# FEATUREBLEED: Inferring Private Enriched Attributes From Sparsity-Optimized AI Accelerators

Darsh Asher, *Graduate Student Member, IEEE*, Farshad Dizani, Joshua Kalyanapu, Rosario Cammarota, Aydin Aysu, *Senior Member, IEEE*, and Samira Mirbagher Ajorpaz, *Member, IEEE*

***Abstract*—Backend enrichment deployed in sensitive domains such as product recommendation pipelines, healthcare, and finance, where models are trained on confidential data and retrieve private features whose values influence inference behavior while remaining hidden from the API caller. This paper presents the first *hardware-level backend retrieval data-stealing attack*, showing that accelerator optimizations designed for performance can directly undermine data confidentiality and bypass state-of-the-art privacy defenses. Our attack, FEATUREBLEED, exploits *zero-skipping* in AI accelerators to infer private backend-retrieved features solely through end-to-end timing, without relying on power analysis, DVFS manipulation, or shared-cache side channels. We evaluate FEATUREBLEED on three datasets spanning medical and non-medical domains—Texas-100X (clinical records), OrganAMNIST (medical imaging), and Census-19 (socioeconomic data). We further evaluate FEATUREBLEED across three hardware backends (Intel AVX, Intel AMX, and NVIDIA A100) and three model architectures (DNNs, CNNs, and hybrid CNN–MLP pipelines), demonstrating that the leakage generalizes across CPU and GPU accelerators, data modalities, and application domains, with an adversarial advantage of up to 98.87 pp. Finally, we identify the root cause of the leakage as sparsity-driven zero-skipping in modern hardware. We quantify the privacy–performance–power trade-off: disabling zero-skipping increases Intel AMX's per-operation energy by up to 25% and incurs 100% performance overhead. We propose a padding-based defense that masks timing leakage by equalizing responses to the worst-case execution time, achieving protection with only 7.24% average performance overhead and no additional power cost is now widely.**

***Index Terms*—Backend enrichment, feature store, activation sparsity, zero-skipping, AI accelerators, timing leakage, machine learning privacy.**

## I. INTRODUCTION

MODERN machine learning (ML) services increasingly operate in settings where the system must make real-time decisions among a large number of possible outcomes, such as selecting content, assessing risk, or generating clinical recommendations. In these settings, the input to inference is rarely limited to what the user explicitly provides. Instead, accurate decisions require *additional contextual information* such as historical behavior, recent activity, or authoritative records that already exist within the service, are private, and must be *retrieved* and *enriched* for the ML network at request time; a widely used example of such systems is Feature Store [1], [2], [6].

Uber Palette [7] within the Michelangelo platform addresses this by pre-computing features offline, and retrieving them during inference. For example, when a rider requests a trip or food delivery, the request contains only basic context (e.g., location, destination, and an entity identifier), while the ML prediction depends on *private*, continuously updated features such as driver acceptance rates, past trip durations, common routes, or restaurant preparation-time statistics. These features are shared across many models, evolve over time, and cannot be re-computed or supplied by the client at inference time. This backend enrichment step is essential for real-time ML systems; without it, predictions would be inaccurate, incur unacceptable latency, or require exposing sensitive internal data to callers.

Recommendation and personalization systems such as NVIDIA's latest real-time recommendation pipelines [5] or Microsoft Copilot also exemplify variants of this design. Large platforms must choose a small set of relevant items from millions of candidates under strict latency constraints. Modern recommender systems retrieve user- and item-specific features from backend storage during inference and use them internally to rank content. This server-side enrichment is essential for *relevance*, *freshness*, and *robustness*, and is a core component of production feeds, ads, and search ranking systems.

A similar architecture underlies credit scoring, underwriting, and fraud detection systems operated by Visa, Mastercard, and FICO, as well as large-scale product recommendation and personalization pipelines deployed by Alibaba, Amazon, and Shopify. These systems must make fast, high-stakes decisions based on sensitive and authoritative information such as transaction history, repayment behavior, device signals, or recent anomalies. Such information cannot be safely or reliably provided by users, as it is subject to manipulation, omission, or staleness. Consequently, risk and underwriting models retrieve features from *trusted backend sources* at decision time and apply *predictive ML models* to estimate default risk or fraud likelihood. Without backend retrieval, these systems would be inaccurate and insecure. A clinical decision support systems and insurance decision systems similarly interact with controlled decision-support services that retrieve authorized summaries and coded records to assess eligibility or coverage, without exposing full clinical records to the insurer. In both settings, the entity issuing the query has legitimate access to the decision-support interface but does not observe the sensitive attributes used internally by the model. Retrieving these attributes server-side is therefore necessary to enable accurate decisions while preserving *privacy* and *regulatory compliance*.

This paper explores the privacy of such systems where *sensitive attributes influence inference behavior while remaining hidden from the API caller*, from the microarchitectural perspective for the first time, specifically by exploiting sparsity-driven zero-skipping optimizations in modern AI accelerators, which are widely implemented in today's hardware. We show for the first time that inference time is distinct for different attribute values of user data. Fig. 1 shows the difference in inference time for a sensitive medical ML workload running on Intel's latest on-core AI accelerator, AMX. We exploit this end-to-end correlation in FEATUREBLEED, to leak user data values used in the training of the ML applications.

An attacker who knows a public or quasi-identifier such as a patient ID or SSN can submit a legitimate inference query associated with that identifier and observe its API response latency. Although sensitive attributes (e.g., whether the individual has a lung or heart condition) are retrieved internally via backend data enrichment and are never exposed through the API, they influence activation sparsity and hardware execution time. As a result, these hidden attributes can be inferred directly

Received 12 January 2026; accepted 15 February 2026. Date of publication 20 February 2026; date of current version 18 March 2026. This work was supported in part by Semiconductor Research Corporation (SRC) under Contract 2025-HW-3306 and in part by Intel Labs. *(Corresponding author: Darsh Asher.)*

Darsh Asher, Farshad Dizani, Joshua Kalyanapu, Aydin Aysu, and Samira Mirbagher Ajorpaz are with the ECE Department, NC State University, Raleigh NC 27606 USA (e-mail: dkasher@ncsu.edu).

Rosario Cammarota is with the UC Irvine, CA USA.

Digital Object Identifier 10.1109/LCA.2026.3666823

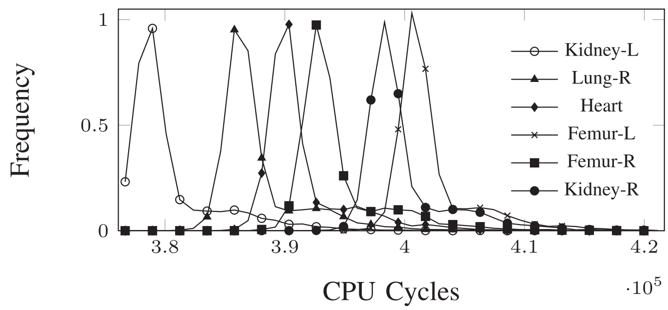


Fig. 1. Normalized CPU-cycle inference-time for the MEDMNIST dataset, showing private diagnosis-dependent separations of up to 30 K cycles (e.g., Kidney-L vs. Kidney-R) and revealing a strong sparsity-induced timing channel.

from the latency of the attacker's own queries, without co-location, shared resources, or timing another user's execution.

To mount FEATUREBLEED, the adversary first profiles inference latency of it's own queries and trains a secondary classifier on the collected timing traces together with the model's returned label and any known (non-sensitive) attributes enabling inferences in systems where state-of-the-art attack strategies [4] fundamentally fail.

These techniques such as attribute inference attacks LOMIA and its confidence-based variant CSMIA [4] is not effective here because (I) first they assume that the attacker can systematically *flip* the sensitive attribute while observing ground-truth labels or confidence scores and, (II) they assume knowledge of the ground-truth output label. These assumptions do not hold in modern production systems where sensitive attributes are retrieved internally via backend data enrichment and never exposed through the API. Consequently, the attacker cannot control the sensitive attribute or construct counterfactual queries that differ only in that attribute. In contrast, our attack does not require *neither (I) flipping hidden attributes nor (II) access to ground-truth labels or (III) confidences*, thus, bypassing common defenses [3], [4].

*Contributions:* This paper makes the following contributions:

- We present FEATUREBLEED, the first hardware-level *backend enrichment attribute-stealing attack* that leaks private attributes exclusively through inference-time side effects, bypassing confidence-masking and label-only defenses. Across Texas100x, Census-19, and OrganAMNIST, FEATUREBLEED enables an adversary to infer sensitive backend-retrieved attributes with high accuracy. For a 10-class surgical procedure inference task, the attack achieves 60.98%, 61.06%, and 70.3% accuracy on AVX, AMX, and NVIDIA A100 backends, respectively, representing an adversarial advantage (AA) up to 98 pp over random guessing.
- We identify the *root cause* of the leakage: data-dependent activation sparsity amplified by accelerator zero-skipping. Reproducing the attack on an NVIDIA A100 GPU reveals operand-dependent latency identical to AMX's behavior and CPU AVX, confirming that the vulnerability generalizes across architectures, even with DVFS locked.
- We analyze mitigation strategies and show that eliminating zero skipping as a defense imposes an overhead of 25% and would double inference latency. We propose a padding defense that equalizes timing leakage across private attributes, with an average performance overhead of 7.24% and no energy overhead.

*Responsible Disclosure:* We disclosed this vulnerability to Intel and NVIDIA, who confirmed our findings while stating that they cannot guarantee privacy against such attacks due to the inherent timing advantage of the optimizations implemented in their products and recommended software-level defenses.

## II. THREAT MODEL

We consider a label-only inference API that returns discrete predictions to the caller. During inference, the model internally retrieves exactly one sensitive attribute from a backend source (e.g., a feature store or database) and incorporates it into the inference pipeline. This sensitive attribute is never exposed through the API and is not part of the client-provided input.

TABLE I
INFERENCE ACCURACY (ACC.), WEIGHTED F1 (W-F1), AND ADVERSARIAL ADVANTAGE (AA) ACROSS ALL SCENARIOS

| Dataset | Attribute | Class | Accuracy | F1 | AA |
|---|---|---|---|---|---|
| OrganAMNIST | CT (Heart) | 11 | 56.08 | 17.62 | +51.69 |
| OrganAMNIST | CT (Femur-L) | 11 | 50.37 | 21.48 | +45.41 |
| OrganAMNIST | CT (Femur-R) | 11 | 48.71 | 20.37 | +43.58 |
| OrganAMNIST | CT (Kidney-L) | 11 | 45.97 | 43.26 | +40.57 |
| OrganAMNIST | CT (Kidney-R) | 11 | 43.27 | 41.34 | +37.60 |
| OrganAMNIST | CT (Liver) | 11 | 43.12 | 43.06 | +37.43 |
| OrganAMNIST | CT (Bladder) | 11 | 11.60 | 14.71 | +2.76 |
| Texas-100X | Surg: C-sec | 10 | 98.98 | 98.71 | +98.87 |
| Texas-100X | Surg: Prostate | 10 | 97.77 | 77.23 | +97.52 |
| Texas-100X | Surg: Appendix | 10 | 81.75 | 88.42 | +79.72 |
| Texas-100X | Surg: Brain | 10 | 74.27 | 82.21 | +71.41 |
| Texas-100X | Surg: Gallbladder | 10 | 74.00 | 65.58 | +71.11 |
| Texas-100X | Surg: Abdomen | 10 | 52.23 | 37.56 | +46.92 |
| Texas-100X | Surg: Derm. | 10 | 30.48 | 32.16 | +22.76 |
| Texas-100X | Surg: Blood Ves. | 10 | 23.83 | 25.19 | +15.37 |
| Census-19 | Marital: Married | 5 | 37.66 | 0.5 | +17.66 |
| Census-19 | Marital: Widowed | 5 | 60.32 | 0.12 | +40.32 |
| Census-19 | Marital: Divorced | 5 | 23.71 | 0.19 | +3.71 |
| Census-19 | Marital: Separated | 5 | 33.12 | 0.04 | +13.12 |
| Census-19 | Marital: Never Married | 5 | 65.59 | 0.70 | +45.59 |

*Attacker:* The attacker in our setting is the entity issuing inference queries to the system, not a separate victim whose execution is being timed. The attacker interacts with the service as a legitimate remote API user, similar to prior black-box settings [4], issues standard inference requests, and measures the end-to-end response latency of *their own* queries. No co-location, shared hardware, or timing of other users' executions is required. The attacker has black-box access only, with no access to model weights, gradients, logits, or confidence scores. FEATUREBLEED exploits class-dependent inference-time gaps of 20–30 K CPU cycles (up to 40 K), which are roughly two orders of magnitude larger than the 50–400 cycle differences used in remote attacks such as HertzBleed, NetSpectre, and GateBleed. As a result, the attack is robust in fully remote settings without co-location or shared resources.

*Target:* The attacker's goal is to infer a sensitive attribute of training data that is never directly present in the client's input but is fetched internally from a backend or feature store during inference (e.g., in retrieval-augmented generation pipelines, offline-to-online feature pipelines, or hospital EHR retrieval). Given non-sensitive inputs, the returned label, and the observed inference latency, the attacker aims to infer the hidden backend attribute of a target individual among $k$ sensitive classes. All measurements are performed remotely by timing the API responses to attacker-issued queries. The timing signal arises solely from the internal execution of the backend-enriched inference pipeline on hardware accelerators and propagates to the observable end-to-end latency. Because the attacker controls the queries they submit, repeated queries used for profiling do not require access to or interaction with any other user.An example is intelligence/military settings, in an online decision service where the request contains only an identifier and minimal context, but the model's prediction requires backend-enriched features pulled from protected databases at inference time such as a target-prioritization API where a client submits a target ID plus current location/time, and the system retrieves classified context such as prior sightings, watchlist status, threat tier, and recent activity from intelligence stores, then runs a ranking model to decide which targets to task next.

## III. FEATUREBLEED

*Step 1. Profiling retrieval-induced timing:* Using auxiliary or synthetic identifiers, the attacker repeatedly queries the system and records inference latency. Each identifier triggers the retrieval of a hidden set of backend attributes, which induces a characteristic activation sparsity pattern and, consequently, a distinct execution-time profile. Over

TABLE II
CHANNEL STRENGTH FOR VARIOUS MODEL SIZES

| Width | Depth | # Params | # Activations | Cohen's $d$ | Accuracy |
|---|---|---|---|---|---|
| 8 | 4 | 336 | 288 | 0.0115 | 50.14% |
| 16 | 4 | 1,184 | 1,088 | 0.0271 | 50.85% |
| 32 | 4 | 4,416 | 4,224 | 0.1045 | 51.35% |
| 64 | 4 | 17,024 | 16,640 | 0.1998 | 53.05% |
| 128 | 2 | 34,048 | 33,024 | 0.0500 | 50.93% |
| 128 | 3 | 50,432 | 49,536 | 0.2800 | 53.96% |
| 128 | 4 | 66,816 | 66,048 | 0.5552 | 63.42% |
| 128 | 5 | 83,200 | 82,560 | 0.9500 | 70.37% |
| 128 | 6 | 99,584 | 99,072 | 0.7800 | 69.17% |
| 128 | 7 | 116,352 | 115,584 | 0.65 | 67.87% |
| 256 | 4 | 264,704 | 263,198 | 0.4767 | 61.61% |
| 512 | 4 | 1,053,696 | 1,050,624 | 4.7800 | 99.37% |

repeated measurements, these profiles form stable timing fingerprints associated with different backend attribute values.

*Step 2. Learning timing clusters for backend attributes:* The attacker clusters the collected timing profiles to identify groups corresponding to different hidden attribute configurations (e.g., high-risk vs. low-risk, genetic marker present vs. absent). When a small labeled subset of identifiers is available through partial knowledge or leakage, the attacker can assign semantic labels to these clusters. To scale to multi-class settings and improve robustness, the attacker trains a Gradient Boosted Decision Tree (GBDT) classifier using inference time, non-sensitive inputs, and predicted labels as features. The attacker is assumed to have access to an auxiliary dataset drawn from the same population but disjoint from the victim model's training set.

*Step 3. Retrieved data inference:* Once the timing clusters are established, the attacker observes the latency of a target query (e.g., associated with a specific identifier) and assigns it to the closest cluster, thereby inferring properties of the retrieved backend data. Because backend retrieval alters activation sparsity and sparsity directly affects execution time on zero-skipping accelerators, this attack bypasses output/confidence masking-based defenses.

## IV. SECURITY ANALYSIS

*Models and Leakage Across Architectures:* We evaluate FEATUREBLEED across **DNNs (tabular inference)**, **CNNs (image-based inference)**, and **hybrid CNN–MLP pipelines**. On NVIDIA A100, DNNs exhibit the strongest leakage, achieving up to 70.3% accuracy, F1 up to 98.7, and adversarial advantage (AA) up to +98.9 pp (percentage points) on Texas-100X. In contrast, CNN-based pipelines on OrganAMNIST leak less on average, with 25.09% accuracy, F1 between 14.7–43.3, and AA up to +51.7 pp, indicating that dense visual representations attenuate but do not eliminate timing leakage. Leakage persists across all model families tested, demonstrating architectural generality.

*Datasets and Attribute Sensitivity:* Across datasets, **Texas-100X** exhibits the highest average leakage, followed by **Census-19** and **OrganAMNIST**. On **Texas-100X**, FeatureBleed achieves an average accuracy of 70.3% over 10 classes with AA exceeding +70 pp for multiple surgical procedures. On **Census-19**, the attack achieves 45.24% average accuracy over 5 marital-status classes with AA up to +45.6 pp. On **OrganAMNIST**, average accuracy is lower (25.09% over 11 classes), yet remains well above random guessing, confirming leakage across data modalities.

*Model Size Impact:* We perform a systematic model-size scaling study, ranging from 336 to 1,053,696 parameters, for fully connected DNNs (Table II). Our results show that increasing model size consistently strengthens the attack. Cohen's $d$ measures the standardized separation between two distributions, capturing how distinguishable they are relative to noise.When the width is increased while keeping the depth fixed, both Cohen's $d$ and attack accuracy grow linearly, indicating that wider models amplify the exploitable leakage signal. In contrast, when depth is varied for a fixed width, Cohen's $d$ and attack accuracy increase up to a depth of five.

TABLE III
INFERENCE ACCURACY ACROSS DIFFERENT ACCELERATORS

| Platform | Proc. Code | Sex | Ethnicity | Race | Admission |
|---|---|---|---|---|---|
| AVX | 60.98% | 58.19% | 74.57% | 23.15% | 41.40% |
| AMX | 61.06% | 61.63% | 74.86% | 20.89% | 43.77% |
| GPU | 70.30% | 66.25% | 80.07% | 43.57% | 50.87% |

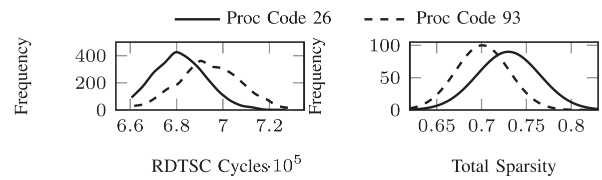


Fig. 2. Timing and sparsity distributions for procedure codes 26 and 93. Left: Inference timing histogram. Right: Total sparsity distribution.

*Hardware Accelerator Generality:* We evaluate FeatureBleed across three widely deployed hardware backends such as**Intel AVX CPUs**, **Intel AMX accelerators**, and **NVIDIA A100 GPUs** as summarized in Table III. The attack consistently infers multiple sensitive attributes across all platforms, demonstrating that the leakage is not tied to a specific microarchitecture. On Intel AVX and AMX, FeatureBleed achieves comparable accuracy for surgical procedure codes (60.98% and 61.06%, respectively), while the A100 exhibits higher leakage, reaching 70.30% accuracy. Similar trends hold across other attributes (sex, ethnicity, race, and source of admission), with GPUs consistently yielding the highest inference accuracy. These results indicate that sparsity-driven timing leakage generalizes across CPU and GPU accelerators, and that more aggressive sparsity optimizations amplify the attack.

## V. ROOT CAUSE ANALYSIS AND MITIGATION

We experimentally confirm that Intel AMX, the on-core accelerator in recent Xeon processors, exhibits zero-skipping behavior where its inference latency scales with operand sparsity, which in turn correlates with specific data attributes. Intel AMX has a special matrix multiplication instruction `TDPBSSD` which we use for ML inference in this paper. This input-dependent latency forms the hardware timing channel that FEATUREBLEED exploits to leak sensitive attributes. We reproduced FEATUREBLEED leakage on the NVIDIA A100 GPU to confirm that zero-skipping optimizations extend beyond Intel AMX. `mma_sync` is a Tensor Core matrix multiply–accumulate instruction on NVIDIA GPUs. It performs warp-level matrix multiplication on Tensor Cores and is used internally by CUDA kernels for ML inference (e.g., GEMM), which exhibits a mean operand-dependent timing difference of 10,000 cycles. This generalizes leakage beyond Intel AMX, though it is weaker due to off-chip data movement.

We measure attribute-dependent activation sparsity and observe that each procedure has a different end-to-end mean sparsity. For example, the procedure code 26 has 4% higher sparsity than the procedure code 93; therefore samples with Procedure code 26 take less time to execute than sample of Procedure code 93 as shown in Fig. 2. Although the distributions partially overlap for some attributes, the sparsity difference is systematic and stable across runs and different datasets and the timing distribution of some attributes completely separates (see Fig. 1) resulting in up to 98% accuracy (see Table I) for example Surgical Procedure Code of C-Section in Texas100x, or heart vs kidney in MEDMNIST dataset.

Thus, sensitive backend retrieved data with varied sparsity creates a novel, observable side-channel. We show this correlation between class vs. sparsity and sparsity vs. time, leads to the exploited correlation between class vs. time and also confirms the presence of zero skipping with energy analysis. Thus, we conclude that the root cause of this leakage is

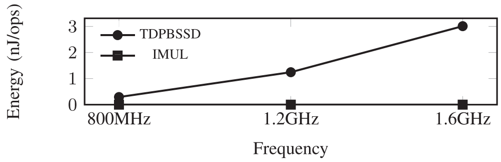


Fig. 3. Energy gap between operands 0 and 255 across fixed CPU frequencies. AMX TDPBSSD remains operand-dependent even when DVFS is locked.

two-fold: (1) data features correlate with sparsity due to representation learning in NNs, and ReLU further amplifies this sparsity by zeroing negative values, and (2) sparsity correlates with execution time due to zero skipping in accelerators in hardware. Therefore, variations in runtime can inadvertently leak information about the retrieved backend data through side-channel timing analysis, revealing sensitive attributes of the data.

Our results show disabling TurboBoost/DVFS prevents frequency-scaling leaks [8] but not FEATUREBLEED; the energy gap between 0 and 255 operands (Fig. 3) grows even under frequency lock, confirming a zero-skipping effect. Therefore, if we disable zero-skipping, the average energy of TDPBSSD operations with all-zero operands is 12 nJ versus all-one operands is 15 nJ, yielding an increase of $[(15nJ - 12nJ)/12nJ] * 100 \approx 25\%$ energy per operation. Defenses that clip or randomize confidences [3], [4] remove statistical cues, but not timing leakage; FEATUREBLEED remains effective since latency still correlates with sparsity.

We propose a padding defense in which the API withholds the response until a fixed budget equal to the worst-case end-to-end inference time elapses, masking timing variations without performing extra computation. The additional energy is largely limited to idle power during the padding interval. Average performance overhead across three hardware on Texas-100X (80–90% sparsity) is 7.24%.

## VI. CONCLUSION

We show that microarchitectural optimizations in AI accelerators, such as zero-skipping, create new timing channels that can leak private backend-retrieved attributes in ML pipelines. Future work should pursue hardware–software co-design defenses that mitigate such rising leakages while preserving the performance and energy benefits of accelerator optimizations.

## REFERENCES

[1] J. de la Rúa Martıńez et al., "The hopsworks feature store for machine learning," in *Proc. Companion 2024 Int. Conf. Manage. Data*, 2024, pp. 135–147.
[2] Feast Project Contributors, "Feast: The open source feature store for machine learning," 2025. Accessed: Dec. 29, 2025. [Online]. Available: https://feast.dev
[3] J. Jia and N. Z. Gong, "AttriGuard: A practical defense against attribute inference attacks via adversarial machine learning," in *Proc. 27th USENIX Secur. Symp.*, Baltimore, MD, USA: USENIX Association, Aug. 2018, pp. 513–529. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/jia-jinyuan
[4] S. Mehnaz, S. V. Dibbo, E. Kabir, N. Li, and E. Bertino, "Are your sensitive attributes private? Novel model inversion attribute inference attacks on classification models," in *Proc. 31st USENIX Secur. Symp.*, Boston, MA, USA: USENIX Association, Aug. 2022, pp. 4579–4596. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/mehnaz
[5] NVIDIA, "Offline-to-online feature storage for real-time recommendation systems," 2023. [Online]. Available: https://developer.nvidia.com/blog/offline-to-online-feature-storage-for-real-time-recommendation-systems-with-nvidia-merlin/
[6] L. Orr, A. Sanyal, X. Ling, K. Goel, and M. Leszczynski, "Managing ML pipelines: Feature stores and the coming wave of embedding ecosystems," *Proc. VLDB Endowment*, vol. 14, no. 12, pp. 3178–3181, 2021, doi: 10.14778/3476311.3476402.
[7] Uber Engineering, "From predictive to generative–How Michelangelo accelerates Uber's AI journey," May 2024. Accessed: Dec. 30, 2025. [Online]. Available: https://www.uber.com/blog/from-predictive-to-generative-ai/
[8] Y. Wang et al., "Hertzbleed: Turning power side-channel attacks into remote timing attacks on x86," in *Proc. USENIX Secur. Symp.*, 2022, pp. 679–697.