). As detailed below, while execution time remains nearly constant across classes under a warm-cache configuration, LLC misses exhibit a consistent semantic ordering.

Execution Time: As expected, execution time is a strong discriminator for aspect ratio but loses predictive power when the geometry is fixed. Since all inputs trigger the same 2×2 preprocessing grid, the relative execution-time difference between sparse documents and dense X-ray images is below 3%, rendering it indistinguishable from normal system variability.

LLC Miss Behavior: In contrast, LLC miss counts reveal a consistent pattern across content classes. Noise-like inputs exhibit the lowest cache contention, while visually dense and structured images induce higher cache pressure. Figure 4 illustrates representative samples from the evaluated categories, highlighting these differences in spatial frequency and structural density.

- **Lowest Contention (Noise and Documents):** Randomized, noise-like images produce the lowest LLC miss counts (1.69×10^{10}), suggesting relatively uniform memory access. Similarly, documents contain large uniform regions and relatively little fine texture. Although randomness typically increases cache misses, attention-based vision encoders fail to extract meaningful features from noise, resulting in smaller active working sets. This aligns with findings linking input entropy to observable side-channel leakage boundaries [46].
- **Highest Contention (Dense Images):** Structured visual content, such as X-rays and natural scenes, induces higher LLC miss rates ($\approx 1.79 \times 10^{10}$). These categories contain dense visual structure and rich textures across the image. Processing these inputs activates additional parts of the vision encoder, forcing the attention mechanism to actively extract and process a larger number of features across the spatial grid, thereby increasing memory pressure and LLC contention.



(a) Document (b) Nature (c) Noise (d) X-Ray

Fig. 4. Visual Density Comparison (Input Images). Representative samples of the actual input images fed into the vision encoder, illustrating differences in spatial frequency and structural density. (Note: These panels display the raw visual stimuli, not data plots or heatmaps; therefore, they do not contain axis labels). High-contention classes (Nature, X-Ray) exhibit dense textures across the image plane, while low-contention classes (Document, Noise) contain large homogeneous regions with limited fine-grained structure.

4.4 The Combined Attack & Robustness

In Section 4.3, we established that while LLC misses indicate semantic density, using this metric alone is inherently noisy due to dynamic OS memory management and non-deterministic cache replacement policies. Consequently, the LLC miss distributions of visually dense classes exhibit non-negligible overlap. Similarly, Section 4.2 showed that execution time broadly clusters inputs by aspect ratio but cannot resolve content differences within identical geometries. To resolve this ambiguity, we propose a **Combined Attack** vector $\boldsymbol{v} = (t, c)$, jointly analyzing execution time (t) and LLC misses (c). We evaluate this against our two representative scenarios: distinguishing Medical Reports from Chest X-Rays (Portrait cluster), and Encrypted Data from Technical Schematics (Square cluster).

Visualizing the Joint Feature Space. Figure 5 visualizes these scenarios by projecting the collected data into the proposed two-dimensional vector space. The physical behavior of the dynamic VLM explains the resulting distribution: First, the X-axis (Execution Time) cleanly separates the inputs into two distant clusters. This substantial gap occurs because the Portrait inputs (Scenario A) are decomposed into three patches, whereas the Square inputs (Scenario B) are decomposed into five patches. However, within each isolated cluster, the data points overlap horizontally, confirming that timing alone cannot resolve semantic content. Second, the Y-axis (LLC Misses) resolves this ambiguity. Within each

geometry cluster, the data points separate vertically based on visual complexity, allowing an attacker to distinguish dense structures from sparse ones.

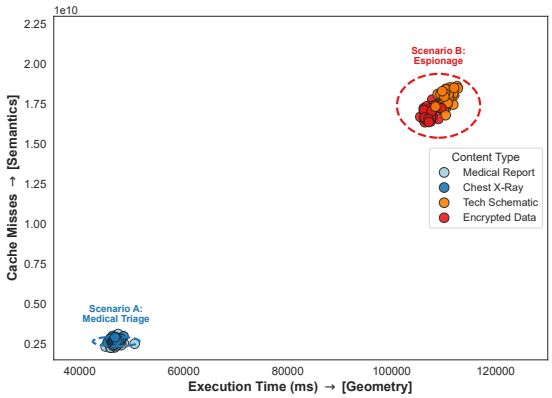


Fig. 5. Joint timing and cache leakage. Projection of the Combined Attack vector into a two-dimensional space. The X-axis (execution time) separates inputs by geometry clusters (portrait vs. square) due to varying patch counts. The Y-axis (LLC misses) resolves the semantic ambiguity within each geometric cluster (e.g., separating dense X-ray images from sparse documents).

Classification Performance. To ensure interpretability and demonstrate that the leakage is systematic rather than incidental, we trained a shallow decision tree (maximum depth of 3), following the methodology of Ribeiro et al. [32]. To rigorously evaluate the model and mitigate the risk of overfitting, we performed a 10-fold cross-validation on the dataset.

The model achieved a robust average accuracy of 83.5% (with a standard deviation of $\pm 1.2\%$) across the cross-validation folds, aligning with an 84.0% accuracy observed on an initial 70%/30% hold-out test set. A closer look at the prediction distribution (Figure 7) reveals the attack's mechanics:

First, as shown in Figure 7, the model never confuses images with different aspect ratios. This shows that the first signal (the overall inference time) separates images by their geometric layout in a fully deterministic manner. In other words, the timing information alone reliably reveals the input's grid structure.

The second signal comes from LLC activity. Unlike the deterministic timing, this microarchitectural signal introduces small variations. However, it is essential for resolving semantic ambiguity, allowing the model to distinguish between visually dense and sparse content within the same geometry.

As explicitly illustrated in Figure 6, the model's extracted decision tree closely follows the physical structure of our threat model: it first splits the data by inference time (capturing the image shape), and then uses cache activity to refine the prediction based on visual content.

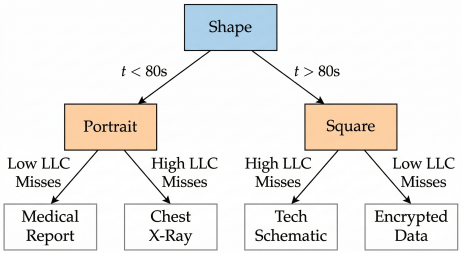


Fig. 6. The Physical Logic of the Attack. The extracted decision tree closely mirrors the proposed dual-layer leakage model. **Layer 1 (Shape):** A fully deterministic root split based on inference time (e.g., $t < 80s$ vs. $t > 80s$) isolates the input's geometric grid into Portrait or Square. **Layer 2 (Substance):** Within each geometry cluster, cache contention levels (Low vs. High LLC Misses) differentiate inputs by visual density. This allows the attacker to reliably isolate dense structures (Chest X-Rays, Tech Schematics) from sparse ones (Medical Reports, Encrypted Data).

Most importantly, the model achieves the highest recall for the most sensitive image categories, correctly identifying 100% of Encrypted Data (F1-score: 0.88) and 93% of Chest X-Rays (F1-score: 0.85). This means that even if some less structured documents are occasionally misclassified, an attacker can still reliably infer highly sensitive visual contexts from the execution of a local VLM.

Cross-Model Validation.

To evaluate whether the observed side-channel is a broader design pattern, we compare LLaVA v1.5 (static) against Qwen2-VL (dynamic). We systematically mapped input aspect ratios to execution times.

As shown in Figure 8, the static model exhibits a nearly constant latency across aspect ratios. In contrast, Qwen2-VL reproduces the step-like timing behavior observed in our primary experiments, where square inputs incur an approximately 60% latency increase compared to rectangular inputs. This consistency across independently developed models indicates that the leakage originates from the dynamic preprocessing algorithm itself. Because the number of processed patches depends on the input grid ($m \times n$), different aspect ratios lead to different control-flow paths, thereby directly affecting inference time. Consequently, the side-channel derives from the fundamental algorithmic design rather than specific model weights or implementation artifacts.

Cross-Architecture Validation

While the previous results established the attack's efficacy on Intel, we now evaluate the influence of cache capacity using an AMD-based system.

We replicated the attack on an AMD Ryzen 9 7950X processor, which has a relatively large LLC. The experiments were conducted using a Python-based framework that runs pre-compiled `llama.cpp` binaries on both the Intel and AMD systems. For each image in the benchmark, the script pins the inference process to a specific CPU core using `taskset` and collects hardware measurements using the Linux `perf` tool. In particular, we record LLC misses and ex-

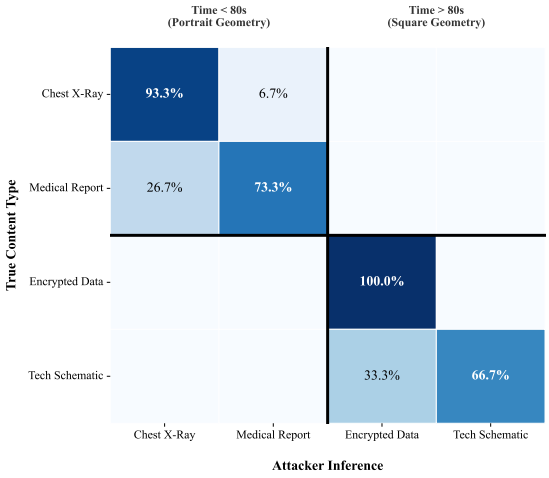


Fig. 7. Confusion Matrix of the Combined Attack. The results visually validate the dual-layer threat model: deterministic timing differences eliminate cross-geometry errors (empty off-diagonal quadrants), while cache-miss telemetry resolves semantic ambiguity within identical geometries. Notably, the attack achieves perfect or near-perfect classification for privacy-critical targets, such as encrypted data and X-ray images.

ecution cycles. This allows us to test whether the observed leakage depends on specific hardware properties or the algorithmic design.

We expect the geometric leakage, which appears as timing differences due to variable-grid processing, to remain visible regardless of cache size, as it is caused by input-dependent control flow in the algorithm. In contrast, cache-based signals such as LLC misses may vary with cache size, because a larger cache can reduce memory contention and thereby affect the number of misses. Table 1 summarizes the results.

As expected, the cross-architecture experiment reveals two different types of leakage: one that remains stable across hardware platforms, and another that depends on microarchitectural characteristics.

1. **Architecture-Independent Leakage (Aspect Ratio):** The timing-based side-channel remains visible on both platforms. Although the AMD processor runs faster overall, the relative latency gap between square and portrait inputs remains clear (1.61×). This suggests that the leakage originates from the input-dependent control flow of the preprocessing algorithm rather than from hardware-specific behavior.
2. **Hardware-Sensitive Leakage (Semantics):** In contrast, semantic leakage is more dependent on hardware characteristics. While clearly observable on the Intel platform, the signal is weaker on the AMD system, presumably due to its larger 64 MB LLC. The larger cache reduces memory contention and dampens the observable cache-miss signal.

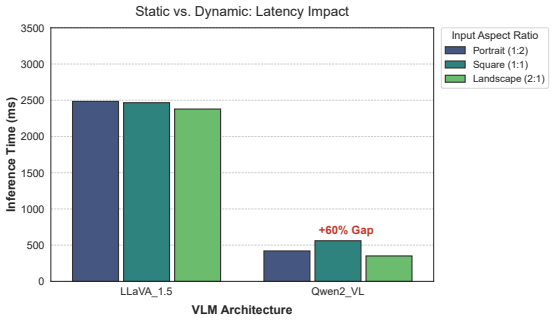


Fig. 8. Cross-Model Comparison. LLaVA v1.5 (static preprocessing) shows similar latency across aspect ratios, whereas Qwen2-VL (dynamic preprocessing) exhibits a clear step-like timing pattern. Square inputs (1 : 1) incur approximately a 60% latency increase compared to rectangular inputs.

Table 1. Cross-Architecture Evaluation. The geometric timing gap remains clear across platforms, reflecting algorithmic control-flow. Conversely, the semantic cache signal is not observable on the AMD architecture due to its larger LLC capacity.

	Intel (30MB)		**AMD (64MB)**	
Evaluation Phase	**Low**	**High**	**Low**	**High**
Time (s)	49	111	18	29
(Portrait vs. Square)	*2.27x Gap*		*1.61x Gap*	
LLC Misses (10^9)	16.9	17.9	4.6	4.6
(Noise vs. X-Ray)	*Observable*		*Not Observable*	

Robustness Under System Load. A common critique of microarchitectural side-channels is their sensitivity to background system activity. To evaluate robustness under realistic noise, we repeated the content-based inference experiment while subjecting the system to controlled synthetic load using `stress-ng`. The stress process was configured to occupy two additional CPU cores and introduce sustained CPU and cache pressure (using the `-cpu` and `-l3-cache` stressors).

As expected, background load increases the absolute LLC miss counts for all inputs, raising the overall noise floor (e.g., average misses for Encrypted Data rose from 16.9×10^9 to 17.1×10^9, and for X-Rays from 17.9×10^9 to 18.1×10^9). However, the **relative separation gap (Δ)** between structurally dense inputs and noise-like inputs remains remarkably stable at approximately 1.0×10^9 misses.

This observation indicates that moderate background load does not eliminate the cache-based separation between content classes. The stability of the separation gap suggests that the vision encoder's memory footprint during the

prompt-evaluation phase dominates cache behavior over short execution intervals, rendering the leakage resilient to typical background activity.

5 Related Work

5.1 Model Extraction vs. Physical Leakage

A significant body of research focuses on stealing machine learning models or extracting sensitive training data via Application Programming Interfaces (APIs). Techniques such as Model Inversion [11] or Membership Inference [37] rely on extensive model querying to reconstruct data from its outputs and confidence scores. In contrast, our work explores physical, passive microarchitectural leakage. While prior studies have successfully demonstrated the extraction of DNN architectures and hyperparameters via hardware side-channels [45], we expand this threat paradigm to the inference data itself. We demonstrate that an attacker does not need API access to the model's outputs. Instead, they can infer properties of the private inputs solely by observing shared hardware resources during local execution.

5.2 Text-Only LLMs vs. Multimodal Encoders

Recent work has shown that LLMs are vulnerable to side-channel attacks. For instance, previous studies have demonstrated that hardware cache behavior can leak information during the token generation phase of local LLMs [12]. Other studies have exploited timing variations in cloud-based serving systems or prompt caching [47,38,15], or targeted specific architectural choices, such as Mixture-of-Experts [8]. However, these attacks focus solely on the text-decoding phase. Our research uniquely targets the vision encoder in multimodal pipelines, demonstrating that processing image data introduces new vulnerabilities before any text is generated.

5.3 Privacy Leaks in User Workloads

Inferring user context from hardware performance is a known threat in applied systems security. Prior research has shown that graphics rendering pipelines and browser activities can leak sensitive information [29]. For example, attackers can use cache-based side channels to recover keystrokes from graphics libraries [41,22] or identify encrypted video streams and websites based on traffic and memory patterns [4,34]. Moreover, recent advances highlight the potency of leveraging LLC telemetry and hardware events to model and detect complex microarchitectural patterns [19], reinforcing the viability of cache activity as a rich semantic signal. Our findings extend this threat model to local AI applications, showing that structural visual complexity translates into a highly reliable side-channel signal.

6 Discussion and Mitigation Challenges

Our findings highlight a fundamental trade-off in Edge AI systems between architectural efficiency and side-channel resilience. Effective mitigation is not trivial and requires addressing both algorithmic control-flow and microarchitectural leakage, often at a steep performance cost.

Application-Level Padding (Tier 1). To eliminate deterministic geometric leakage, the VLM must perform a constant amount of computational work regardless of the input shape (e.g., forcing a worst-case 2×2 grid). However, this incurs a severe performance cost. On our reference Intel platform, processing a portrait image naturally requires 49 seconds, while the worst-case square configuration requires 111 seconds. Enforcing constant-work padding would therefore introduce an approximate 126% increase in inference latency, neutralizing the performance benefits of dynamic resolution.

System-Level Controls & The Hardware Lottery (Tier 2). Mitigating semantic cache leakage requires OS-level interventions, such as restricting access to hardware performance counters or injecting continuous background memory noise, both of which negatively impact overall system throughput. Furthermore, our evaluation reveals a "hardware lottery" [17]: while high-end systems with large shared caches (e.g., 64 MB) incidentally absorb working set variance, resource-constrained Edge AI hardware inherently features significantly smaller LLC capacities. Consequently, semantic side-channel leakage will become even more prominent in dedicated edge deployments, making microarchitectural resilience a mandatory design consideration.

7 Conclusion

The primary motivation for deploying Edge AI and local VLMs is to preserve data privacy. However, we have shown that the architectural shift toward dynamic high-resolution preprocessing introduces a previously overlooked multi-dimensional side-channel. By jointly exploiting deterministic geometric leakage (Tier 1) through execution time and probabilistic semantic leakage (Tier 2) through cache behavior, an unprivileged attacker can infer the geometric structure of visual inputs and use cache contention as a heuristic to estimate their visual density. In tightly constrained operational domains, these two signals combined can reveal highly sensitive contextual properties.

Our evaluation across architectures and system conditions indicates that the geometric component of this leakage stems from the algorithmic design of dynamic preprocessing. In contrast, semantic leakage is influenced by underlying microarchitectural characteristics such as cache capacity.

More broadly, our results highlight that local execution alone does not guarantee privacy on shared hardware. As VLMs are increasingly deployed in sensitive domains—such as medical diagnostics and enterprise operations—this vulnerability allows attackers to infer distinct operational tasks and compromise

user confidentiality. While our evaluation optimally isolated this algorithmic signal, future work should systematically quantify the impact of varied microarchitectural states (e.g., Simultaneous Multithreading (SMT), Non-Uniform Memory Access (NUMA)) and establish exact cache thresholds at which semantic variance becomes unobservable. Ultimately, future edge AI systems and dedicated hardware must treat side-channel resilience as a first-class design consideration.

Acknowledgments. To facilitate reproducibility, the experimental measurement scripts, feature extraction tools, and a representative sample of the dataset have been made publicly available at: https://anonymous.4open.science/r/Shape-and-Substance-Dual-Layer-Side-Channel-Attacks-on-Local-Vision-Language-Models-C6D7.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Achiam, J., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
2. Adiletta, A., Sunar, B.: Spill the beans: Exploiting cpu cache side-channels to leak tokens from large language models. arXiv preprint arXiv:2505.00817 (2025)
3. Alayrac, J.B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al.: Flamingo: a visual language model for few-shot learning. Advances in neural information processing systems **35**, 23716–23736 (2022)
4. Cai, X., Zhang, X.C., Joshi, B., Johnson, R.: Touching from a distance: Website fingerprinting attacks and defenses. In: Proceedings of the 2012 ACM conference on Computer and communications security. pp. 605–616 (2012)
5. Crosby, S.A., Wallach, D.S.: Denial of service via algorithmic complexity attacks. In: 12th USENIX Security Symposium (USENIX Security 03). pp. 29–44 (2003)
6. Dao, T., Fu, D.Y., Ermon, S., Rudra, A., Ré, C.: Flashattention: Fast and memory-efficient exact attention with io-awareness. In: Advances in Neural Information Processing Systems. vol. 35, pp. 16344–16359 (2022)
7. Dehghani, M., Mustafa, B., Djolonga, J., Heek, J., Gilmer, J., et al.: Patch n' pack: Navit, a vision transformer for any aspect ratio and resolution. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 36 (2024)
8. Ding, R., Xu, T., Shen, X., Ding, A.A., Fei, Y.: Moecho: Exploiting side-channel attacks to compromise user privacy in mixture-of-experts llms. In: Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security. pp. 2159–2173 (2025)
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021)
10. Dutta, S.B., Naghibijouybari, H., Abu-Ghazaleh, N., Marquez, A., Barker, K.: Leaky buddies: Cross-component covert channels on integrated cpu-gpu systems. In: 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). pp. 972–984. IEEE (2021)

11. Fredrikson, M., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS). pp. 1322–1333 (2015)
12. Gao, Z., Hu, J., Guo, F., Zhang, Y., Han, Y., Liu, S., Li, H., Lv, Z.: I know what you said: Unveiling hardware cache side-channels in local large language model inference. arXiv preprint arXiv:2505.06738 (2025)
13. Ge, Q., Yarom, Y., Cock, D., Heiser, G.: A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. Journal of Cryptographic Engineering **8**(1), 1–27 (2018)
14. Gerganov, G., contributors: llama.cpp. https://github.com/ggml-org/llama.cpp (2024)
15. Gu, C., Li, X.L., Kuditipudi, R., Liang, P., Hashimoto, T.: Auditing prompt caching in language model apis. arXiv preprint arXiv:2502.07776 (2025)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
17. Hooker, S.: The hardware lottery. Communications of the ACM **64**(12), 58–65 (2021)
18. Horvath, P., Chmielewski, L., Weissbart, L., Batina, L., Yarom, Y.: {BarraCUDA}: Edge {GPUs} do leak {DNN} weights. In: 34th USENIX Security Symposium (USENIX Security 25). pp. 4017–4034 (2025)
19. Kim, H., Hahn, C., Kim, H.J., Shin, Y., Hur, J.: Deep learning-based detection for multiple cache side-channel attacks. IEEE Transactions on Information Forensics and Security **19**, 1672–1686 (2023)
20. Kocher, P., Horn, J., Fogh, A., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., et al.: Spectre attacks: Exploiting speculative execution. In: 2019 IEEE Symposium on Security and Privacy (SP). pp. 1–19. IEEE (2019)
21. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Annual international cryptology conference. pp. 104–113. Springer (1996)
22. Kotcher, R., Pei, Y., Jumde, P., Jackson, C.: Cross-origin pixel stealing: timing attacks using css filters. In: 20th ACM Conference on Computer and Communications Security (CCS). pp. 1055–1062 (2013)
23. Li, E., Zeng, L., Zhou, Z., Chen, X.: Edge ai: On-demand accelerating deep neural network inference via edge computing. IEEE transactions on wireless communications **19**(1), 447–457 (2019)
24. Li, F., Zhang, R., Zhang, H., Zhang, Y., Li, B., Li, W., Ma, Z., Li, C.: Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. arXiv preprint arXiv:2407.07895 (2024)
25. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: International conference on machine learning. pp. 19730–19742. PMLR (2023)
26. Liu, F., Yarom, Y., Ge, Q., Heiser, G., Lee, R.B.: Last-level cache side-channel attacks are practical. In: 2015 IEEE symposium on security and privacy. pp. 605–622. IEEE (2015)
27. Liu, H., Li, C., Li, Y., Lee, Y.J.: Improved baselines with visual instruction tuning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 26296–26306 (2024)

28. Miao, Y., Zhang, Y., Wu, D., Zhang, D., Tan, G., Zhang, R., Kandemir, M.T.: Veiled pathways: Investigating covert and side channels within gpu uncore. in 2024 57th ieee. In: ACM International Symposium on Microarchitecture (MICRO). pp. 1169–1183
29. Naghibijouybari, H., Neupane, A., Qian, Z., Abu-Ghazaleh, N.: Rendered insecure: Gpu side channel attacks are practical. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 2139–2156 (2018)
30. Pessl, P., Gruss, D., Maurice, C., Schwarz, M., Mangard, S.: {DRAMA}: Exploiting {DRAM} addressing for {Cross-CPU} attacks. In: 25th USENIX security symposium (USENIX security 16). pp. 565–581 (2016)
31. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
32. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1135–1144 (2016)
33. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS). pp. 199–212 (2009)
34. Schuster, R., Shmatikov, V., Tromer, E.: Beauty and the burst: Remote identification of encrypted video streams. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 1357–1374 (2017)
35. Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green ai. Communications of the ACM **63**(12), 54–63 (2020)
36. Shacham, H., Page, M., Pfaff, B., Goh, E.J., Modadugu, N., Boneh, D.: On the effectiveness of address-space randomization. In: Proceedings of the 11th ACM conference on Computer and communications security. pp. 298–307 (2004)
37. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE symposium on security and privacy (SP). pp. 3–18. IEEE (2017)
38. Song, L., Pang, Z., Wang, W., Wang, Z., Wang, X., Chen, H., Song, W., Jin, Y., Meng, D., Hou, R.: The early bird catches the leak: Unveiling timing side channels in llm serving systems. IEEE Transactions on Information Forensics and Security (2025)
39. Tschannen, M., Gritsenko, A., Wang, X., Naeem, M.F., Alabdulmohsin, I., Parthasarathy, N., Evans, T., Beyer, L., Xia, Y., Mustafa, B., et al.: Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. arXiv preprint arXiv:2502.14786 (2025)
40. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
41. Wang, D., Neupane, A., Qian, Z., Abu-Ghazaleh, N.: Unveiling your keystrokes: A cache-based side-channel attack on graphics libraries. In: NDSS (2019)
42. Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., et al.: Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. arXiv preprint arXiv:2409.12191 (2024)
43. Yan, M., Fletcher, C.W., Torrellas, J.: Cache telepathy: Leveraging shared resource attacks to learn dnn architectures. In: 29th USENIX Security Symposium (USENIX Security 20). pp. 2003–2020 (2020)

44. Yarom, Y., Falkner, K.: Flush+ reload: A high resolution, low noise, l3 cache side-channel attack. In: 23rd USENIX Security Symposium (USENIX Security 14). pp. 719–732 (2014)
45. Zhang, Y., Yasaei, R., Chen, H., Li, Z., Al Faruque, M.A.: Stealing neural network structure through remote fpga side-channel analysis. IEEE Transactions on Information Forensics and Security **16**, 4377–4388 (2021)
46. Zhang, Z., Ding, A.A., Fei, Y.: A guessing entropy-based framework for deep learning-assisted side-channel analysis. IEEE Transactions on Information Forensics and Security **18**, 3018–3030 (2023)
47. Zheng, X., Han, H., Shi, S., Fang, Q., Du, Z., Hu, X., Guo, Q.: Inputsnatch: Stealing input in llm services via timing side-channel attacks. arXiv preprint arXiv:2411.18191 (2024)
48. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., Zhang, J.: Edge intelligence: Paving the last-mile of artificial intelligence with edge computing. Proceedings of the IEEE **107**(8), 1738–1762 (2019)